Courier Management System

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

---

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

23. Retrieve Payments with Courier Information

24. Retrieve Payments with Location Information

25. Retrieve Payments with Courier and Location Information

26. List all payments with courier details

27. Total payments received for each courier

28. List payments made on a specific date

29. Get Courier Information for Each Payment

30. Get Payment Details with Location

31. Calculating Total Payments for Each Courier

32. List Payments Within a Date Range

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

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

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

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

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

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

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

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

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

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

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

44. List all employees who share the same role.

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

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

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

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

---

This document provides SQL queries for various operations on the Courier Management System database.

These queries were executed on the schema designed in Task 1.

23. Retrieve Payments with Courier Information

This will return all payments along with their corresponding courier details.

```
SELECT p.PaymentID, p.CourierID, p.Amount, p.PaymentDate,
    c.SenderName, c.ReceiverName, c.Cost
FROM Payment p
INNER JOIN Courier c ON p.CourierID = c.CourierID;
```

Explanation :

- It **only** returns payments that have a matching courier.

Output :

PaymentID	CourierID	Amount	PaymentDate	SenderName	ReceiverName	Weight	Status	TrackingNumber	DeliveryDate
1	1	100.00	2024-03-21	John Doe	Alice Smith	2.50	In Transit	TRK10001	2024-03-20
2	2	250.00	2024-03-19	Alice Smith	Michael Brown	1.20	Delivered	TRK10002	2024-03-18
3	3	500.00	2024-03-22	Michael Brown	Emma Wilson	3.00	Pending	TRK10003	2024-03-22
4	4	400.00	2024-03-20	Emma Wilson	Daniel Lee	2.80	In Transit	TRK10004	2024-03-21
5	5	800.00	2024-03-18	Daniel Lee	Sophia Green	4.10	Shipped	TRK10005	2024-03-19
6	6	300.00	2024-03-23	Sophia Green	William Clark	1.90	Delivered	TRK10006	2024-03-17
7	7	150.00	2024-03-17	William Clark	Olivia Adams	2.20	Pending	TRK10007	2024-03-23
8	8	600.00	2024-03-25	Olivia Adams	James Turner	3.50	Shipped	TRK10008	2024-03-20
9	9	350.00	2024-03-24	James Turner	Charlotte Evans	2.70	In Transit	TRK10009	2024-03-25
10	10	1000.00	2024-03-26	Charlotte Evans	John Doe	5.00	Pending	TRK10010	2024-03-24
11	11	110.00	2024-03-27	Ethan Carter	Isabella Lewis	3.20	In Transit	TRK10011	2024-03-26
12	12	260.00	2024-03-25	Isabella Lewis	Liam Parker	2.50	Delivered	TRK10012	2024-03-24
13	13	150.00	2024-03-29	Liam Parker	Ava Martinez	4.10	Pending	TRK10013	2024-03-28
14	14	500.00	2024-03-28	Ava Martinez	Mason Scott	1.80	In Transit	TRK10014	2024-03-27

---

24. Retrieve Payments with Location Information

This query retrieves all payments along with their corresponding location details.

```
SELECT p.PaymentID, p.CourierID, p.LocationId, p.Amount, p.PaymentDate,
l.LocationName, l.Address
FROM Payment p
INNER JOIN Location l ON p.LocationId = l.LocationID;
```

Explanation :

It ensures that only payments linked to a valid location appear in the result.

Output :

PaymentID	CourierID	LocationId	Amount	PaymentDate	LocationName	Address
1	1	1	100.00	2024-03-21	Warehouse A	789 Industrial Road
2	2	2	250.00	2024-03-19	Warehouse B	321 Commercial St
3	3	3	500.00	2024-03-22	Hub Center	555 Logistics Park
4	4	4	400.00	2024-03-20	Main Office	100 Business Ave
5	5	5	800.00	2024-03-18	Sorting Facility 1	222 Shipping Lane
6	6	6	300.00	2024-03-23	Sorting Facility 2	333 Distribution St
7	7	7	150.00	2024-03-17	Retail Store 1	444 Market Square
8	8	8	600.00	2024-03-25	Retail Store 2	555 Plaza Blvd
9	9	9	350.00	2024-03-24	Drop-off Point A	666 Highway Exit 1
10	10	10	1000.00	2024-03-26	Drop-off Point B	777 Main St
11	11	1	110.00	2024-03-27	Warehouse A	789 Industrial Road
12	12	2	260.00	2024-03-25	Warehouse B	321 Commercial St
13	13	3	150.00	2024-03-29	Hub Center	555 Logistics Park
14	14	4	500.00	2024-03-28	Main Office	100 Business Ave
15	15	5	700.00	2024-03-26	Sorting Facility 1	222 Shipping Lane

---

25. Retrieve Payments with Courier and Location Information

This query retrieves payments along with their corresponding courier and location details

```
SELECT p.PaymentID, p.CourierID, p.LocationId, p.Amount, p.PaymentDate,
c.SenderName, c.ReceiverName, c.TrackingNumber, c.status ,
l.LocationName, l.Address
FROM Payment p
INNER JOIN Courier c ON p.CourierID = c.CourierID
INNER JOIN Location l ON p.LocationId = l.LocationID;
```

Explanation:

This query joins the Payment, Courier, and Location tables. INNER JOIN ensures that only records where a payment is linked to both a valid courier and a valid location are retrieved.

Output :

PaymentID	CourierID	LocationId	Amount	PaymentDate	SenderName	ReceiverName	TrackingNumber	status	LocationName	Address
1	1	1	100.00	2024-03-21	John Doe	Alice Smith	TRK10001	In Transit	Warehouse A	789 Industrial Road
2	2	2	250.00	2024-03-19	Alice Smith	Michael Brown	TRK10002	Delivered	Warehouse B	321 Commercial St
3	3	3	500.00	2024-03-22	Michael Brown	Emma Wilson	TRK10003	Pending	Hub Center	555 Logistics Park
4	4	4	400.00	2024-03-20	Emma Wilson	Daniel Lee	TRK10004	In Transit	Main Office	100 Business Ave
5	5	5	800.00	2024-03-18	Daniel Lee	Sophia Green	TRK10005	Shipped	Sorting Facility 1	222 Shipping Lane
6	6	6	300.00	2024-03-23	Sophia Green	William Clark	TRK10006	Delivered	Sorting Facility 2	333 Distribution St
7	7	7	150.00	2024-03-17	William Clark	Olivia Adams	TRK10007	Pending	Retail Store 1	444 Market Square
8	8	8	600.00	2024-03-25	Olivia Adams	James Turner	TRK10008	Shipped	Retail Store 2	555 Plaza Blvd
9	9	9	350.00	2024-03-24	James Turner	Charlotte Evans	TRK10009	In Transit	Drop-off Point A	666 Highway Exit 1
10	10	10	1000.00	2024-03-26	Charlotte Evans	John Doe	TRK10010	Pending	Drop-off Point B	777 Main St

---

26. List all payments with courier details

This query retrieves all payment records along with their associated courier details.

```
SELECT p.PaymentID, p.CourierID, p.Amount, p.PaymentDate,
c.SenderName, c.ReceiverName, c.TrackingNumber, c.Status , c.DeliveryDate
FROM Payment p
LEFT JOIN Courier c ON p.CourierID = c.CourierID;
```

Explanation:

This query uses a LEFT JOIN, meaning it retrieves all payment records, even if there's no matching courier. If a payment exists but doesn't have a corresponding courier, the courier fields will show NULL.

Output :

PaymentID	CourierID	Amount	PaymentDate	SenderName	ReceiverName	TrackingNumber	Status	DeliveryDate
1	1	100.00	2024-03-21	John Doe	Alice Smith	TRK10001	In Transit	2024-03-20
2	2	250.00	2024-03-19	Alice Smith	Michael Brown	TRK10002	Delivered	2024-03-18
3	3	500.00	2024-03-22	Michael Brown	Emma Wilson	TRK10003	Pending	2024-03-22
4	4	400.00	2024-03-20	Emma Wilson	Daniel Lee	TRK10004	In Transit	2024-03-21
5	5	800.00	2024-03-18	Daniel Lee	Sophia Green	TRK10005	Shipped	2024-03-19
6	6	300.00	2024-03-23	Sophia Green	William Clark	TRK10006	Delivered	2024-03-17
7	7	150.00	2024-03-17	William Clark	Olivia Adams	TRK10007	Pending	2024-03-23
8	8	600.00	2024-03-25	Olivia Adams	James Turner	TRK10008	Shipped	2024-03-20
9	9	350.00	2024-03-24	James Turner	Charlotte Evans	TRK10009	In Transit	2024-03-25
10	10	1000.00	2024-03-26	Charlotte Evans	John Doe	TRK10010	Pending	2024-03-24
11	11	110.00	2024-03-27	Ethan Carter	Isabella Lewis	TRK10011	In Transit	2024-03-26
12	12	260.00	2024-03-25	Isabella Lewis	Liam Parker	TRK10012	Delivered	2024-03-24

---

27. Total payments received for each courier

This query calculates the total amount of payments received for each courier.

```
SELECT p.CourierID, c.SenderName, c.ReceiverName, c.TrackingNumber,
SUM(p.Amount) AS TotalPayment
FROM Payment p
INNER JOIN Courier c ON p.CourierID = c.CourierID
GROUP BY p.CourierID, c.SenderName, c.ReceiverName, c.TrackingNumber;
```

Explanation:

INNER JOIN ensures only couriers with payments are included. GROUP BY groups payments by CourierID, ensuring that payments are summed per courier. SUM(p.Amount) calculates the total payments for each courier.

Output :

CourierID	SenderName	ReceiverName	TrackingNumber	TotalPayment
1	John Doe	Alice Smith	TRK10001	319.25
2	Alice Smith	Michael Brown	TRK10002	512.55
3	Michael Brown	Emma Wilson	TRK10003	709.99
4	Emma Wilson	Daniel Lee	TRK10004	520.50
5	Daniel Lee	Sophia Green	TRK10005	885.75
6	Sophia Green	William Clark	TRK10006	450.30
7	William Clark	Olivia Adams	TRK10007	150.00
8	Olivia Adams	James Turner	TRK10008	600.00
9	James Turner	Charlotte Evans	TRK10009	350.00
10	Charlotte Evans	John Doe	TRK10010	1000.00
11	Ethan Carter	Isabella Lewis	TRK10011	110.00
12	Isabella Lewis	Liam Parker	TRK10012	260.00
13	Liam Parker	Ava Martinez	TRK10013	150.00
14	Ava Martinez	Mason Scott	TRK10014	500.00

---

28. List payments made on a specific date

This query retrieves all payments made on a given date.

```
SELECT p.PaymentID, p.CourierID, p.LocationId, p.Amount, p.PaymentDate,
    c.SenderName, c.ReceiverName, c.TrackingNumber
FROM Payment p
INNER JOIN Courier c ON p.CourierID = c.CourierID
WHERE p.PaymentDate = '2025-03-18';
```

Explanation:

- **INNER JOIN** links the Payment table with the Courier table to get additional courier details.
- **WHERE p.PaymentDate = '2025-03-17'** filters payments made on March 17, 2025.

Output :

PaymentID	CourierID	LocationId	Amount	PaymentDate	SenderName	ReceiverName	TrackingNumber
22	1	2	98.75	2025-03-18	John Doe	Alice Smith	TRK10001
27	3	4	99.99	2025-03-18	Michael Brown	Emma Wilson	TRK10003
31	61	10	3500.00	2025-03-18	Alice	Bob	TRK0031

---

29. Get Courier Information for Each Payment

This query retrieves all payments along with their corresponding courier details.

```
SELECT p.PaymentID, p.CourierID, p.LocationId, p.Amount, p.PaymentDate,
```

c.SenderName, c.SenderAddress, c.ReceiverName, c.ReceiverAddress,

c.Weight, c.Status, c.TrackingNumber, c.DeliveryDate

FROM Payment p

INNER JOIN Courier c ON p.CourierID = c.CourierID;

Explanation :

- INNER JOIN is used to fetch courier details for each payment.
- Retrieves payment-related information (PaymentID, CourierID, LocationId, Amount, PaymentDate).
- Includes courier details such as (SenderName, SenderAddress, ReceiverName, ReceiverAddress, Weight, Status, TrackingNumber, DeliveryDate).
- Ensures that only payments with valid courier records are shown.

Output :

Payment	Courier	LocationId	Amount	PaymentDate	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate
1	1	1	100.00	2024-03-21	John Doe	123 Main St	Alice Smith	456 Elm St	2.50	In Transit	TRK10001	2024-03-20
2	2	2	250.00	2024-03-19	Alice Smith	456 Elm St	Michael Brown	789 Oak St	1.20	Delivered	TRK10002	2024-03-18
3	3	3	500.00	2024-03-22	Michael Brown	789 Oak St	Emma Wilson	321 Pine St	3.00	Pending	TRK10003	2024-03-22
4	4	4	400.00	2024-03-20	Emma Wilson	321 Pine St	Daniel Lee	654 Cedar St	2.80	In Transit	TRK10004	2024-03-21
5	5	5	800.00	2024-03-18	Daniel Lee	654 Cedar St	Sophia Green	987 Maple St	4.10	Shipped	TRK10005	2024-03-19
6	6	6	300.00	2024-03-23	Sophia Green	987 Maple St	William Clark	258 Birch St	1.90	Delivered	TRK10006	2024-03-17
7	7	7	150.00	2024-03-17	William Clark	258 Birch St	Olivia Adams	369 Walnut St	2.20	Pending	TRK10007	2024-03-23
8	8	8	600.00	2024-03-25	Olivia Adams	369 Walnut St	James Turner	741 Cherry St	3.50	Shipped	TRK10008	2024-03-20
9	9	9	350.00	2024-03-24	James Turner	741 Cherry St	Charlotte Evans	852 Willow St	2.70	In Transit	TRK10009	2024-03-25
10	10	10	1000.00	2024-03-26	Charlotte Evans	852 Willow St	John Doe	123 Main St	5.00	Pending	TRK10010	2024-03-24

30. Get Payment Details with Location

This query retrieves all payments along with their corresponding location details.

SELECT p.PaymentID, p.CourierID, p.Amount, p.PaymentDate,

l.LocationID, l.LocationName, l.Address

FROM Payment p

INNER JOIN Location l ON p.LocationID = l.LocationID;

Explanation:

INNER JOIN is used to fetch location details for each payment.

Retrieves payment-related information (PaymentID, CourierID, Amount, PaymentDate).

Includes location details such as (LocationID, LocationName, City, State).

Ensures only payments with valid location records are displayed.

Output :

Payment	Courier	Amount	PaymentDate	LocationID	LocationName	Address
1	1	100.00	2024-03-21	1	Warehouse A	789 Industrial Road
2	2	250.00	2024-03-19	2	Warehouse B	321 Commercial St
3	3	500.00	2024-03-22	3	Hub Center	555 Logistics Park
4	4	400.00	2024-03-20	4	Main Office	100 Business Ave
5	5	800.00	2024-03-18	5	Sorting Facility 1	222 Shipping Lane
6	6	300.00	2024-03-23	6	Sorting Facility 2	333 Distribution St
7	7	150.00	2024-03-17	7	Retail Store 1	444 Market Square
8	8	600.00	2024-03-25	8	Retail Store 2	555 Plaza Blvd
9	9	350.00	2024-03-24	9	Drop-off Point A	666 Highway Exit 1
10	10	1000.00	2024-03-26	10	Drop-off Point B	777 Main St
11	11	110.00	2024-03-27	1	Warehouse A	789 Industrial Road
12	12	260.00	2024-03-25	2	Warehouse B	321 Commercial St

31. Calculating Total Payments for Each Courier

This query calculates the total amount received for each courier.

```
SELECT p.CourierID, SUM(p.Amount) AS TotalAmount
FROM Payment p
GROUP BY p.CourierID
ORDER BY TotalAmount DESC;
```

Explanation:

- **GROUP BY p.CourierID** ensures we calculate the total amount per courier.
- **SUM(p.Amount) AS TotalAmount** sums up all payments received for each courier.
- **ORDER BY TotalAmount DESC** sorts the results in descending order to show the highest-paid couriers first.

Output:

Courier	TotalAmount
64	4500.00
65	4100.00
63	4000.00
61	3500.00
62	3200.00
19	1500.00
20	1200.00
10	1000.00
17	900.00
5	885.75
3	709.99
15	700.00
8	600.00
18	550.00
4	520.50
2	512.55
14	500.00
6	450.30
9	350.00
1	319.25
16	300.00
12	260.00
7	150.00
13	150.00
11	110.00

32. List Payments Within a Date Range

This query retrieves all payments made within a specific date range.

```
SELECT *
FROM Payment
WHERE PaymentDate BETWEEN '2025-03-15' AND '2025-03-25';
```

Explanation:

- **WHERE PaymentDate BETWEEN '2025-03-15' AND '2025-03-25'** filters payments that occurred within the given date range.
- This helps track payments over a specific period for analysis.

Output :

Payment	Courier	LocationID	Amount	PaymentDate
21	1	1	120.50	2025-03-17
22	1	2	98.75	2025-03-18
23	2	3	85.30	2025-03-19
24	2	4	102.25	2025-03-20
25	3	5	110.00	2025-03-21
26	2	3	75.00	2025-03-17
27	3	4	99.99	2025-03-18
28	4	5	120.50	2025-03-19
29	5	1	85.75	2025-03-20
30	6	2	150.30	2025-03-21
31	61	10	3500.00	2025-03-18
32	62	9	3200.00	2025-03-19

---

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

This requires a **FULL OUTER JOIN** between the User table and the Courier table to include all users and couriers, even if they don't have a match.

MySQL **does not support FULL OUTER JOIN** directly. Instead,we can achieve the same result using a **UNION** of **LEFT JOIN** and **RIGHT JOIN**.

```
SELECT U.UserID, U.Name,U.ContactNumber, C.CourierID, C.TrackingNumber, C.Status , C.DeliveryDate
FROM User U
LEFT JOIN Courier C ON U.UserID = C.SenderID
UNION
SELECT U.UserID, U.Name,U.ContactNumber, C.CourierID, C.TrackingNumber, C.Status , C.DeliveryDate
```

FROM User U

RIGHT JOIN Courier C ON U.UserID = C.SenderID;

Explanation

- **LEFT JOIN** ensures that all users are included, even if they haven't sent a courier.
- **RIGHT JOIN** ensures that all couriers are included, even if they aren't linked to a user.
- **UNION** combines both results to simulate a **FULL OUTER JOIN**.

Output :

CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	EmployeeID	SenderID
1	John Doe	123 Main St	Alice Smith	456 Elm St	2.50	In Transit	TRK10001	2024-03-20	1	1
2	Alice Smith	456 Elm St	Michael Brown	789 Oak St	1.20	Delivered	TRK10002	2024-03-18	1	2
3	Michael Brown	789 Oak St	Emma Wilson	321 Pine St	3.00	Pending	TRK10003	2024-03-22	2	3
4	Emma Wilson	321 Pine St	Daniel Lee	654 Cedar St	2.80	In Transit	TRK10004	2024-03-21	2	4
5	Daniel Lee	654 Cedar St	Sophia Green	987 Maple St	4.10	Shipped	TRK10005	2024-03-19	3	5
6	Sophia Green	987 Maple St	William Clark	258 Birch St	1.90	Delivered	TRK10006	2024-03-17	3	6
7	William Clark	258 Birch St	Olivia Adams	369 Walnut St	2.20	Pending	TRK10007	2024-03-23	4	7
8	Olivia Adams	369 Walnut St	James Turner	741 Cherry St	3.50	Shipped	TRK10008	2024-03-20	4	8
9	James Turner	741 Cherry St	Charlotte Evans	852 Willow St	2.70	In Transit	TRK10009	2024-03-25	5	9
10	Charlotte Evans	852 Willow St	John Doe	123 Main St	5.00	Pending	TRK10010	2024-03-24	5	10

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

To Retrieve a list of all couriers and their corresponding services including cases where there are no matches on either side OUTER JOIN is used

MySQL **does not support FULL OUTER JOIN** directly. Instead,we can achieve the same result using a **UNION** of **LEFT JOIN** and **RIGHT JOIN**.

SELECT C.CourierID, C.SenderName, C.ReceiverName, C.Weight, C.Status,

C.TrackingNumber, C.DeliveryDate, CS.ServiceID, CS.ServiceName, CS.Cost

FROM Courier C

LEFT JOIN CourierServices CS ON C.ServiceID = CS.ServiceID

UNION

SELECT C.CourierID, C.SenderName, C.ReceiverName, C.Weight, C.Status,

C.TrackingNumber, C.DeliveryDate, CS.ServiceID, CS.ServiceName, CS.Cost

FROM Courier C

RIGHT JOIN CourierServices CS ON C.ServiceID = CS.ServiceID;

Explanation:

1. LEFT JOIN ensures that all couriers are retrieved, even if they don't have a matching service.

2. RIGHT JOIN ensures that all services are retrieved, even if they are not linked to any courier.
3. UNION combines both results to simulate a FULL OUTER JOIN.

Output :

CourierID	SenderName	ReceiverName	Weight	Status	TrackingNumber	DeliveryDate	ServiceID	ServiceName	Cost
1	John Doe	Alice Smith	2.50	In Transit	TRK10001	2024-03-20	11	Express Plus	600.00
2	Alice Smith	Michael Brown	1.20	Delivered	TRK10002	2024-03-18	16	Local Express	200.00
3	Michael Brown	Emma Wilson	3.00	Pending	TRK10003	2024-03-22	11	Express Plus	600.00
4	Emma Wilson	Daniel Lee	2.80	In Transit	TRK10004	2024-03-21	11	Express Plus	600.00
5	Daniel Lee	Sophia Green	4.10	Shipped	TRK10005	2024-03-19	9	Fragile Item Handling	350.00
6	Sophia Green	William Clark	1.90	Delivered	TRK10006	2024-03-17	11	Express Plus	600.00
7	William Clark	Olivia Adams	2.20	Pending	TRK10007	2024-03-23	11	Express Plus	600.00
8	Olivia Adams	James Turner	3.50	Shipped	TRK10008	2024-03-20	9	Fragile Item Handling	350.00
9	James Turner	Charlotte Evans	2.70	In Transit	TRK10009	2024-03-25	11	Express Plus	600.00
10	Charlotte Evans	John Doe	5.00	Pending	TRK10010	2024-03-24	9	Fragile Item Handling	350.00
11	Ethan Carter	Isabella Lewis	3.20	In Transit	TRK10011	2024-03-26	9	Fragile Item Handling	350.00
12	Isabella Lewis	Liam Parker	2.50	Delivered	TRK10012	2024-03-24	11	Express Plus	600.00

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

To retrieve a list of all employees and their corresponding payments, including cases where there are no matches on either side, we need a FULL OUTER JOIN. Since MySQL does not support FULL OUTER JOIN, we can achieve the same result using a UNION of LEFT JOIN and RIGHT JOIN.

```
SELECT E.EmployeeID, E.Name AS EmployeeName, P.PaymentID, P.Amount, P.PaymentDate

FROM Employees E

LEFT JOIN Courier C ON E.EmployeeID = C.EmployeeID

LEFT JOIN Payments P ON C.CourierID = P.CourierID

UNION

SELECT E.EmployeeID, E.Name AS EmployeeName, P.PaymentID, P.Amount, P.PaymentDate

FROM Employees E

RIGHT JOIN Courier C ON E.EmployeeID = C.EmployeeID

RIGHT JOIN Payments P ON C.CourierID = P.CourierID;
```

Explanation:

1. **LEFT JOIN** retrieves all employees, including those who have **not** received any payments.
2. **RIGHT JOIN** retrieves all payments, ensuring that payments without a corresponding employee are also included.
3. **UNION** combines both results, effectively simulating a **FULL OUTER JOIN**.

Output :

EmployeeID	EmployeeName	PaymentID	Amount	PaymentDate
1	David Wilson	1	100.00	2024-03-21
1	David Wilson	21	120.50	2025-03-17
1	David Wilson	22	98.75	2025-03-18
1	David Wilson	2	250.00	2024-03-19
1	David Wilson	23	85.30	2025-03-19
1	David Wilson	24	102.25	2025-03-20
1	David Wilson	26	75.00	2025-03-17
1	David Wilson	NULL	NULL	NULL
2	Emma Brown	3	500.00	2024-03-22
2	Emma Brown	25	110.00	2025-03-21
2	Emma Brown	27	99.99	2025-03-18
2	Emma Brown	4	400.00	2024-03-20
2	Emma Brown	28	120.50	2025-03-19

EmployeeID	EmployeeName	PaymentID	Amount	PaymentDate
6	Olivia Davis	11	110.00	2024-03-27
6	Olivia Davis	12	260.00	2024-03-25
7	Ethan Carter	13	150.00	2024-03-29
7	Ethan Carter	14	500.00	2024-03-28
7	Ethan Carter	32	3200.00	2025-03-19
8	Isabella Lewis	15	700.00	2024-03-26
8	Isabella Lewis	16	300.00	2024-03-24
8	Isabella Lewis	NULL	NULL	NULL
9	Liam Parker	17	900.00	2024-03-30
9	Liam Parker	18	550.00	2024-03-27
9	Liam Parker	33	4000.00	2025-03-20
10	Ava Martinez	19	1500.00	2024-03-31
10	Ava Martinez	20	1200.00	2024-04-01

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

To achieve this, we need to list every user with every courier service, which can be done using a CROSS JOIN.

```
SELECT U.UserID, U.Name AS UserName, CS.ServiceID, CS.ServiceName, CS.Cost

FROM Users U

CROSS JOIN CourierServices CS;
```

Explanation:

1. **CROSS JOIN** creates all possible combinations between users and courier services.
2. Each **user** will be paired with **every courier service**.
3. The result will have **(Total Users × Total Courier Services) rows**.

Output :

UserID	UserName	ServiceID	ServiceName	Cost
20	Noah White	1	Standard Delivery	100.00
19	Emma Brown	1	Standard Delivery	100.00
18	David Wilson	1	Standard Delivery	100.00
17	Olivia Davis	1	Standard Delivery	100.00
16	Sophia Johnson	1	Standard Delivery	100.00
15	Mason Scott	1	Standard Delivery	100.00
14	Ava Martinez	1	Standard Delivery	100.00
13	Liam Parker	1	Standard Delivery	100.00
12	Isabella Lewis	1	Standard Delivery	100.00

UserID	UserName	ServiceID	ServiceName	Cost
15	Mason Scott	2	Express Delivery	250.00
14	Ava Martinez	2	Express Delivery	250.00
13	Liam Parker	2	Express Delivery	250.00
12	Isabella Lewis	2	Express Delivery	250.00
11	Ethan Carter	2	Express Delivery	250.00
10	Charlotte Evans	2	Express Delivery	250.00
9	James Turner	2	Express Delivery	250.00
8	Olivia Adams	2	Express Delivery	250.00

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

To achieve this, we need to **list every employee with every location**, which can be done using a **CROSS JOIN**.

```
SELECT E.EmployeeID, E.Name AS EmployeeName, L.LocationID, L.LocationName

FROM Employees E

CROSS JOIN Locations L;
```

Explanation:

1. CROSS JOIN generates all possible combinations between employees and locations.
2. Each employee will be paired with every location.
3. The result will have (Total Employees × Total Locations) rows.

Output :

EmployeeID	EmployeeName	Email	contactNumber	LocationID	LocationName
20	Grace Hall	grace@example.com	1980000010	1	Warehouse A
19	Jack Turner	jack@example.com	1090000009	1	Warehouse A
18	Aria Ross	aria@example.com	2100000008	1	Warehouse A
17	Owen King	owen@example.com	3210000007	1	Warehouse A
16	Lily Baker	lily@example.com	4321000006	1	Warehouse A
15	Henry Green	henry@example.com	5432000005	1	Warehouse A
14	Evelyn Scott	evelyn@example.com	6543000004	1	Warehouse A
13	Daniel Evans	daniel@example.com	7654000003	1	Warehouse A
12	Harper Reed	harper@example.com	8765000002	1	Warehouse A
11	Lucas Foster	lucas@example.com	9876000001	1	Warehouse A
10	Ava Martinez	ava@example.com	1987654321	1	Warehouse A
9	Liam Parker	liam@example.com	1098765432	1	Warehouse A

EmployeeID	EmployeeName	Email	contactNumber	LocationID	LocationName
18	Aria Ross	aria@example.com	2100000008	3	Hub Center
17	Owen King	owen@example.com	3210000007	3	Hub Center
16	Lily Baker	lily@example.com	4321000006	3	Hub Center
15	Henry Green	henry@example.com	5432000005	3	Hub Center
14	Evelyn Scott	evelyn@example.com	6543000004	3	Hub Center
13	Daniel Evans	daniel@example.com	7654000003	3	Hub Center
12	Harper Reed	harper@example.com	8765000002	3	Hub Center
11	Lucas Foster	lucas@example.com	9876000001	3	Hub Center
10	Ava Martinez	ava@example.com	1987654321	3	Hub Center
9	Liam Parker	liam@example.com	1098765432	3	Hub Center
8	Isabella Lewis	isabella@example.com	2109876543	3	Hub Center

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

Since couriers already have sender details in the table (SenderName and SenderAddress), we can retrieve the courier details along with the sender's information.

SELECT C.CourierID, C.TrackingNumber, C.Status, C.SenderName, C.SenderAddress

FROM Courier C;

Explanation:

1. This query directly retrieves the courier details along with sender information.
2. Since sender details (SenderName, SenderAddress) are already part of the Courier table, we do not need a JOIN.
3. Also ,

SELECT C.CourierID, C.TrackingNumber, C.Status, U.UserID, U.Name AS SenderName, U.Address AS SenderAddress

FROM Courier C

LEFT JOIN Users U ON C.SenderName = U.Name;

Output :

CourierID	TrackingNumber	Status	UserID	SenderName	SenderAddress
1	TRK10001	In Transit	1	John Doe	123 Main St
2	TRK10002	Delivered	2	Alice Smith	456 Elm St
3	TRK10003	Pending	3	Michael Brown	789 Oak St
4	TRK10004	In Transit	4	Emma Wilson	321 Pine St
5	TRK10005	Shipped	5	Daniel Lee	654 Cedar St
6	TRK10006	Delivered	6	Sophia Green	987 Maple St
7	TRK10007	Pending	7	William Clark	258 Birch St
8	TRK10008	Shipped	8	Olivia Adams	369 Walnut St
9	TRK10009	In Transit	9	James Turner	741 Cherry St
10	TRK10010	Pending	10	Charlotte Evans	852 Willow St
11	TRK10011	In Transit	11	Ethan Carter	741 Hill St
12	TRK10012	Delivered	12	Isabella Lewis	852 Valley St

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

Since the **Courier** table already contains receiver details (ReceiverName and ReceiverAddress), we can directly retrieve the required information.

```
SELECT C.CourierID, C.TrackingNumber, C.Status, C.ReceiverName, C.ReceiverAddress

FROM Courier C;
```

Explanation:

1. This query fetches CourierID, TrackingNumber, Status, and Receiver information (ReceiverName and ReceiverAddress).
2. Since receiver details are stored within the Courier table, a JOIN is not necessary.
3. If there were a separate Users table containing receiver details, we would use a LEFT JOIN like this:

```
SELECT C.CourierID, C.TrackingNumber, C.Status, U.UserID, U.Name AS ReceiverName, U.Address AS ReceiverAddress

FROM Courier C

LEFT JOIN Users U ON C.ReceiverName = U.Name;
```

Output :

CourierID	TrackingNumber	Status	DeliveryDate	UserID	ReceiverName	ReceiverAddress
1	TRK10001	In Transit	2024-03-20	2	Alice Smith	456 Elm St
2	TRK10002	Delivered	2024-03-18	3	Michael Brown	789 Oak St
3	TRK10003	Pending	2024-03-22	4	Emma Wilson	321 Pine St
4	TRK10004	In Transit	2024-03-21	5	Daniel Lee	654 Cedar St
5	TRK10005	Shipped	2024-03-19	6	Sophia Green	987 Maple St
6	TRK10006	Delivered	2024-03-17	7	William Clark	258 Birch St
7	TRK10007	Pending	2024-03-23	8	Olivia Adams	369 Walnut St
8	TRK10008	Shipped	2024-03-20	9	James Turner	741 Cherry St
9	TRK10009	In Transit	2024-03-25	10	Charlotte Evans	852 Willow St
10	TRK10010	Pending	2024-03-24	1	John Doe	123 Main St
11	TRK10011	In Transit	2024-03-26	12	Isabella Lewis	852 Valley St
12	TRK10012	Delivered	2024-03-24	13	Liam Parker	963 River St

---

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

To Retrieve a list of couriers along with the courier service details , we can use a LEFT JOIN to fetch courier details along with their corresponding service details.

```
SELECT C.CourierID, C.TrackingNumber, C.Status, C.SenderName, C.ReceiverName, C.Weight, CS.ServiceID, CS.ServiceName, CS.Cost

FROM Courier C

LEFT JOIN CourierServices CS ON C.ServiceID = CS.ServiceID;
```

Explanation:

1. **Retrieves courier details** such as **CourierID, TrackingNumber, Status, SenderName, ReceiverName, and Weight**.
2. **Joins** the **Courier** table with the **CourierServices** table using **ServiceID**.
3. **Uses LEFT JOIN** to ensure all couriers are included, even if some don't have an associated service.

Output :

CourierID	TrackingNumber	Status	SenderName	ReceiverName	Weight	ServiceID	ServiceName	Cost
1	TRK10001	In Transit	John Doe	Alice Smith	2.50	11	Express Plus	600.00
2	TRK10002	Delivered	Alice Smith	Michael Brown	1.20	16	Local Express	200.00
3	TRK10003	Pending	Michael Brown	Emma Wilson	3.00	11	Express Plus	600.00
4	TRK10004	In Transit	Emma Wilson	Daniel Lee	2.80	11	Express Plus	600.00
5	TRK10005	Shipped	Daniel Lee	Sophia Green	4.10	9	Fragile Item Handling	350.00
6	TRK10006	Delivered	Sophia Green	William Clark	1.90	11	Express Plus	600.00
7	TRK10007	Pending	William Clark	Olivia Adams	2.20	11	Express Plus	600.00
8	TRK10008	Shipped	Olivia Adams	James Turner	3.50	9	Fragile Item Handling	350.00
9	TRK10009	In Transit	James Turner	Charlotte Evans	2.70	11	Express Plus	600.00
10	TRK10010	Pending	Charlotte Evans	John Doe	5.00	9	Fragile Item Handling	350.00

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

Since each **Courier** record has an **EmployeeID**, we can count the number of couriers handled by each employee using **GROUP BY**.

```
SELECT E.EmployeeID, E.Name, COUNT(C.CourierID) AS TotalCouriersAssigned
FROM Employee E
LEFT JOIN Courier C ON E.EmployeeID = C.EmployeeID
GROUP BY E.EmployeeID, E.EmployeeName
ORDER BY TotalCouriersAssigned DESC;
```

Explanation:

1. **Selects EmployeeID and EmployeeName** from the **Employee** table.
2. **Counts the number of couriers** assigned to each employee using COUNT(C.CourierID).
3. **Uses LEFT JOIN** to ensure employees without assigned couriers are included (if any).
4. **Groups by EmployeeID and EmployeeName** to aggregate the counts.

Output :

EmployeeID	Name	TotalCouriersAssigned
5	Mason Scott	5
12	Harper Reed	4
1	David Wilson	3
2	Emma Brown	3
3	Noah White	3
4	Sophia Johnson	3
7	Ethan Carter	3
8	Isabella Lewis	3
9	Liam Parker	3
17	Owen King	3
6	Olivia Davis	2
10	Ava Martinez	2
11	Lucas Foster	2
16	Lily Baker	2
18	Aria Ross	2
19	Jack Turner	2
20	Grace Hall	2
13	Daniel Evans	1
14	Evelyn Scott	1
15	Henry Green	1

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

Since the Payment table has a LocationID and an Amount, we can sum the total payments per location using GROUP BY.

```
SELECT L.LocationID, L.LocationName, SUM(P.Amount) AS TotalPaymentReceived
FROM Location L
LEFT JOIN Payment P ON L.LocationID = P.LocationID
GROUP BY L.LocationID, L.LocationName
ORDER BY TotalPaymentReceived DESC;
```

Explanation:

1. **Selects LocationID and LocationName** from the **Location** table.
2. **Sums the Amount column** from the **Payment** table to calculate total payments per location.
3. **Uses LEFT JOIN** to ensure all locations are included, even if they have no payments.
4. **Groups by LocationID and LocationName** to aggregate the payments.

Output :

LocationID	LocationName	TotalPaymentReceived
5	Sorting Facility 1	5730.50
10	Drop-off Point B	5700.00
8	Retail Store 2	5650.00
4	Main Office	5202.24
9	Drop-off Point A	5050.00
7	Retail Store 1	1050.00
3	Hub Center	810.30
2	Warehouse B	759.05
6	Sorting Facility 2	600.00
1	Warehouse A	416.25

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

To find all couriers sent by each sender, we will **group them based on SenderName**

SELECT SenderName, CourierID, TrackingNumber, ReceiverName, ReceiverAddress, Status, DeliveryDate

FROM Courier

ORDER BY SenderName, DeliveryDate DESC;

Explanation :

- First by SenderName to group couriers from the same sender.
- Then by DeliveryDate (DESC) so the most recent courier appears first.

Output :

SenderName	CourierID	TrackingNumber	ReceiverName	ReceiverAddress	Status	DeliveryDate
Alice	61	TRK0031	Bob	Los Angeles	Delivered	2025-03-15
Alice Johnson	22	TRK1002	Charlie Brown	789 Road, TX	Pending	2025-03-19
Alice Johnson	21	TRK1001	Bob Smith	456 Lane, CA	In Transit	2025-03-18
Alice Smith	52	TRK5002	Bob Johnson	321 Blvd, CA	Delivered	2025-03-17
Alice Smith	51	TRK5001	Bob Johnson	321 Blvd, CA	Delivered	2025-03-17
Alice Smith	2	TRK10002	Michael Brown	789 Oak St	Delivered	2024-03-18
Ava Martinez	14	TRK10014	Mason Scott	357 Bay St	In Transit	2024-03-27
Brad Pitt	43	TRK4003	Angelina Jolie	333 Sunset Blvd, LA	Scheduled	2025-03-17
Charlie	62	TRK0032	David	Chicago	Delivered	2025-03-16
Charlotte Evans	10	TRK10010	John Doe	123 Main St	Pending	2024-03-24

44. List all employees who share the same role.

To retrieve all employees who share the same role, your Employees table must have a column for Role. Assuming the table structure includes a Role column, the query would be:

SELECT EmployeeID, Name, Role

FROM Employees

WHERE Role IN (SELECT Role FROM Employees GROUP BY Role HAVING COUNT(*) > 1)

ORDER BY Role, Name;

Explanation:

- Finds employees with the same role by checking if their Role appears more than once in the table.
- Orders the result by Role and Name.

Output :

EmployeeID	Name	Role
10	Ava Martinez	Customer Support
20	Grace Hall	Customer Support
1	David Wilson	Delivery Agent
14	Evelyn Scott	Delivery Agent
11	Lucas Foster	Delivery Agent
4	Sophia Johnson	Delivery Agent
19	Jack Turner	Field Staff
9	Liam Parker	Field Staff
18	Aria Ross	Finance Analyst
8	Isabella Lewis	Finance Analyst
16	Lily Baker	HR Executive
6	Olivia Davis	HR Executive
7	Ethan Carter	IT Specialist

---

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

The SQL query to retrieve all payments made for couriers sent from the same location:

```
SELECT P.PaymentID, P.CourierID, P.Amount, P.PaymentDate, C.SenderAddress
FROM Payment P
JOIN Courier C ON P.CourierID = C.CourierID
WHERE C.SenderAddress IN (
   SELECT SenderAddress
   FROM Courier
   GROUP BY SenderAddress
   HAVING COUNT(*) > 1
)
ORDER BY C.SenderAddress, P.PaymentDate;
```

Explanation:

- Joins the Payment and Courier tables on CourierID to get sender address details.
- Filters for sender locations that have sent more than one courier.
- Orders the result by SenderAddress and PaymentDate

Output :

Payment	Courier	Amount	PaymentDate	SenderAddress
1	1	100.00	2024-03-21	123 Main St
21	1	120.50	2025-03-17	123 Main St
22	1	98.75	2025-03-18	123 Main St

---

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

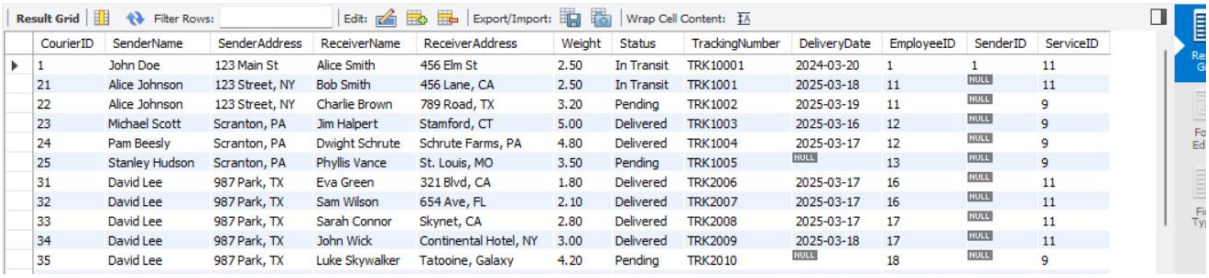
To retrieve all couriers sent from the same location (based on SenderAddress), you can use the following SQL query:

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

Explanation:

1. The subquery identifies SenderAddress values that appear more than once (i.e., locations from which multiple couriers have been sent).
2. The main query retrieves all couriers where the SenderAddress matches one of those identified locations.

Output :



CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	EmployeeID	SenderID	ServiceID
1	John Doe	123 Main St	Alice Smith	456 Elm St	2.50	In Transit	TRK10001	2024-03-20	1	1	11
21	Alice Johnson	123 Street, NY	Bob Smith	456 Lane, CA	2.50	In Transit	TRK1001	2025-03-18	11	NULL	11
22	Alice Johnson	123 Street, NY	Charlie Brown	789 Road, TX	3.20	Pending	TRK1002	2025-03-19	11	NULL	9
23	Michael Scott	Scranton, PA	Jim Halpert	Stamford, CT	5.00	Delivered	TRK1003	2025-03-16	12	NULL	9
24	Pam Beesly	Scranton, PA	Dwight Schrute	Schrute Farms, PA	4.80	Delivered	TRK1004	2025-03-17	12	NULL	9
25	Stanley Hudson	Scranton, PA	Phyllis Vance	St. Louis, MO	3.50	Pending	TRK1005	NULL	13	NULL	9
31	David Lee	987 Park, TX	Eva Green	321 Blvd, CA	1.80	Delivered	TRK2006	2025-03-17	16	NULL	11
32	David Lee	987 Park, TX	Sam Wilson	654 Ave, FL	2.10	Delivered	TRK2007	2025-03-17	16	NULL	11
33	David Lee	987 Park, TX	Sarah Connor	Skynet, CA	2.80	Delivered	TRK2008	2025-03-17	17	NULL	11
34	David Lee	987 Park, TX	John Wick	Continental Hotel, NY	3.00	Delivered	TRK2009	2025-03-18	17	NULL	11
35	David Lee	987 Park, TX	Luke Skywalker	Tatooine, Galaxy	4.20	Pending	TRK2010	NULL	18	NULL	9

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

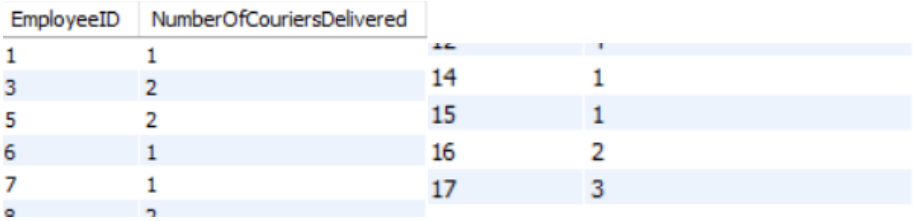
To list employees and the number of couriers they have delivered, use the following SQL query:

```
SELECT EmployeeID, COUNT(*) AS NumberOfCouriersDelivered
FROM Courier WHERE Status = 'Delivered'
GROUP BY EmployeeID;
```

Explanation:

- The query counts the number of couriers that have a status of **'Delivered'**.
- It groups the results by EmployeeID, so each employee's total delivered couriers are displayed.

Output :



EmployeeID	NumberOfCouriersDelivered
1	1
3	2
5	2
6	1
7	1
14	1
15	1
16	2
17	3

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

To find couriers where the payment amount is greater than the cost of the corresponding courier service, use the following SQL query:

SELECT c.CourierID, c.TrackingNumber, p.Amount AS PaymentAmount, cs.Cost AS ServiceCost

FROM Payment p JOIN Courier c ON p.CourierID = c.CourierID

JOIN CourierServices cs ON c.ServiceID = cs.ServiceID

WHERE p.Amount > cs.Cost;

Explanation:

Joins the Payment table with the Courier table on CourierID.Joins the CourierServices table using ServiceID to get the service cost.

Output :

CourierID	TrackingNumber	PaymentAmount	ServiceCost
2	TRK10002	250.00	200.00
5	TRK10005	800.00	350.00
8	TRK10008	600.00	350.00
10	TRK10010	1000.00	350.00
15	TRK10015	700.00	350.00
17	TRK10017	900.00	600.00
18	TRK10018	550.00	350.00
19	TRK10019	1500.00	600.00
20	TRK10020	1200.00	350.00
61	TRK0031	3500.00	350.00

Conclusion :

Task 4 involved retrieving various details related to payments, couriers, employees, and locations through SQL queries. Each query was designed to extract specific information, such as payments linked to couriers, employee assignments, and service details. All queries requested in Task 4 have been successfully implemented.
