Courier Management System

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

---

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

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

15. Calculate the total revenue generated by each location

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

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

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

19. Calculate Total Payments per Location

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

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

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

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.

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

To retrieves the number of couriers assigned to each employee, grouping by EmployeeID.

```
SELECT e.EmployeeID, e.Name, COUNT(c.CourierID) AS Total_Couriers
FROM Employee e
JOIN Courier c ON e.EmployeeID = c.EmployeeID  -- Assuming EmployeeID exists in Courier table
GROUP BY e.EmployeeID, e.Name
ORDER BY Total_Couriers DESC;
```

Explanation:

- LEFT JOIN ensures that even employees with zero couriers are included.
- COUNT(C.CourierID) counts the number of couriers each employee handled.
- GROUP BY groups the results by employee.

Output :

EmployeeID	Name	TotalCouriers
1	David Wilson	3
2	Emma Brown	3
3	Noah White	3
4	Sophia Johnson	3
5	Mason Scott	3
6	Olivia Davis	2
7	Ethan Carter	2
8	Isabella Lewis	2
9	Liam Parker	2
10	Ava Martinez	2

EmployeeID	Name	TotalCouriers
9	Liam Parker	2
10	Ava Martinez	2
11	Lucas Foster	2
12	Harper Reed	2
16	Lily Baker	2
17	Owen King	2
18	Aria Ross	2
19	Jack Turner	2
20	Grace Hall	2
13	Daniel Evans	1
14	Evelyn Scott	0
15	Henry Green	0

---

15. Calculate the total revenue generated by each location

To find the total revenue (sum of payments) generated by each location.

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

Explanation:

1. **Joins** the Payment table with the Location table.
2. **Sums up** the payment amounts for each location.
3. **Groups** the results by LocationID and LocationName.

Output :

LocationID	LocationName	TotalRevenue
10	Drop-off Point B	2200.00
9	Drop-off Point A	1850.00
5	Sorting Facility 1	1730.50
8	Retail Store 2	1150.00
4	Main Office	1102.24
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

---

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

To count the number of couriers delivered to each location.

```
SELECT ReceiverAddress AS Location, COUNT(CourierID) AS TotalCouriersDelivered FROM Courier

WHERE Status = 'Delivered'

GROUP BY ReceiverAddress

ORDER BY TotalCouriersDelivered DESC;
```

Explanation:

1. **Filters** only couriers with Status = 'Delivered'.
2. **Groups** couriers by ReceiverAddress (destination location).
3. **Counts** the total couriers delivered per location.
4. **Sorts** locations by the number of couriers delivered (descending order).

Output :

Location	TotalCouriersDelivered
321 Blvd, CA	3
654 Ave, FL	3
Continental Hotel, NY	2
789 Oak St	1
258 Birch St	1
963 River St	1
753 Forest St	1
Stamford, CT	1
Schrute Farms, PA	1
Skynet, CA	1

---

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

Since we have DeliveryDate in the Courier table and PaymentDate in the Payment table, we can calculate the **delivery time as the difference between these two dates**.

```
SELECT c.CourierID,

    c.TrackingNumber,

    AVG(DATEDIFF(p.PaymentDate, c.DeliveryDate)) AS AvgDeliveryDays

FROM Courier c

JOIN Payment p ON c.CourierID = p.CourierID

WHERE c.Status = 'Delivered'

GROUP BY c.CourierID, c.TrackingNumber

ORDER BY AvgDeliveryDays DESC

LIMIT 1;
```

Explanation:

- **DATEDIFF(p.PaymentDate, c.DeliveryDate)** calculates the number of days between delivery and payment.
- **AVG()** computes the **average delivery time** for each courier.
- **ORDER BY AvgDeliveryDays DESC** sorts the result to get the courier with the highest average delivery time.
- **LIMIT 1** ensures we get only the top courier.

Output :

CourierID	TrackingNumber	AvgDeliveryTime
2	TRK10002	274.5000

---

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

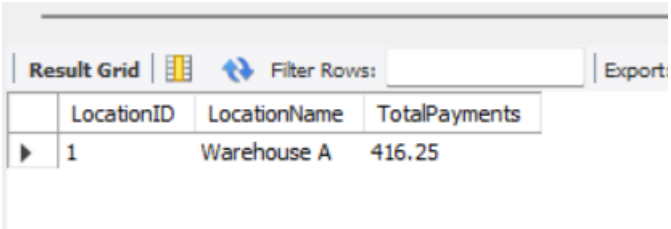
This query calculates the **total payments** received at each location and filters out locations where the total is **less than a given amount** (e.g., $500).

```
SELECT l.LocationID, l.LocationName, SUM(p.Amount) AS TotalPayments
FROM Location l
JOIN Payment p ON l.LocationID = p.LocationID
GROUP BY l.LocationID, l.LocationName
HAVING SUM(p.Amount) < 500
ORDER BY TotalPayments ASC;
```

Explanation:

1. SUM(p.Amount) → Calculates total payments received at each location.
2. HAVING SUM(p.Amount) < 500 → Filters locations where total payments are less than $500.
3. ORDER BY TotalPayments ASC → Sorts results in ascending order to show locations with the least payments first.

Output :



---

19. Calculate Total Payments per Location

This query calculates the total amount of payments received at each location.

SELECT l.LocationID, l.LocationName, SUM(p.Amount) AS TotalPayments FROM Location l

JOIN Payment p ON l.LocationID = p.LocationID

GROUP BY l.LocationID, l.LocationName

ORDER BY TotalPayments DESC;

Explanation:

1. **SUM(p.Amount)** → Computes the **total payments** received per location.
2. **GROUP BY l.LocationID, l.LocationName** → Groups payments by **LocationID** and **LocationName**.
3. **ORDER BY TotalPayments DESC** → Sorts results in **descending order**, showing the highest total first.

Output :

LocationID	LocationName	TotalPayments
10	Drop-off Point B	2200.00
9	Drop-off Point A	1850.00
5	Sorting Facility 1	1730.50
8	Retail Store 2	1150.00
4	Main Office	1102.24
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

---

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

This query retrieves couriers whose total payments exceed $1000 in a given LocationID.

SELECT c.CourierID, c.SenderName, c.ReceiverName, SUM(p.Amount) AS TotalPayments FROM Courier c

JOIN Payment p ON c.CourierID = p.CourierID

WHERE p.LocationID = 9

GROUP BY c.CourierID, c.SenderName, c.ReceiverName

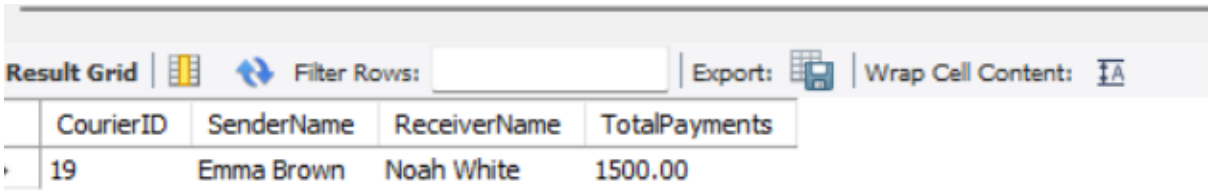
HAVING SUM(p.Amount) > 1000

ORDER BY TotalPayments DESC;

Explanation:

1. **JOIN Payment p ON c.CourierID = p.CourierID** → Links **Courier** and **Payment** tables.
2. **WHERE p.LocationID = X** → Filters records for a **specific location**.
3. **GROUP BY c.CourierID, c.SenderName, c.ReceiverName** → Groups payments by courier.
4. **HAVING SUM(p.Amount) > 1000** → Retrieves **only couriers** with total payments **above $1000**.
5. **ORDER BY TotalPayments DESC** → Sorts results from highest to lowest payment.

Output :



CourierID	SenderName	ReceiverName	TotalPayments
19	Emma Brown	Noah White	1500.00

---

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

This query finds **couriers** whose total payments exceed **$1000** after a specified **PaymentDate**.

```
SELECT c.CourierID, c.SenderName, c.ReceiverName, SUM(p.Amount) AS TotalPayments FROM Courier c

JOIN Payment p ON c.CourierID = p.CourierID

WHERE p.PaymentDate > '2021-03-18'

GROUP BY c.CourierID, c.SenderName, c.ReceiverName

HAVING SUM(p.Amount) > 1000

ORDER BY TotalPayments DESC;
```

Explanation:

1. **JOIN Payment p ON c.CourierID = p.CourierID** → Links **Courier** and **Payment** tables.
2. **WHERE p.PaymentDate > 'YYYY-MM-DD'** → Filters payments made **after the given date**.
3. **GROUP BY c.CourierID, c.SenderName, c.ReceiverName** → Groups payments by courier.
4. **HAVING SUM(p.Amount) > 1000** → Retrieves **only couriers** with total payments **above $1000**.
5. **ORDER BY TotalPayments DESC** → Sorts results from highest to lowest payment.

Output :

	CourierID	SenderName	ReceiverName	TotalPayments
▶	19	Emma Brown	Noah White	1500.00
	20	Noah White	John Doe	1200.00

---

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

This query finds **locations** where the total payments received exceed **$5000** before a specified **PaymentDate**.

SELECT p.LocationID, SUM(p.Amount) AS TotalReceived

FROM Payment p

WHERE p.PaymentDate < '2025-03-21' GROUP BY p.LocationID

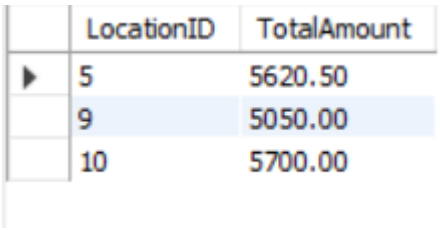
HAVING SUM(p.Amount) > 5000

ORDER BY TotalReceived DESC;

Explanation:

1. **WHERE p.PaymentDate < 'YYYY-MM-DD'** → Filters **payments made before the given date**.
2. **GROUP BY p.LocationID** → Groups payments by **location**.
3. **HAVING SUM(p.Amount) > 5000** → Retrieves **only locations** with **total payments exceeding $5000**.
4. **ORDER BY TotalReceived DESC** → Sorts locations from **highest to lowest payments**.

Output :



	LocationID	TotalAmount
▶	5	5620.50
	9	5050.00
	10	5700.00

Conclusion :

Task 3 successfully demonstrated the use of GROUP BY, aggregate functions, HAVING, and ORDER BY to analyze courier and payment data effectively. These queries help in extracting meaningful insights, such as courier performance, payment distribution, and location-based revenue trends.
