Courier Shipping Management System

- **Basic queries with conditions and joins:**

1) Get all couriers for a particular customer within a given date range.

Relational Query:

$\sigma_{(Customer\_ID = '23110100002' \text{ AND } Date >= '08\text{-}11\text{-}2023' \text{ AND } Date <= '08\text{-}12\text{-}2023')}(Courier)$

SQL Query & Output:

```
-- Get all couriers for a particular customer within a given date range.
SELECT * FROM Courier
WHERE Customer_ID = '23110100002'
  AND Date BETWEEN '08-11-2023' AND '08-12-2023';
```

Data Output | Messages | Notifications

	reference_id [PK] character	date date	distance numeric (4)	weight numeric (4)	expected_delivery_date date	status character varying (9)	type character varying (16)	customer_id numeric (11)	vehicle_id character	branch_id numeric (10)	to_branch numeric (10)	price numeric (5)	profit numeric (5)
1	23110100ab	2023-11-10	280	5	2023-11-11	Collected	Express	23110100002	GJ06KL0123	2310110930	2310210930	910	182

2) Get the city name travelled by a particular vehicleID. (Ex. GJ06KL0123):

Relational Query:

$\pi_{city}(\sigma_{vehicle\_id = "GJ06KL0123"}$ (vehiclecitymapping))

SQL Query & Output:

```
-- Get the city name travelled by a particular vehicleID.
SELECT City FROM VehicleCityMapping
WHERE Vehicle_ID = 'GJ06KL0123';
```

Data Output | Messages | Notifications

	city character varying (50)
1	Valsad
2	Bharuch

3) Get the lost courier within a given date range.

Relational Query:

$\sigma_{(status = "Lost" \text{ AND } "2024\text{-}01\text{-}01" <= date \text{ AND } date <= "2024\text{-}01\text{-}10")}(courier)$

SQL Query & Output:

```
-- Get the lost courier within a given date range.
SELECT * FROM Courier
WHERE Status = 'Lost'
AND Date BETWEEN '2024-01-01' AND '2024-01-10';
```

Data Output | Messages | Notifications

	reference_id [PK] character	date date	distance numeric (4)	weight numeric (4)	expected_delivery_date date	status character varying (9)	type character varying (16)	customer_id numeric (11)	vehicle_id character	branch_id numeric (10)	to_branch numeric (10)	price numeric (5)	profit numeric (5)
1	24010100ad	2024-01-06	700	7	2024-01-10	Lost	Standard	23110100010	TN07IJ6789	2310020930	2310130930	1960	392

4) Get the top 5 most rated drivers.

Relational Query:

$\pi_{d.id,\ e.name,\ d.rating}(\rho(d,driver) \bowtie_{(d.id = e.employee\_id)} \rho(e,employee))$

SQL Query & Output:

```
27  SELECT D.ID, E.Name, D.Rating
28  FROM Driver D
29  JOIN Employee E ON D.ID = E.Employee_id
30  ORDER BY D.Rating DESC
31  LIMIT 5;
```

Data Output | Messages | Notifications

	id numeric (10)	name character varying (50)	rating numeric (2,1)
1	2310030002	Suresh Reddy	4.9
2	2310010002	Rajesh Kumar	4.8
3	2310020002	Anil Mehta	4.7
4	2310140002	Amit Yadav	4.5
5	2310110002	Mohit Verma	4.2

5) Find out the top 5 employees that have been with the company the longest.

Relational Query:

$\pi_{employee\_id,\ name,\ date\_of\_joining}$ (employee)

SQL Query & Output:

```
43  SELECT Employee_id, Name, Date_of_Joining
44  FROM Employee
45  ORDER BY Date_of_Joining ASC
46  LIMIT 5;
```

Data Output | Messages | Notifications

	employee_id [PK] numeric (10)	name character varying (50)	date_of_joining date
1	2310010001	Rahul Sharma	2023-10-01
2	2310020001	Priya Mehta	2023-10-02
3	2310030001	Amit Gupta	2023-10-03
4	2310040001	Sneha Reddy	2023-10-04
5	2310050001	Vikram Singh	2023-10-05

6) Get Active Couriers of a particular customer

Relational Query:

$\pi_{c.reference\_id, c.date, c.status, c.type, p.amount}$

($\sigma_{cu.customer\_id = 23110100003 \text{ AND } (c.status = \text{"Collected"} \text{ OR } c.status = \text{"Shipped"} \text{ OR } c.status = \text{"Arrived"})}$

(ρ(Cu,customer) $\bowtie_{cu.customer\_id = c.customer\_id}$ ρ(C,courier) $\bowtie_{c.reference\_id = p.reference\_id}$ ρ(P,payment)))

SQL Query & Output:

```
135  SELECT C.Reference_ID, C.Date, C.Status, C.Type, P.Amount
136  FROM Customer CU
137  INNER JOIN Courier C ON CU.Customer_ID = C.Customer_ID
138  INNER JOIN Payment P ON C.Reference_ID = P.Reference_ID
139  WHERE CU.Customer_ID = 23110100003
140  AND C.Status IN ('Collected','Shipped', 'Arrived')
141  ORDER BY C.Date DESC;
```

Data Output | Messages | Notifications

	reference_id character	date date	status character varying (9)	type character varying (16)	amount numeric (5)
1	23110100ad	2023-11-05	Shipped	Standard	3500

- **Queries with aggregation in SELECT and aggregated conditions in HAVING clause:**

7) Get the total weight of couriers sent from a particular city.

Relational Query:

$\pi_{SUM\ (c.weight) \rightarrow total\_weight}\ (\mathcal{F}_{SUM\ (c.weight)}\ (\sigma_{b\ .\ city\ =\ "Valsad"}\ (\rho(c,courier) \bowtie_{c\ .\ branch\_id\ =\ b\ .\ branch\_id}\ \rho(b,branch))))$

SQL Query & Output:

```sql
-- Get the total weight of couriers sent from a particular city.
SELECT SUM(C.Weight) AS Total_Weight
FROM Courier C
JOIN Branch B ON C.Branch_ID = B.Branch_ID
WHERE B.City = 'Valsad';
```

Data Output | Messages | Notifications

	total_weight numeric
1	99

8) Get the top 5 most frequent destination cities for couriers.

Relational Query:

$\pi_{b\ .\ city,\ COUNT\ (*) \rightarrow courier\_count}(_{b.city}\mathcal{F}_{COUNT\ (*)}(\rho(C,courier) \bowtie_{c\ .\ to\_branch\ =\ b\ .\ branch\_id}\ \rho(b,branch)))$

SQL Query & Output:

```sql
-- Get the top 5 most frequent destination cities for couriers.
SELECT B.City, COUNT(*) AS Courier_Count
FROM Courier C
JOIN Branch B ON C.To_Branch = B.Branch_ID
GROUP BY B.City
ORDER BY Courier_Count DESC
LIMIT 5;
```

Data Output | Messages | Notifications

	city character varying (30)	courier_count bigint
1	Vadodara	4
2	Valsad	4
3	Bharuch	3
4	Surat	2
5	Mumbai	2

9) Get the average price per courier for all delivery partners.

Relational Query:

$\pi_{\text{AVG (price\_per\_courier)} \rightarrow \text{average\_price\_per\_courier}}(\mathcal{F}_{\text{AVG (price\_per\_courier)}}(\text{delivery\_partner}))$

SQL Query & Output:

```
34  SELECT AVG(Price_per_courier) AS Average_Price_Per_Courier
35  FROM delivery_partner
```

Data Output | Messages | Notifications

	average_price_per_courier numeric
1	50.3750000000000000

10) Get the profit earned by a branch in a particular month and a year.

Relational Query:

$\pi_{\text{b . city, SUM (c.profit)} \rightarrow \text{total\_profit}}(_{\text{b.city}}\mathcal{F}_{\text{SUM (c.profit)}}(\sigma_{\text{month = 11 AND year = 2023}}(\rho(C,\text{courier}) \bowtie_{\text{c . branch\_id = b . branch\_id}} \rho(b,\text{branch}))))$

SQL Query & Output:

```
48  -- Get the profit earned by a branch in a month and a year.
49  SELECT B.City, SUM(C.Profit) AS Total_Profit
50  FROM Courier C
51  JOIN Branch B ON C.Branch_ID = B.Branch_ID
52  WHERE EXTRACT(MONTH FROM C.Date) = 11
53    AND EXTRACT(YEAR FROM C.Date) = 2023
54  GROUP BY B.City
55  ORDER BY total_profit DESC;
```

Data Output | Messages | Notifications

	city character varying (30)	total_profit numeric
1	New Delhi	17400
2	Bhavnagar	2340
3	Vadodara	1650
4	Mumbai	700
5	Valsad	197

11) Get the total courier and average weight shipped by a given Vehicle. (Ex. GJ06KL0123)

Relational Query:

π COUNT (*) → total_couriers, AVG (weight) → average_weight
($\mathcal{F}$ COUNT (*),AVG (weight)((σ vehicle_id = "GJ06KL0123"(courier))))

SQL Query & Output:

```
58  SELECT COUNT(*) AS Total_Couriers, AVG(Weight) AS Average_Weight
59  FROM Courier
60  WHERE Vehicle_ID = 'GJ06KL0123';
```

Data Output | Messages | Notifications

	total_couriers bigint	average_weight numeric
1	2	3.0000000000000000

12) Get the total revenue generated by each branch, showing only branches where revenue exceeds 5000 Rupees.

Relational Query:

π b . city, c . branch_id, SUM (c.price) → total_revenue
σ SUM (c.price) > 5000($_{\text{b.city, c.branch\_id}}\mathcal{F}$ SUM (c.price)(ρ(C,courier) $\bowtie$ c . branch_id = b . branch_id ρ(b,branch)))

SQL Query & Output:

```
64  SELECT B.City, C.Branch_ID, SUM(C.Price) AS Total_Revenue
65  FROM Courier C
66  JOIN Branch B ON C.Branch_ID = B.Branch_ID
67  GROUP BY B.City, C.Branch_ID
68  HAVING SUM(C.Price) > 5000
69  ORDER BY total_revenue DESC;
```

Data Output | Messages | Notifications

	city character varying (30)	branch_id numeric (10)	total_revenue numeric
1	New Delhi	2310010930	123400
2	Valsad	2310110930	44773
3	Vadodara	2310150930	30750
4	Bhavnagar	2310140930	12838
5	Rajkot	2310130930	12025
6	Surat	2310120930	7770
7	Mumbai	2310020930	5460

13) Get the number of couriers sent by each customer and show only those who have sent more than 2 couriers.

Relational Query:

$\pi_{\text{customer\_id, COUNT (*)} \rightarrow \text{total\_couriers}}(\sigma_{\text{COUNT (*)} > 2}(_{\text{customer\_id}}\mathcal{F}_{\text{COUNT (*)}}(\text{courier})))$

SQL Query & Output:

```
73  SELECT Customer_ID, COUNT(*) AS Total_Couriers
74  FROM Courier
75  GROUP BY Customer_ID
76  HAVING COUNT(*) > 2;
```

Data Output | Messages | Notifications

	customer_id numeric (11)	total_couriers bigint
1	23110100003	3
2	23110100007	3
3	23110100010	3

14) Find branches that have handled more than 1 courier in a specific month.

Relational Query:

$\pi_{\text{branch\_id, COUNT (*)} \rightarrow \text{courier\_count}}(\sigma_{\text{COUNT (*)} > 1}(_{\text{branch\_id}}\mathcal{F}_{\text{COUNT (*)}}(\sigma_{\textit{month} = 11}(\text{courier}))))$

SQL Query & Output:

```
79  SELECT Branch_ID, COUNT(*) AS Courier_Count
80  FROM Courier
81  WHERE EXTRACT(MONTH FROM Date) = 11
82  GROUP BY Branch_ID
83  HAVING COUNT(*) > 1;
84
```

Data Output | Messages | Notifications

	branch_id numeric (10)	courier_count bigint
1	2310110930	2

15) Count of Couriers by Type and Status:

Relational Query:

$\pi_{c.type,\ c.status,\ COUNT(reference\_id) \rightarrow courier\_count}(_{type,\ status}\mathcal{F}_{COUNT(c.reference\_id)}(\rho(C,courier)))$

SQL Query & Output:

```
200  -- Count of Couriers By Type and Status
201  SELECT C.Type, C.Status, COUNT(C.Reference_ID) AS Courier_Count
202  FROM Courier C
203  GROUP BY C.Type, C.Status
204  ORDER BY C.Type, C.Status;
```

Data Output | Messages | Notifications

	type character varying (16)	status character varying (9)	courier_count bigint
1	Express	Arrived	1
2	Express	Collected	2
3	Express	Lost	1
4	Express	Picked_up	1
5	Express	Return	1
6	Express	Shipped	1
7	Handle With Care	Arrived	1
8	Handle With Care	Collected	1
9	Handle With Care	Picked_up	1
10	Handle With Care	Return	1
11	Handle With Care	Shipped	2
12	Standard	Collected	1
13	Standard	Lost	2
14	Standard	Picked_up	1
15	Standard	Return	1
16	Standard	Shipped	2

16) Most Frequently Used Payment Methods.

Relational Query:

$\pi_{p.method,\ COUNT(p.payment\_id) \rightarrow cnt}(_{p.method}\mathcal{F}_{COUNT(p.payment\_id)}(\rho(p,payment)))$

SQL Query & Output:

```
212  SELECT P.Method, COUNT(P.Payment_ID) AS cnt
213  FROM Payment P
214  GROUP BY P.Method
215  ORDER BY cnt DESC;
```

Data Output | Messages | Notifications

	method character varying (17)	cnt bigint
1	Net Banking	5
2	UPI	5
3	Credit/Debit Card	4
4	Cash	3

17) Total Number of Couriers Handled by Each Delivery Partner

Relational Query:

$\pi_{dp.name,\ COUNT(cm.reference\_id) \rightarrow cnt}(_{name}\mathcal{F}_{COUNT(cm.reference\_id)}(\rho(dp,delivery\_partner) \bowtie_{dp.registration\_number = cm.registration\_number} \rho(cm,couriermapping)))$

SQL Query & Output:

```
218  SELECT DP.Name, COUNT(CM.Reference_ID) AS cnt
219  FROM Delivery_Partner DP
220  JOIN CourierMapping CM ON DP.Registration_Number = CM.Registration_Number
221  GROUP BY DP.Name
222  ORDER BY cnt DESC;
```

Data Output | Messages | Notifications

	name character varying (50)	cnt bigint
1	FastDeliver Co.	4
2	Parcel Path	3
3	Speedy Couriers	2
4	GoExpress Services	2
5	QuickShip Ltd.	2
6	Courier Express	2
7	OnTime Logistics	1
8	Eagle Delivery	1

18) Most Profitable Courier Types

Relational Query:

$\pi_{c.type,\ SUM(c.profit) \rightarrow total\_profit}(_{c.type}\mathcal{F}_{SUM(c.profit)}\ (\rho(C,courier)))$

SQL Query & Output:

```
225  SELECT C.Type, SUM(C.Profit) AS Total_Profit
226  FROM Courier C
227  GROUP BY C.Type
228  ORDER BY Total_Profit DESC;
```

Data Output | Messages | Notifications

	type character varying (16)	total_profit numeric
1	Handle With Care	39515
2	Express	5382
3	Standard	2553

19) Average Delivery Time by Type.

Relational Query:

$\pi_{type,\ AVG(Expected\_Delivery\_Date - Date) \rightarrow Average\_Delivery\_Time}(_{type}\mathcal{F}_{AVG(Expected\_Delivery\_Date - Date)}\ (courier))$

SQL Query & Output:

```
246  -- Average Delivery Time by Type
247  SELECT Type, CEIL(AVG((Expected_Delivery_Date - Date))) AS Average_Delivery_Time
248  FROM Courier
249  GROUP BY Type;
```

Data Output | Messages | Notifications

	type character varying (16)	average_delivery_time numeric
1	Standard	4
2	Handle With Care	6
3	Express	2

- **Queries having nested queries:**

20) Find vehicles that have not been assigned to any courier.

Relational Query:

$\pi_{vehicle\_id}$ (Vehicle) EXCEPT $\pi_{vehicle\_id}$ (courier)

SQL Query & Output:

```
85  -- Find vehicles that have not been assigned to any courier
86  SELECT Vehicle_id
87  FROM Vehicle
88  WHERE Vehicle_id NOT IN (SELECT Vehicle_ID FROM Courier);
```

Data Output | Messages | Notifications

	vehicle_id [PK] character
1	GJ01AB1234
2	GJ04GH2345
3	GJ08OP8910
4	GJ12WX3456

21) Retrieve all customers who have not provided feedback.

Relational Query:

$\pi_{Name,\ Contact\_Of\_Sender}(\sigma_{customer\_ID\ EXCEPT\ \pi_{Customer\_ID}(Feedback)}(Customer))$

SQL Query & Output:

```
90  -- Retrieve all customers who have not provided feedback
91  SELECT Name, Contact_Of_Sender
92  FROM Customer
93  WHERE Customer_ID NOT IN (SELECT Customer_ID FROM Feedback);
```

Data Output | Messages | Notifications

	name character varying (50)	contact_of_sender numeric (10)
1	Suresh Iyer	9001234567
2	Shalini Reddy	9331456789

- **Queries having aggregation in nested queries:**

22) Find couriers that have a distance greater than the average distance of couriers in the same branch

Relational Query:

$\pi_{\text{reference\_id, distance}}(\sigma_{\text{distance} > (\pi_{\text{AVG (distance)}}(\mathcal{F}_{\text{AVG (distance)}}(\sigma_{\text{c1 . branch\_id = c2 . branch\_id}}(\rho(\text{c2,courier})))))}(\rho(\text{c1,courier})))$

SQL Query & Output:

```
124  -- Find couriers that have a distance greater than the average distance of couriers
125  -- in the same branch
126  SELECT Reference_ID, Distance
127  FROM Courier c1
128  WHERE Distance > (
129      SELECT AVG(Distance)
130      FROM Courier c2
131      WHERE c1.Branch_ID = c2.Branch_ID
132  );
```

Data Output | Messages | Notifications

	reference_id [PK] character	distance numeric (4)
1	23110100ad	1000
2	23110100af	750
3	23120100ac	550
4	23120100af	1250
5	24010100aa	500
6	24010100ae	650
7	23110100ac	1450
8	24010100ah	650

23) Retrieve the vehicles with a capacity greater than the average capacity of all vehicles

Relational Query:

$\pi_{\text{vehicle\_id, capacity}}(\sigma_{\text{capacity} > (\pi_{\text{AVG (capacity)}}(\mathcal{F}_{\text{AVG (capacity)}}(\text{vehicle})))}(\text{vehicle}))$

SQL Query & Output:

```
119  -- Retrieve the vehicles with a capacity greater than
120  -- the average capacity of all vehicles
121  SELECT Vehicle_ID, Capacity
122  FROM Vehicle
123  WHERE Capacity > (SELECT AVG(Capacity) FROM Vehicle);
124
```

Data Output | Messages | Notifications

	vehicle_id [PK] character	capacity numeric (4)
1	GJ02CD5678	1200
2	GJ04GH2345	1800
3	GJ05IJ6789	2000
4	GJ08OP8910	1300
5	GJ12WX3456	1600
6	RJ14AB1234	1900
7	MP09EF9101	1350
8	TN07IJ6789	2000
9	DL04KL0123	1400

24) List customers whose total payments are higher than the average total payment of all customers

Relational Query:

$\pi_{\text{customer\_id, SUM (amount)} \rightarrow \text{total\_payment}}(\sigma_{\text{SUM (amount)} > (\pi_{\text{AVG (amount)}} (\mathcal{F}_{\text{AVG (amount)}} (\text{payment})))} (_{\text{customer\_id}}\mathcal{F}_{\text{SUM (amount)}}(\rho(p,\text{payment}) \bowtie_{\text{p . reference\_id = c . reference\_id}} \rho(C,\text{courier}))))$

SQL Query & Output:

```
101  -- List customers whose total payments are higher than
102  -- the average total payment of all customers
103  SELECT Customer_ID, SUM(Amount) AS Total_Payment
104  FROM Payment p
105  JOIN Courier c ON p.Reference_ID = c.Reference_ID
106  GROUP BY Customer_ID
107  HAVING SUM(Amount) > (SELECT AVG(Amount) FROM Payment);
```

Data Output | Messages | Notifications

	customer_id numeric (11)	total_payment numeric
1	23110100009	72150
2	23110100011	87000
3	23110100004	22500

25) Get the details of couriers that have a higher price than the average price of all couriers

Relational Query:

$\pi_{\text{reference\_id, price}}(\sigma_{\text{price} > (\pi_{\text{AVG (price)}} (\mathcal{F}_{\text{AVG (price)}} (\text{courier})))} (\text{Courier}))$

SQL Query & Output:

```
98   -- Get the details of couriers that have a higher
99   -- price than the average price of all couriers
100  SELECT Reference_ID, Price
101  FROM Courier
102  WHERE Price > (SELECT AVG(Price) FROM Courier);
```

Data Output | Messages | Notifications

	reference_id [PK] character	price numeric (5)
1	23120100ac	35750
2	23120100af	22500
3	24010100ac	36400
4	23110100ac	87000

- **Correlated queries:**

26) Find names of delivery partners who have delivered at least 3 couriers

Relational Query:

$\pi_{dp.name}(\sigma_{dp.registration\_number\ SEMI\text{-}INTERSECTION_{<dp.registration\_number=cm.registration\_number>}}$

$(\pi_{cm.registration\_number}(\sigma_{COUNT(cm.reference\_id)>=3}({}_{cm.registration\_number}\mathcal{F}_{COUNT(cm.reference\_id)}(\sigma_{dp.registration\_number=cm.registration\_number}(\rho(cm,couriermapping))))))$

$(\rho(dp,delivery\_partner)))$

SQL Query & Output:

```
228  -- Find names of delivery partner who have delivered at least 3 courier
229  SELECT dp.Name
230  FROM Delivery_Partner dp
231  WHERE dp.Registration_Number IN (
232      SELECT cm.Registration_Number
233      FROM CourierMapping cm where dp.Registration_Number = cm.Registration_Number
234      GROUP BY cm.Registration_Number
235      HAVING COUNT(cm.Reference_ID) >= 3
236  );
```

Data Output | Messages | Notifications

	name character varying (50)
1	FastDeliver Co.
2	Parcel Path

27) Find details of drivers who have rating less than certain average rating.

Relational Query:

$\pi_{e.*,d.vehicle\_id}(\sigma_{d.id=\pi_{d2.id}(\sigma_{d2.rating<(\pi_{AVG(rating)/1.2}(\mathcal{F}_{AVG(rating)}(driver))}({}_{d2.id}\mathcal{F}(\sigma_{d.id=d2.id}(\rho(d2,driver)))))}$
$(\rho(d,driver) \bowtie_{d.id=e.employee\_id} \rho(e,employee)))$

SQL Query & Output:

```
238  -- Find details of drivers who have rating less than certain average rating (semi-join)
239  SELECT e.*,d.vehicle_id
240  FROM Driver d
241  JOIN Employee e ON d.ID = e.Employee_id
242  WHERE d.ID IN (
243      SELECT d2.ID
244      FROM Driver d2 where d.ID = d2.ID
245      GROUP BY d2.ID
246      HAVING d2.Rating < (SELECT AVG(Rating)/1.2 FROM Driver)
247  );
248
```

Data Output | Messages | Notifications

	employee_id numeric (10)	name character varying (50)	contact_number numeric (10)	date_of_joining date	role character varying (10)	branch_id numeric (10)	vehicle_id character
1	2310120002	Rahul Joshi	3210987653	2023-10-25	Driver	2310120930	GJ05IJ6789

- **Queries with division operation:**

28) Retrieve driver details who have driven couriers of all types.

Relational Query:

$\pi_{Employee.*}(\sigma_{e.Employee\_ID= d.ID}$ (ρ(e,Employee) ⋈
(π_{ID}(d) EXCEPT π_{ID} (r2)(ρ (r2, ($\pi_{d.ID->ID,\ cour.type\ ->\ type}$(ρ (d,Driver) X ($\pi_{Type}$ ρ(cour,Courier)))
EXCEPT $\pi_{d.ID,\ c.Type}$(ρ(d,Driver) ⋈ $_{<d.vehicle\_id=c.vehicle\_id>}$ ρ (c,Courier)))))))

SQL Query & Output:

```
-- Retrieve driver details who have driven couriers of all types
SELECT *
FROM Employee AS e
WHERE e.employee_id = (
    SELECT DISTINCT d.id
    FROM Driver d
    WHERE d.ID NOT IN (
        SELECT ID
        FROM (
            SELECT d.ID AS ID, cour.Type AS Type
            FROM Driver d
            CROSS JOIN (SELECT DISTINCT Type FROM Courier) AS cour
            EXCEPT
            SELECT d.ID, c.Type
            FROM Driver d
            JOIN Courier c ON d.Vehicle_ID = c.Vehicle_ID
        ) AS r2
    )
);
```

Data Output | Messages | Notifications

	employee_id [PK] numeric (10)	name character varying (50)	contact_number numeric (10)	date_of_joining date	role character varying (10)	branch_id numeric (10)
1	2310030002	Suresh Reddy	8765432105	2023-10-25	Driver	2310030930

29) Find branches that have managed couriers with all possible status.

Relational Query:

$\sigma_{bcv.branch\_ID = b.branch\_ID}$ (ρ(b,Branch) ⋈
(π_{Branch_ID}(BCV) EXCEPT π_{Branch_ID} (r2)(ρ (r2, ($\pi_{b.branch\_id->branch\_id,\ cour.status\ ->\ status}$(ρ (b,Branch) X ($\pi_{Status}$ ρ(cour,Courier))) EXCEPT $\pi_{bcv.branch\_ID,\ bcv.Status}$(BCV)))))

SQL Query & Output:

```sql
-- Find branches that have managed couriers with all possible status
CREATE VIEW BCV AS
SELECT b.Branch_ID, c.Status
FROM Branch b
JOIN Courier c ON b.Branch_ID = c.Branch_ID;

SELECT * FROM branch AS b
WHERE b.branch_id = (
    SELECT DISTINCT bcv.Branch_ID FROM BCV
    WHERE Branch_ID NOT IN (
        SELECT Branch_ID
        FROM (
            SELECT b.Branch_ID AS Branch_ID, cour.Status AS Status
            FROM Branch b
            CROSS JOIN (SELECT DISTINCT Status FROM Courier ) AS cour
            EXCEPT
            SELECT bcv.Branch_ID, bcv.Status
            FROM BCV
        ) AS r2
    )
);
```

Data Output | Messages | Notifications

	branch_id [PK] numeric (10)	city character varying (30)	pincode numeric (6)
1	2310110930	Valsad	396001

30) Find Promotion Companies Details which are promoted by all the vehicles.

Relational Query:

$\sigma_{promotion\_ID= promotion\_ID}$ (promotion $\bowtie$ ($\pi_{promotion\_ID}$(VehiclePromotionMapping) EXCEPT $\pi_{promotion\_ID}$ (r2)(ρ (r2, ($\pi_{v.vehicle\_id->vid, vpm.promotion\_id}$(ρ (v,Vehicle) X (ρ(vpm,VehiclePromotionMapping))) EXCEPT $\pi_{vpm.vehicle\_id,vpm.promotion\_id}$(ρ(vpm,VehiclePromotionMapping))))))

SQL Query & Output:

```
176  -- Find Promotion Companies Details which are promoted by all the vehicles
177  SELECT * FROM promotion
178  WHERE promotion_id = (
179      SELECT DISTINCT promotion_id
180      FROM VehiclePromotionMapping
181      WHERE promotion_id NOT IN (
182          SELECT promotion_id
183          FROM (
184              SELECT v.Vehicle_id AS vid, vpm.promotion_id
185              FROM Vehicle v
186              CROSS JOIN (SELECT * FROM VehiclePromotionMapping) AS vpm
187              EXCEPT
188              SELECT vpm.Vehicle_id, vpm.promotion_id
189              FROM VehiclePromotionMapping AS vpm
190          ) AS r2
191      )
192  );
```

Data Output | Messages | Notifications

	promotion_id [PK] character	price numeric (7)	company_name character varying (50)
1	231101abcd	15000	Amul

Views:

- We have created One View to use it in query.

```
156 CREATE VIEW BCV AS
157 SELECT b.Branch_ID, c.Status
158 FROM Branch b
159 JOIN Courier c ON b.Branch_ID = c.Branch_ID;
```

Data Output | Messages | Notifications

	branch_id numeric (10)	status character varying (9)
1	2310020930	Shipped
2	2310150930	Shipped
3	2310140930	Shipped
4	2310130930	Arrived
5	2310110930	Arrived
6	2310120930	Picked_up
7	2310110930	Picked_up
8	2310150930	Picked_up
9	2310110930	Return
10	2310140930	Return
11	2310010930	Return
12	2310020930	Lost
13	2310120930	Lost
14	2310010930	Collected
15	2310110930	Collected
16	2310180930	Collected
17	2310110930	Collected
18	2310130930	Shipped
19	2310110930	Lost
20	2310110930	Shipped