SQL JOIN Exercises – Online Business Case Study

-By SUNDAR

Short Description

This case study is designed to practice SQL JOINs using a realistic online shopping database.
It helps understand how different tables like customers, orders, products, categories, order items, and payments are connected and how business insights can be extracted using SQL queries.

The exercises progress from basic JOINs to advanced real-world analytics, covering order tracking, payment status, revenue analysis, and customer behavior.
This case study is useful for students, beginners, and interview preparation.

Database Tables Used

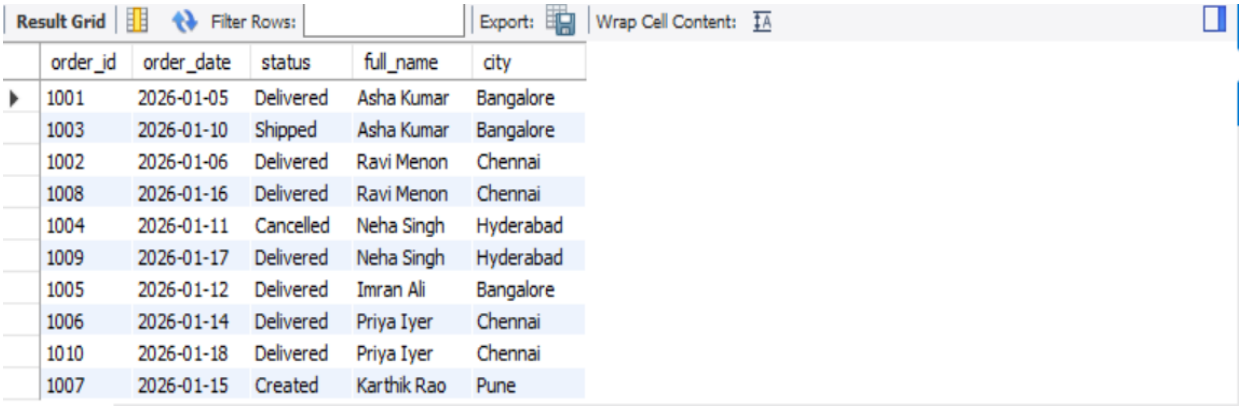
- Customers
- Orders
- Order Items
- Products
- Categories
- Payments

SECTION-A **Basic JOINs (Customers & Orders)**

1.Display all orders along with customer name and city

```
SELECT
        orders.order_id,orders.order_date,orders.status, customers.full_name,customers.city
FROM orders
JOIN customers
ON orders.customer_id = customers.customer_id;
```

Output



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	order_date	status	full_name	city
1001	2026-01-05	Delivered	Asha Kumar	Bangalore
1003	2026-01-10	Shipped	Asha Kumar	Bangalore
1002	2026-01-06	Delivered	Ravi Menon	Chennai
1008	2026-01-16	Delivered	Ravi Menon	Chennai
1004	2026-01-11	Cancelled	Neha Singh	Hyderabad
1009	2026-01-17	Delivered	Neha Singh	Hyderabad
1005	2026-01-12	Delivered	Imran Ali	Bangalore
1006	2026-01-14	Delivered	Priya Iyer	Chennai
1010	2026-01-18	Delivered	Priya Iyer	Chennai
1007	2026-01-15	Created	Karthik Rao	Pune

2.Show all orders placed by customers from Bangalore

```
SELECT
   orders.order_id,orders.order_date, orders.status,customers.full_name,
   customers.city
FROM customers
JOIN orders
ON customers.customer_id = customers.customer_id
WHERE customers.city = 'Bangalore';
```

Output

Preview 'output.csv'

Order_id	Order_date	Status	Full_name	City
1001	2026-01-05	Delivered	Imran Ali	Bangalore
1001	2026-01-05	Delivered	Asha Kumar	Bangalore
1002	2026-01-06	Delivered	Imran Ali	Bangalore
1002	2026-01-06	Delivered	Asha Kumar	Bangalore
1003	2026-01-10	Shipped	Imran Ali	Bangalore
1003	2026-01-10	Shipped	Asha Kumar	Bangalore
1004	2026-01-11	Cancelled	Imran Ali	Bangalore
1004	2026-01-11	Cancelled	Asha Kumar	Bangalore
1005	2026-01-12	Delivered	Imran Ali	Bangalore
1005	2026-01-12	Delivered	Asha Kumar	Bangalore
1006	2026-01-14	Delivered	Imran Ali	Bangalore
1006	2026-01-14	Delivered	Asha Kumar	Bangalore
1007	2026-01-15	Created	Imran Ali	Bangalore
1007	2026-01-15	Created	Asha Kumar	Bangalore
1008	2026-01-16	Delivered	Imran Ali	Bangalore
1008	2026-01-16	Delivered	Asha Kumar	Bangalore
1009	2026-01-17	Delivered	Imran Ali	Bangalore
1009	2026-01-17	Delivered	Asha Kumar	Bangalore
1010	2026-01-18	Delivered	Imran Ali	Bangalore
1010	2026-01-18	Delivered	Asha Kumar	Bangalore

3.List only Delivered orders with customer name and order date

```
SELECT
   customers.full_name,  orders.order_date,orders.status
FROM customers JOIN orders
ON customers.customer_id = orders.customer_id
WHERE orders.status = 'Delivered';
```

Output

Full_name	Order_date	Status
Asha Kumar	2026-01-05	Delivered
Imran Ali	2026-01-12	Delivered
Neha Singh	2026-01-17	Delivered
Priya Iyer	2026-01-14	Delivered
Priya Iyer	2026-01-18	Delivered
Ravi Menon	2026-01-06	Delivered
Ravi Menon	2026-01-16	Delivered

4.Find customers who have placed at least one order

```
SELECT DISTINCT customers.customer_id, customers.full_name

FROM customers

JOIN orders

ON customers.customer_id = orders.customer_id;
```

Output

Customer_id	Full_name
1	Asha Kumar
2	Ravi Menon
3	Neha Singh
4	Imran Ali
5	Priya Iyer
6	Karthik Rao

5.Find customers who have never placed any order

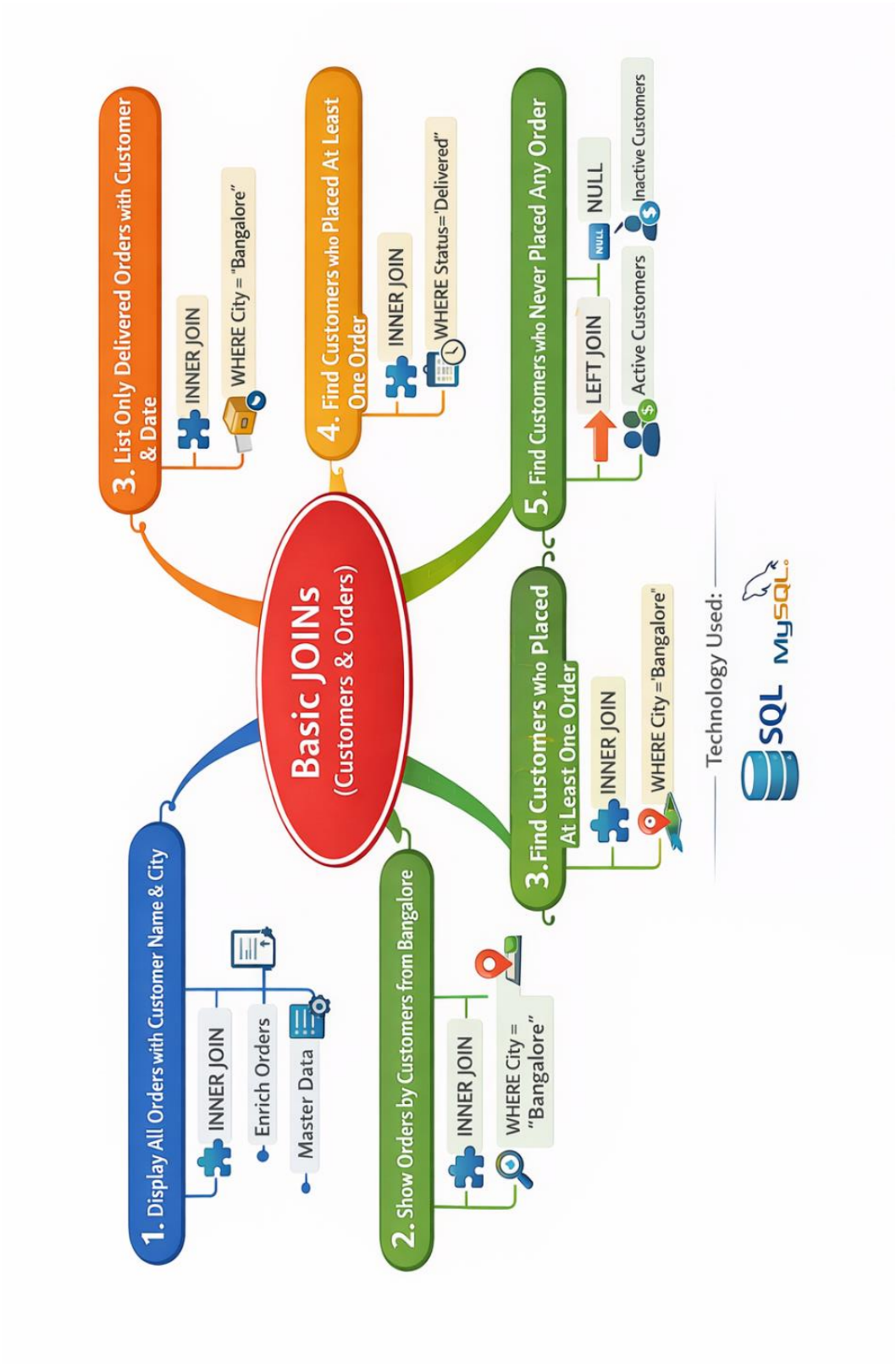
```
SELECT customers.customer_id, customers.full_name

FROM customers LEFT JOIN orders

ON customers.customer_id = orders.customer_id

WHERE orders.customer_id IS NULL;
```

Output

Result Grid | Filter Rows:

customer_id	full_name

SUMMARY FOR SECTION-A



SECTION-B Orders & Products (Multi-table JOINs)

6.Display order_id, product_name, quantity, and unit_price

```
SELECT
   oi.order_id,p.product_name,oi.quantity,oi.unit_price
FROM order_items oi JOIN products p
ON oi.product_id = p.product_id;
```

Output:

Order_id	Product_name	Quantity	Unit_price
1001	Wireless Mouse	1	799
1006	Wireless Mouse	1	799
1001	USB-C Charger	1	999
1008	USB-C Charger	1	999
1002	T-Shirt	2	499
1006	T-Shirt	1	499
1003	Jeans	1	1499
1010	Jeans	1	1499
1002	Water Bottle	1	299
1008	Water Bottle	2	299
1005	Cooking Pan	1	1299
1010	Cooking Pan	1	1299
1004	Novel - Mystery	1	399
1009	Novel - Mystery	3	399
1005	Face Wash	2	249

7.Calculate line total for each product in an order (quantity × unit_price)

```
SELECT
   oi.order_id,p.product_name,oi.quantity,oi.unit_price,
   (oi.quantity * oi.unit_price) AS line_total
   FROM order_items oiJOIN products p
   ON oi.product_id = p.product_id;
```

Output:

Order_id	Product_name	Quantity	Unit_price	Line_total
1001	Wireless Mouse	1	799	799
1006	Wireless Mouse	1	799	799
1001	USB-C Charger	1	999	999
1008	USB-C Charger	1	999	999
1002	T-Shirt	2	499	998
1006	T-Shirt	1	499	499
1003	Jeans	1	1499	1499
1010	Jeans	1	1499	1499
1002	Water Bottle	1	299	299
1008	Water Bottle	2	299	598
1005	Cooking Pan	1	1299	1299
1010	Cooking Pan	1	1299	1299
1004	Novel - Mystery	1	399	399
1009	Novel - Mystery	3	399	1197
1005	Face Wash	2	249	498

8.Calculate total order amount for each order

SELECT

oi.order_id, SUM (oi.quantity * oi.unit_price) AS Sum_of_order_amount

FROM order_items oi GROUP BY oi.order_id;

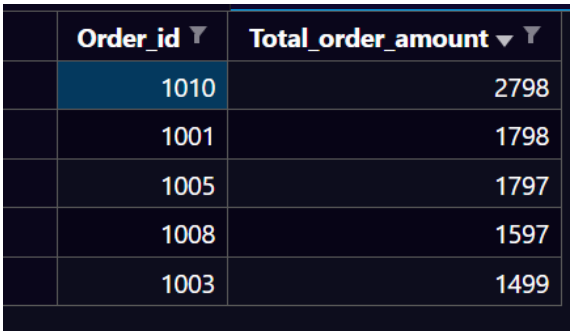
Output:

Order_id	Sum_of_order_amount
1001	1798
1002	1297
1003	1499
1004	399
1005	1797
1006	1298
1008	1597
1009	1197
1010	2798

9.List the top 5 highest value orders

```
SELECT
   oi.order_id,  SUM(oi.quantity * oi.unit_price) AS total_order_amount
FROM order_items oi  GROUP BY oi.order_id
ORDER BY total_order_amount  DESC
LIMIT 5;
```

Output:



Order_id	Total_order_amount
1010	2798
1001	1798
1005	1797
1008	1597
1003	1499

10.Display order details along with product name and category name

```
SELECT
   o.order_id, o.order_date, o.status, p.product_name, c.category_name
FROM orders o
JOIN order_items o   ON o.order_id = oi.order_id
JOIN products p  ON oi.product_id = p.product_id
JOIN categories c  ON p.category_id = c.category_id;
```

Output:

Order_id	Order_date	Status	Product_name	Category_name
1001	2026-01-05	Delivered	Wireless Mouse	Electronics
1006	2026-01-14	Delivered	Wireless Mouse	Electronics
1001	2026-01-05	Delivered	USB-C Charger	Electronics
1008	2026-01-16	Delivered	USB-C Charger	Electronics
1002	2026-01-06	Delivered	T-Shirt	Fashion
1006	2026-01-14	Delivered	T-Shirt	Fashion
1003	2026-01-10	Shipped	Jeans	Fashion
1010	2026-01-18	Delivered	Jeans	Fashion
1002	2026-01-06	Delivered	Water Bottle	Home
1008	2026-01-16	Delivered	Water Bottle	Home
1005	2026-01-12	Delivered	Cooking Pan	Home
1010	2026-01-18	Delivered	Cooking Pan	Home
1004	2026-01-11	Cancelled	Novel - Mystery	Books
1009	2026-01-17	Delivered	Novel - Mystery	Books
1005	2026-01-12	Delivered	Face Wash	Beauty

SUMMARY FOR SECTION-B

SECTION-C: Product & Category Analysis

11.Display all products with their category name

```
SELECT
    p.product_id, p.product_name, c.category_name
FROM products p  JOIN categories c
ON p.category_id = c.category_id;
```

Output:

Product_id	Product_name	Category_name
101	Wireless Mouse	Electronics
102	USB-C Charger	Electronics
103	T-Shirt	Fashion
104	Jeans	Fashion
105	Water Bottle	Home
106	Cooking Pan	Home
107	Novel - Mystery	Books
108	Face Wash	Beauty

12.Find categories that have no products

```
SELECT
    c.category_id, c.category_name
FROM categories c   LEFT JOIN products p   ON c.category_id = p.category_id
WHERE p.product_id IS NULL;
```

Output:

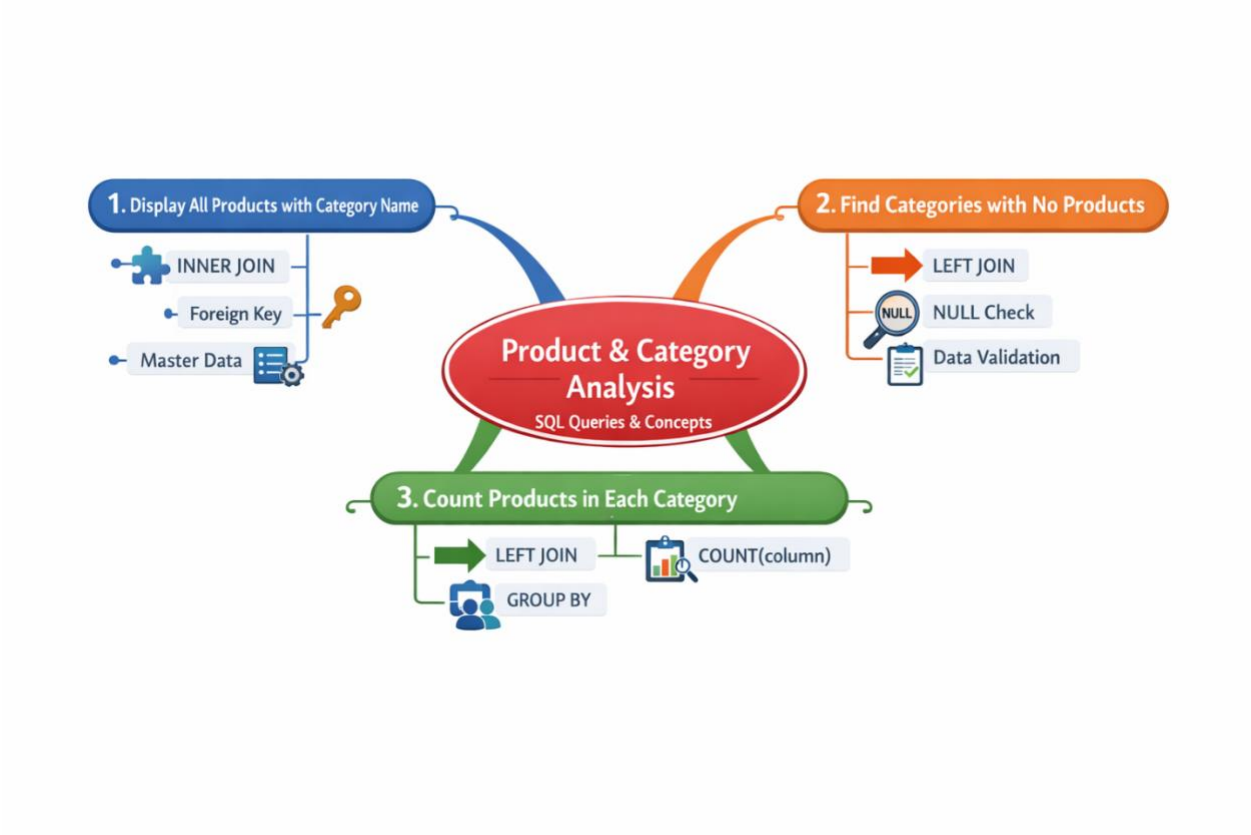
Category_id	Category_name

13.Count number of products in each category

```
SELECT
   c.category_name, COUNT(p.product_id) AS product_count  FROM categories c
LEFT JOIN products p  ON c.category_id = p.category_id
GROUP BY c.category_name;
```

Output:

Category_name	Product_count
Electronics	2
Fashion	2
Home	2
Books	1
Beauty	1



SECTION D – Payments & Order Status

14. Display all orders along with payment status

```
SELECT o.order_id,
CASE
WHEN p.payment_id IS NULL
THEN 'Not Paid'   ELSE 'Paid'
END AS payment_status
FROM orders o  LEFT JOIN payments p
ON o.order_id=p.order_id;
```

Output:

	Order_id	Order_date	Status	Payment_status
	1001	2026-01-05	Delivered	Paid
	1002	2026-01-06	Delivered	Paid
	1003	2026-01-10	Shipped	Paid
	1004	2026-01-11	Cancelled	Not Paid
	1005	2026-01-12	Delivered	Paid
	1006	2026-01-14	Delivered	Paid
	1007	2026-01-15	Created	Not Paid
	1008	2026-01-16	Delivered	Paid
	1009	2026-01-17	Delivered	Paid
	1010	2026-01-18	Delivered	Not Paid

15.Find orders where payment amount is less than order total

```
SELECT
    o.order_id,  o.order_date,  o.total_amount AS order_total,
  SUM(p.payment_amount) AS total_paid
FROM orders o   JOIN payments p  ON o.order_id = p.order_id
GROUP BY o.order_id, o.order_date, o.total_amount
HAVING SUM(p.payment_amount) < o.total_amount;
```

Output:

	Order_id	Status	Order_total	Total_paid
	1003	Shipped	1499	500
	1004	Cancelled	399	0
	1008	Delivered	1597	1000
	1010	Delivered	2798	0

---

16. Identify orders that have multiple payments

```
SELECT order_id, COUNT(payment_id) AS total_payments
FROM payments
GROUP BY order_id
HAVING COUNT(payment_id)>1;
```

Output:

	Order_id	Payment_count
	1005	2

---

17. Calculate total amount paid for each order

SELECT order_id, SUM(paid_amount) AS total_paid

FROM payments GROUP BY order_id;

Output:

	Order_id	Total_paid
	1001	1798
	1002	1297
	1003	500
	1005	1797
	1006	1298
	1008	500
	1009	1197

---

18. Find Delivered orders that are not paid yet

SELECT o.order_id

FROM orders o

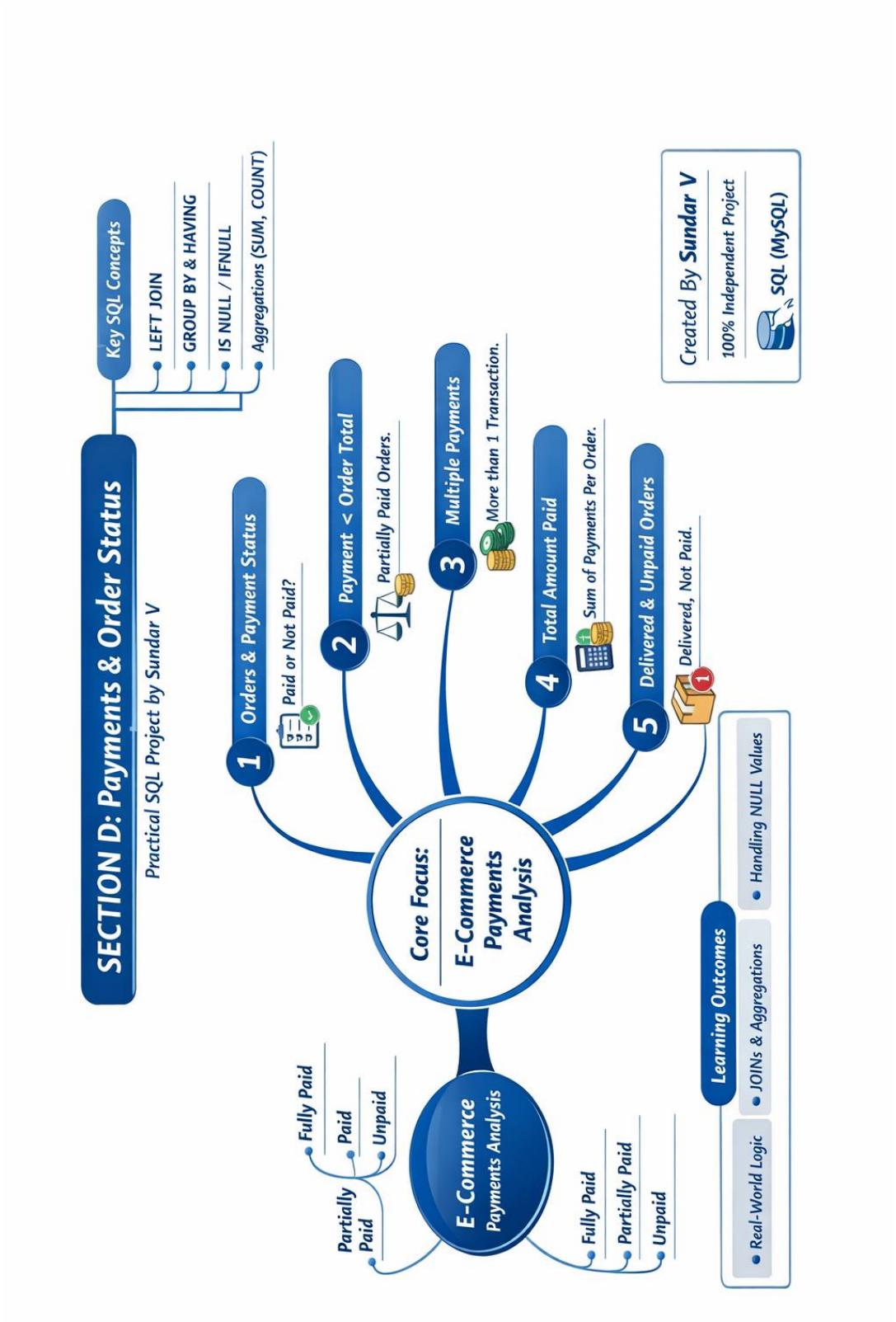
LEFT JOIN payments p ON o.order_id=p.order_id

WHERE o.status='Delivered' AND p.order_id IS NULL;

Output:

	Order_id	Status	Total_paid
	1010	Delivered	0

SUMMARY FOR SECTION-D



SECTION E: Business Insights

19.Calculate total revenue generated by each customer

```
SELECT
   c.customer_id,  c.full_name,
   SUM(oi.quantity * oi.unit_price) AS total_revenue
FROM customers c  JOIN orders o  ON c.customer_id = o.customer_id
JOIN order_items oi  ON o.order_id = oi.order_id
GROUP BY c.customer_id, c.full_name   ORDER BY total_revenue DESC;
```

Output:

Customer_id	Full_name	Total_revenue
5	Priya Iyer	4096
1	Asha Kumar	3297
2	Ravi Menon	2894
4	Imran Ali	1797
3	Neha Singh	1596

20.Identify top 3 customers by revenue

```
SELECT
   c.customer_id,  c.full_name,
    SUM(oi.quantity * oi.unit_price) AS total_revenue
FROM customers c  JOIN orders o  ON c.customer_id = o.customer_id
JOIN order_items oi  ON o.order_id = oi.order_id
GROUP BY c.customer_id, c.full_name
ORDER BY total_revenue DESC  LIMIT 3;
```

Output:

	Customer_id	Full_name	Total_revenue
	5	Priya Iyer	4096
	1	Asha Kumar	3297
	2	Ravi Menon	2894

21.Calculate revenue by product category

```
SELECT
   c.category_name,  SUM(oi.quantity * oi.unit_price) AS total_revenue
FROM order_items oi
JOIN products p
        ON oi.product_id = p.product_id
JOIN categories c
         ON p.category_id = c.category_id
GROUP BY c.category_name
ORDER BY  total_revenue
DESC;
```

Output:

	Category_name	Total_revenue
	Fashion	4495
	Electronics	3596
	Home	3495
	Books	1596
	Beauty	498

22.Find the best-selling product based on quantity sold

```
SELECT
   p.product_id, p.product_name,
   SUM(oi.quantity) AS total_quantity_sold FROM order_items oi
JOIN products p  ON oi.product_id = p.product_id
GROUP BY p.product_id, p.product_name  ORDER BY total_quantity_sold DESC
LIMIT 1;
```

Output:

Product_id	Product_name	Total_quantity_sold
107	Novel - Mystery	4

23.Calculate city-wise revenue.

```
SELECT
   c.city,  SUM(oi.quantity * oi.unit_price) AS total_revenue
FROM customers c  JOIN orders o ON c.customer_id = o.customer_id
JOIN order_items oi  ON o.order_id = oi.order_id
GROUP BY c.city  ORDER BYtotal_revenue DESC;
```

Output:

City	Total_revenue
Chennai	6990
Bangalore	5094
Hyderabad	1596

24.Find orders that have no order items (data issue)

```
SELECT
   o.order_id, o.order_date, o.status  FROM orders o
LEFT JOIN order_items oi  ON o.order_id = oi.order_id
WHERE oi.order_id IS NULL;
```

Output:

Order_id	Order_date	Status
1007	2026-01-15	Created

25.Find products that were never sold

```
SELECT
   p.product_id,
   p.product_name
FROM products p
LEFT JOIN order_items oi
   ON p.product_id = oi.product_id
WHERE oi.product_id IS NULL;
```

Output:

Product_id	Product_name

SUMMARY FOR SECTION-E

SECTION F: Advanced / Interview-Level Challenges

26.For each customer, show their latest order date

(include customers with no orders)

```
SELECT
   c.customer_id,
   c.full_name,
   MAX(o.order_date) AS latest_order_date
FROM customers c
LEFT JOIN orders o
   ON c.customer_id = o.customer_id
GROUP BY
   c.customer_id,
   c.full_name;
```

Output:

	Customer_id	Full_name	Latest_order_date
	1	Asha Kumar	2026-01-10
	4	Imran Ali	2026-01-12
	6	Karthik Rao	2026-01-15
	2	Ravi Menon	2026-01-16
	3	Neha Singh	2026-01-17
	5	Priya Iyer	2026-01-18

27.For each category, find the top-selling product

```
SELECT
   category_name,  product_name,  total_quantity_sold
FROM (SELECT c.category_name,  p.product_name,
      SUM(oi.quantity) AS total_quantity_sold, ROW_NUMBER() OVER (
         PARTITION BY c.category_name  ORDER BY SUM(oi.quantity) DESC
    ) AS rn
   FROM order_items oi  JOIN products p  ON oi.product_id = p.product_id
   JOIN categories c
        ON p.category_id = c.category_id
   GROUP BY
         c.category_name,   p.product_name
) ranked_products
WHERE rn = 1;
```

Output:

Category_name	Product_name	Total_quantity_sold
Beauty	Face Wash	2
Books	Novel - Mystery	4
Electronics	Wireless Mouse	2
Fashion	T-Shirt	3
Home	Water Bottle	3

28.Find customers who bought products from at least two different categories.

```
SELECT
   c.customer_id,  c.full_name,
   COUNT(DISTINCT p.category_id) AS category_count
FROM customers c
JOIN orders o
   ON c.customer_id = o.customer_id
JOIN order_items oi
   ON o.order_id = oi.order_id
JOIN products p
   ON oi.product_id = p.product_id
GROUP BY
   c.customer_id, c.full_name
HAVING
   COUNT(DISTINCT p.category_id) >= 2;
```

Output:

Customer_id	Full_name	Category_count
1	Asha Kumar	2
4	Imran Ali	2
2	Ravi Menon	3
5	Priya Iyer	3

29.Identify repeat customers with more than one order

```
SELECT
   c.customer_id, c.full_name,
   COUNT(o.order_id) AS total_orders  FROM customers c
JOIN orders o  ON c.customer_id = o.customer_id
GROUP BY c.customer_id, c.full_name
HAVING  COUNT(o.order_id) > 1;
```

Output:

	Customer_id	Full_name	Total_orders
	1	Asha Kumar	2
	2	Ravi Menon	2
	3	Neha Singh	2
	5	Priya Iyer	2

30.Generate monthly revenue trend.

```
SELECT
   DATE_FORMAT(o.order_date, '%M-%Y') AS order_month,
   SUM(oi.quantity * oi.unit_price) AS monthly_revenue
FROM orders o
JOIN order_items oi  ON o.order_id = oi.order_id
GROUP
 BY  DATE_FORMAT(o.order_date, '%M-%Y')
ORDER BY
   order_month;
```

Output:

	Order_month	Monthly_revenue
	January-2026	13680

SUMMARY FOR SECTION-F