**Sales Order Dataset - Case Study**

Problem Statements:

1. Identify the total no of products sold.
2. Other than Completed, display the available delivery status's.
3. Display the order id, order\_date and product\_name for all the completed orders.
4. Sort the above query to show the earliest orders at the top. Also display the customer who purchased these orders.
5. Display the total no of orders corresponding to each delivery status.
6. For orders purchasing more than 1 item, how many are still not completed?
7. Find the total no of orders corresponding to each delivery status by ignoring the case in delivery status. Status with highest no of orders should be at the top.
8. Write a query to identify the total products purchased by each customer.
9. Display the total sales and average sales done for each day.
10. Display the customer name, employee name and total sale amount of all orders which are either on hold or pending.
11. Fetch all the orders which were neither completed/pending or were handled by the employee Abrar. Display employee name and all details or order.
12. Fetch the orders which cost more than 2000 but did not include the macbook pro. Print the total sale amount as well.
13. Identify the customers who have not purchased any product yet.
14. Write a query to identify the total products purchased by each customer. Return all customers irrespective of wether they have made a purchase or not. Sort the result with highest no of orders at the top.
15. Corresponding to each employee, display the total sales they made of all the completed orders. Display total sales as 0 if an employee made no sales yet.
16. Re-write the above query so as to display the total sales made by each employee corresponding to each customer. If an employee has not served a customer yet then display "-" under the customer.
17. Re-write above query so as to display only those records where the total sales is above 1000.
18. Identify employees who have served more than 2 customer.
19. Identify the customers who have purchased more than 5 products.
20. Identify customers whose average purchase cost exceeds the average sale of all the orders

select \* from customers

select \* from employees

select \* from products

select \* from sales\_order

--- 1. Identify the total no of products sold

select sum(quantity) as total\_sold\_products from sales\_order;

-- 2. Other than Completed, display the available delivery status's.

SELECT status

FROM sales\_order

WHERE status NOT IN ('Completed', 'completed');

SELECT status

FROM sales\_order

WHERE LOWER(status) <> 'completed';

-- 3. Display the order id, order\_date and product\_name for all the completed orders.

SELECT order\_id, order\_date, name

FROM sales\_order so

INNER JOIN products p ON so.prod\_id = p.id

WHERE LOWER(status) = 'completed'

-- 4. Sort the above query to show the earliest orders at the top. Also display the customer who purchased these orders.

SELECT order\_id, order\_date, p.name AS product\_name, c.name AS customer\_name

FROM sales\_order so

INNER JOIN products p ON so.prod\_id = p.id

INNER JOIN customers c ON so.customer\_id = c.id

WHERE LOWER(status) = 'completed'

ORDER BY order\_date

-- 5. Display the total no of orders corresponding to each delivery status

SELECT status, COUNT(\*) AS total\_order

FROM sales\_order

GROUP BY status

-- 6. For orders purchasing more than 1 item, how many are still not completed?

SELECT COUNT(\*)

FROM sales\_order

WHERE quantity > 1

AND LOWER(status) <> 'completed'

--7. Find the total no of orders corresponding to each delivery status by ignoring the case in delivery status.Status with highest no of orders should be at the top.

SELECT LOWER(status) as status, COUNT(\*) AS total\_order

FROM sales\_order

GROUP BY LOWER(status)

ORDER BY total\_order DESC

SELECT updated\_status, COUNT(\*) AS total\_order

FROM (SELECT status, CASE

WHEN status = 'completed' THEN

'Completed'

ELSE

status

END AS updated\_status

FROM sales\_order)

GROUP BY updated\_status

ORDER BY total\_order DESC

--8. Write a query to identify the total products purchased by each customer

SELECT c.name as customer\_name, SUM(so.quantity) AS total\_purchase\_product

FROM sales\_order so

INNER JOIN customers c ON so.customer\_id = c.id

GROUP BY c.name

--9. Display the total sales and average sales done for each day.

SELECT order\_date, SUM(quantity \* price), AVG(quantity \*price)

FROM sales\_order so

JOIN products p ON p.id = so.prod\_id

GROUP BY order\_date

ORDER BY order\_date

--10. Display the customer name, employee name and total sale amount of all orders which are either on hold or pending

SELECT c.name as customer\_name, e.name as employee\_name, SUM(so.quantity \* p.price), so.status

FROM sales\_order so

JOIN customers c ON so.customer\_id = c.id

JOIN employees e ON so.emp\_id = e.id

JOIN products p ON so.prod\_id = p.id

WHERE status IN ('Pending', 'On Hold')

GROUP BY c.name, e.name

--11) Fetch all the orders which were neither completed/pending or were handled by the employee Abrar. Display employee name and all details of order.

SELECT e.name as employee\_name, so.\*

FROM sales\_order so

JOIN employees e ON e.id = so.emp\_id

WHERE LOWER(status) NOT IN ('completed', 'pending')

OR LOWER(e.name) LIKE '%abrar%'

--12. Fetch the orders which cost more than 2000 but did not include the macbook pro. Print the total sale amount as well.

SELECT so.\*, p.name, (quantity \* price) as total\_Sale

FROM sales\_order so

JOIN products p ON so.prod\_id = p.id

WHERE quantity \* price > '2000'

AND LOWER(p.name) NOT LIKE '%macbook%'

--13. Identify the customers who have not purchased any product yet

SELECT \* from customers

WHERE id NOT IN (SELECT DISTINCT customer\_id

FROM sales\_order)

SELECT c.\*

FROM customers c

LEFT JOIN sales\_order so ON so.customer\_id = c.id

WHERE order\_id ISNULL

--14. Write a query to identify the total products purchased by each customer.Return all customers irrespective of wether they have made a purchase or not. Sort the result with highest no of orders at the top.

SELECT c.name, COALESCE(SUM(quantity), 0) AS total\_products\_purchased

FROM sales\_order so

RIGHT JOIN customers c ON so.customer\_id = c.id

GROUP BY c.name

ORDER BY total\_products\_purchased DESC

--15. Corresponding to each employee, display the total sales they made of all the completed orders. Display total sales as 0 if an employee made no sales yet.

SELECT e.name, COALESCE(SUM(quantity \* price),0) AS total\_sales

FROM sales\_order so

JOIN products p ON so.prod\_id = p.id

RIGHT JOIN employees e ON so.emp\_id = e.id

AND LOWER(status) = 'completed'

GROUP BY e.name

--16. Rewrite the above query so as to display the total sales made by each employee corresponding to each customer. If an employee has not served a customer yet then display "-" under the customer.

SELECT e.name, COALESCE(c.name, '-'), COALESCE(SUM(quantity \* price),0) AS total\_sales

FROM sales\_order so

JOIN products p ON so.prod\_id = p.id

JOIN customers c ON so.customer\_id = c.id

RIGHT JOIN employees e ON so.emp\_id = e.id

AND LOWER(status) = 'completed'

GROUP BY e.name, c.name

ORDER BY 1,2

--17. Re-write above query so as to display only those records where the total sales is above 1000

SELECT e.name, COALESCE(c.name, '-'), COALESCE(SUM(quantity \* price),0) AS total\_sales

FROM sales\_order so

JOIN products p ON so.prod\_id = p.id

JOIN customers c ON so.customer\_id = c.id

RIGHT JOIN employees e ON so.emp\_id = e.id

AND LOWER(status) = 'completed'

GROUP BY e.name, c.name

HAVING COALESCE(SUM(quantity \* price),0) > '1000'

ORDER BY 1,2

--18. Identify employees who have served more than 2 customer.

SELECT e.name as employee, COUNT(DISTINCT c.name) as customer

FROM sales\_order so

JOIN customers c ON so.customer\_id = c.id

JOIN employees e ON so.emp\_id = e.id

GROUP BY e.name

HAVING COUNT(DISTINCT c.name) > 2

ORDER BY 1

--19. Identify the customers who have purchased more than 5 products

SELECT c.name AS customer, SUM(quantity) as total\_products

FROM sales\_order so

JOIN customers c ON so.customer\_id = c.id

GROUP BY c.name

HAVING SUM(quantity) > 5

--20. Identify customers whose average purchase cost exceeds the average sale of all the orders

SELECT c.name, AVG(quantity \* price)

FROM sales\_order so

JOIN products p ON so.prod\_id = p.id

JOIN customers c ON so.customer\_id = c.id

GROUP BY c.name

HAVING AVG(quantity \* price) > (SELECT AVG(quantity \* price)

FROM sales\_order so

JOIN products p ON so.prod\_id = p.id)