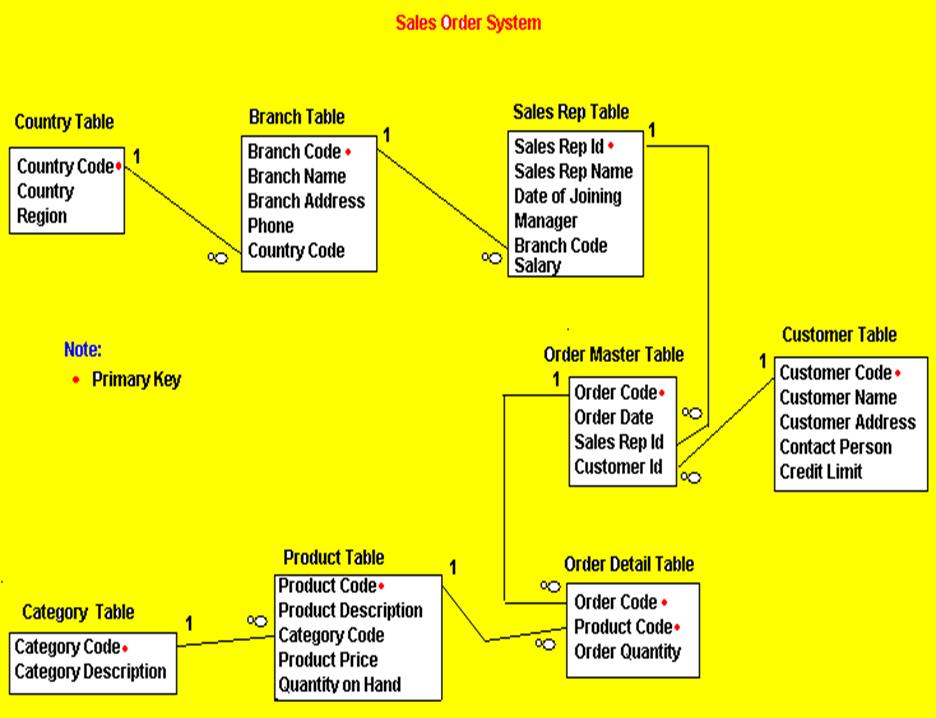
Sales Order Management



USE salesordermanagement;

-- 1. Which category has maximum products?

-- Method 1: Using RANK function

SELECT category\_code , product\_count

FROM (

SELECT c.category\_code , COUNT(p.product\_code) AS product\_count,

RANK() OVER(ORDER BY COUNT(p.product\_code) DESC) AS rnk

FROM category AS c

JOIN product AS p

ON c.category\_code = p.category\_code

GROUP BY category\_code

) AS ranked\_products

WHERE rnk = 1;

-- Method 2: Using NOT IN

SELECT t.category\_code, t.product\_count

FROM (

SELECT c.category\_code, COUNT(p.product\_code) AS product\_count

FROM category AS c

LEFT JOIN product AS p

ON p.category\_code = c.category\_code

GROUP BY c.category\_code

) AS t

WHERE t.product\_count NOT IN (

SELECT t2.product\_count

FROM (

SELECT c2.category\_code, COUNT(p2.product\_code) AS product\_count

FROM category AS c2

LEFT JOIN product AS p2

ON p2.category\_code = c2.category\_code

GROUP BY c2.category\_code

) AS t2

WHERE t2.product\_count > t.product\_count -- any bigger count

);

-- Method 3: Using HAVING

SELECT c.category\_code, COUNT(p.product\_code) AS product\_count

FROM category AS c

JOIN product AS p

ON p.category\_code = c.category\_code

GROUP BY c.category\_code

HAVING COUNT(p.product\_code) = (

SELECT MAX(cnt)

FROM (

SELECT COUNT(p2.product\_code) AS cnt

FROM category AS c2

JOIN product AS p2

ON p2.category\_code = c2.category\_code

GROUP BY c2.category\_code

) AS counts

);

-- Method 4: Using JOIN

SELECT cnt.category\_code, cnt.product\_count

FROM (

SELECT c.category\_code, COUNT(p.product\_code) AS product\_count

FROM category AS c

LEFT JOIN product AS p

ON p.category\_code = c.category\_code

GROUP BY c.category\_code

) AS cnt

JOIN (

SELECT MAX(cnt2) AS max\_cnt

FROM (

SELECT COUNT(p2.product\_code) AS cnt2

FROM category AS c2

LEFT JOIN product AS p2

ON p2.category\_code = c2.category\_code

GROUP BY c2.category\_code

) AS lvl2

) AS m

ON cnt.product\_count = m.max\_cnt;

-- Method 5: Using NOT EXISTS (Correlated Subquery)

SELECT c.category\_code,(

SELECT COUNT(p.product\_code)

FROM product AS p

WHERE p.category\_code = c.category\_code

) AS product\_count

FROM category AS c

WHERE NOT EXISTS (

SELECT 1

FROM category AS c2

WHERE (SELECT COUNT(p2.product\_code)

FROM product AS p2

WHERE p2.category\_code = c2.category\_code) > (

SELECT COUNT(\*)

FROM product AS p3

WHERE p3.category\_code = c.category\_code

)

);

-- Method 6: Using UNION (SET operators)

-- First approach (direct max)

SELECT c.category\_code, COUNT(p.product\_code) AS product\_count

FROM category AS c

LEFT JOIN product AS p

ON p.category\_code = c.category\_code

GROUP BY c.category\_code

HAVING COUNT(p.product\_code) = (

SELECT MAX(pcount) FROM (

SELECT COUNT(p2.product\_code) AS pcount

FROM category AS c2

LEFT JOIN product AS p2

ON p2.category\_code = c2.category\_code

GROUP BY c2.category\_code

) AS max\_counts

)

UNION

-- Second approach (for handling ties)

SELECT c.category\_code, COUNT(p.product\_code) AS product\_count

FROM category AS c

LEFT JOIN product AS p ON p.category\_code = c.category\_code

GROUP BY c.category\_code

HAVING COUNT(p.product\_code) = (

SELECT MAX(pcount) FROM (

SELECT COUNT(p2.product\_code) AS pcount

FROM category AS c2

LEFT JOIN product AS p2

ON p2.category\_code = c2.category\_code

GROUP BY c2.category\_code

) AS max\_counts

);

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT \*

FROM (

SELECT c.category\_code, COUNT(p.product\_code) AS product\_count

FROM category AS c

LEFT JOIN product AS p

ON p.category\_code = c.category\_code

GROUP BY c.category\_code

) AS counts

EXCEPT

SELECT dominated.\*

FROM (

SELECT c1.category\_code, COUNT(p1.product\_code) AS product\_count

FROM category AS c1

LEFT JOIN product AS p1

ON p1.category\_code = c1.category\_code

GROUP BY c1.category\_code

) AS dominated

JOIN (

SELECT c2.category\_code, COUNT(p2.product\_code) AS product\_count

FROM category AS c2

LEFT JOIN product AS p2

ON p2.category\_code = c2.category\_code

GROUP BY c2.category\_code

) AS dominator

ON dominator.product\_count > dominated.product\_count;

-- Method 8: Using INTERSECT (SET operators)

SELECT c.category\_code, COUNT(p.product\_code) AS product\_count

FROM category AS c

LEFT JOIN product AS p

ON p.category\_code = c.category\_code

GROUP BY c.category\_code

INTERSECT

SELECT c.category\_code, COUNT(p.product\_code) AS product\_count

FROM category AS c

LEFT JOIN product AS p

ON p.category\_code = c.category\_code

GROUP BY c.category\_code

HAVING COUNT(p.product\_code) = (

SELECT MAX(pcount) FROM (

SELECT COUNT(p2.product\_code) AS pcount

FROM category AS c2

LEFT JOIN product AS p2

ON p2.category\_code = c2.category\_code

GROUP BY c2.category\_code

) AS max\_counts

);

\*/

-- 2. Which category has minimum products?

-- Method 1: Using RANK function

SELECT category\_code , product\_count

FROM (

SELECT c.category\_code , COUNT(p.product\_code) AS product\_count,

RANK() OVER(ORDER BY COUNT(p.product\_code) ASC) AS rnk

FROM category AS c

JOIN product AS p

ON c.category\_code = p.category\_code

GROUP BY category\_code

) AS ranked\_products

WHERE rnk = 1;

-- Method 2: Using NOT IN

SELECT c.category\_code, COUNT(p.product\_code) AS product\_count

FROM category AS c

LEFT JOIN product AS p

ON c.category\_code = p.category\_code

GROUP BY c.category\_code

HAVING COUNT(p.product\_code) NOT IN (

-- All counts that are NOT the minimum count

SELECT cnt FROM (

SELECT COUNT(p2.product\_code) AS cnt

FROM category AS c2

LEFT JOIN product AS p2

ON c2.category\_code = p2.category\_code

GROUP BY c2.category\_code

) all\_counts

WHERE cnt > (

SELECT MIN(cnt) FROM (

SELECT COUNT(p3.product\_code) AS cnt

FROM category AS c3

LEFT JOIN product AS p3

ON c3.category\_code = p3.category\_code

GROUP BY c3.category\_code

) min\_counts

)

);

-- Method 3: Using HAVING

SELECT c.category\_code, COUNT(p.product\_code) AS product\_count

FROM category AS c

JOIN product AS p

ON p.category\_code = c.category\_code

GROUP BY c.category\_code

HAVING COUNT(p.product\_code) = (

SELECT MIN(inner\_cnt)

FROM (

SELECT COUNT(p2.product\_code) AS inner\_cnt

FROM category AS c2

JOIN product AS p2

ON p2.category\_code = c2.category\_code

GROUP BY c2.category\_code

) AS z

);

-- Method 4: Using JOIN

/\* Step 1: build counts; Step 2: join to a onerow table holding the MIN count \*/

SELECT cnt.category\_code, cnt.product\_count

FROM (

SELECT c.category\_code, COUNT(p.product\_code) AS product\_count

FROM category AS c

LEFT JOIN product AS p

ON p.category\_code = c.category\_code

GROUP BY c.category\_code

) AS cnt

JOIN (

SELECT MIN(cnt2) AS min\_cnt

FROM (

SELECT COUNT(p2.product\_code) AS cnt2

FROM category AS c2

LEFT JOIN product AS p2

ON p2.category\_code = c2.category\_code

GROUP BY c2.category\_code

) AS lvl2

) AS m

ON cnt.product\_count = m.min\_cnt;

-- Method 5: Using NOT EXISTS (Correlated Subquery)

SELECT c.category\_code, (

SELECT COUNT(\*)

FROM product p

WHERE p.category\_code = c.category\_code

) AS product\_count

FROM category c

WHERE NOT EXISTS (

/\* if a category with a strictly smaller count exists, exclude this one \*/

SELECT 1

FROM category c2

WHERE (

SELECT COUNT(\*)

FROM product p2

WHERE p2.category\_code = c2.category\_code

) < (

SELECT COUNT(\*)

FROM product p3

WHERE p3.category\_code = c.category\_code

)

);

-- Method 6: Using UNION ALL (SET operators)

-- First get categories with zero products (if any exist)

SELECT c.category\_code, 0 AS product\_count

FROM category c

WHERE NOT EXISTS (

SELECT 1 FROM product p

WHERE p.category\_code = c.category\_code

)

UNION ALL

-- Then get categories with non-zero but minimum products

SELECT c.category\_code, COUNT(p.product\_code) AS product\_count

FROM category AS c

JOIN product AS p

ON c.category\_code = p.category\_code

GROUP BY c.category\_code

HAVING COUNT(p.product\_code) = (

SELECT MIN(cnt)

FROM (

SELECT COUNT(p2.product\_code) AS cnt

FROM category AS c2

JOIN product AS p2

ON c2.category\_code = p2.category\_code

GROUP BY c2.category\_code

) min\_counts

)

AND NOT EXISTS (

SELECT 1

FROM category AS c3

WHERE NOT EXISTS (

SELECT 1

FROM product AS p3

WHERE p3.category\_code = c3.category\_code

)

);

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

-- Get all category counts

SELECT c.category\_code, COUNT(p.product\_code) AS product\_count

FROM category c

LEFT JOIN product p

ON c.category\_code = p.category\_code

GROUP BY c.category\_code

EXCEPT

-- Remove categories that have counts larger than some other category

SELECT c1.category\_code, COUNT(p1.product\_code)

FROM category c1

LEFT JOIN product p1

ON c1.category\_code = p1.category\_code

GROUP BY c1.category\_code

WHERE COUNT(p1.product\_code) > (

SELECT MIN(cnt) FROM (

SELECT COUNT(p2.product\_code) AS cnt

FROM category c2

LEFT JOIN product p2

ON c2.category\_code = p2.category\_code

GROUP BY c2.category\_code

) min\_counts

);

-- Method 8: Using INTERSECT (SET operators)

-- Get all category counts

SELECT c.category\_code, COUNT(p.product\_code) AS product\_count

FROM category c

LEFT JOIN product p

ON c.category\_code = p.category\_code

GROUP BY c.category\_code

INTERSECT

-- Keep only those with the minimum count

SELECT c.category\_code, COUNT(p.product\_code)

FROM category c

LEFT JOIN product p ON c.category\_code = p.category\_code

GROUP BY c.category\_code

HAVING COUNT(p.product\_code) = (

SELECT MIN(cnt) FROM (

SELECT COUNT(p2.product\_code) AS cnt

FROM category c2

LEFT JOIN product p2

ON c2.category\_code = p2.category\_code

GROUP BY c2.category\_code

) min\_counts

);

\*/

-- 3. Which category has no products?

-- Method 1: Using JOIN

SELECT c.category\_code

FROM category AS c

LEFT JOIN product AS p

ON c.category\_code = p.category\_code

WHERE p.category\_code IS NULL;

-- Method 2: Using NOT IN

SELECT category\_code

FROM category

WHERE category\_code NOT IN (

SELECT DISTINCT category\_code

FROM product

);

-- Method 3: Using HAVING

SELECT c.category\_code

FROM category AS c

LEFT JOIN product AS p

ON c.category\_code = p.category\_code

GROUP BY c.category\_code

HAVING COUNT(p.product\_code) = 0;

-- Method 4: Using NOT EXISTS (Correlated Subquery)

SELECT c.category\_code

FROM category AS c

WHERE NOT EXISTS (

SELECT 1

FROM product p

WHERE p.category\_code = c.category\_code

);

-- Method 5: Using UNION ALL (SET operators)

-- Explicitly find categories with zero products

SELECT c.category\_code

FROM category AS c

LEFT JOIN product AS p

ON c.category\_code = p.category\_code

GROUP BY c.category\_code

HAVING COUNT(p.product\_code) = 0

UNION ALL

-- Include categories that don't appear in product table at all

SELECT c.category\_code

FROM category AS c

WHERE c.category\_code NOT IN (

SELECT DISTINCT category\_code

FROM product

);

4. Which is the costliest product?

Using NOT IN

SELECT product\_code, prod\_desc, price

FROM product p1

WHERE price NOT IN (

SELECT price

FROM product p2

WHERE p2.price > p1.price

);

Using a plain subquery

SELECT product\_code, prod\_desc, price

FROM product

WHERE price = (

SELECT MAX(price)

FROM product

);

Using JOIN

SELECT p.product\_code, p.prod\_desc, p.price

FROM product p

JOIN (

SELECT MAX(price) AS max\_price

FROM product

) AS maxval

ON p.price = maxval.max\_price;

Using a correlated subquery

SELECT p1.product\_code, p1.prod\_desc, p1.price

FROM product p1

WHERE NOT EXISTS (

SELECT 1

FROM product p2

WHERE p2.price > p1.price

);

Using set operators

SELECT product\_code, prod\_desc, price

FROM product

EXCEPT

SELECT p1.product\_code, p1.prod\_desc, p1.price

FROM product p1

JOIN product p2

ON p2.price > p1.price;

5. Which category has costliest product?

Using RANK function

SELECT category\_code, price

FROM (

SELECT c.category\_code , p.price ,

RANK() OVER(ORDER BY (p.price) DESC) AS rnk

FROM category c

JOIN product p

ON c.category\_code = p.category\_code

)AS rnked\_price

WHERE rnk = 1;

Using NOT IN

SELECT c.category\_code, p.price

FROM category c

JOIN product p

ON c.category\_code = p.category\_code

WHERE p.price NOT IN (

SELECT price

FROM product

WHERE price > p.price

);

Using a plain subquery

SELECT c.category\_code, p.price

FROM category c

JOIN product p

ON c.category\_code = p.category\_code

WHERE p.price = (

SELECT MAX(price)

FROM product

);

Using JOIN

SELECT c.category\_code, p.price

FROM category c

JOIN product p

ON c.category\_code = p.category\_code

JOIN (

SELECT MAX(price) AS max\_price

FROM product

) AS maxval

ON p.price = maxval.max\_price;

Using a correlated subquery

SELECT c.category\_code, p.price

FROM category c

JOIN product p

ON c.category\_code = p.category\_code

WHERE NOT EXISTS (

SELECT 1

FROM product p2

WHERE p2.price > p.price

);

Using set operators

SELECT c.category\_code, p.price

FROM category c

JOIN product p

ON c.category\_code = p.category\_code

EXCEPT

SELECT c1.category\_code, p1.price

FROM category c1

JOIN product p1

ON c1.category\_code = p1.category\_code

JOIN product p2

ON p2.price > p1.price;

6. Which category has lot of products (with respect to quantity on hand)?

Using RANK function

SELECT category\_code, total\_qty

FROM (

SELECT category\_code, SUM(qty\_on\_hand) AS total\_qty,

RANK() OVER (ORDER BY SUM(qty\_on\_hand) DESC) AS rnk

FROM product

GROUP BY category\_code

) ranked

WHERE rnk = 1;

Using NOT IN

SELECT t.category\_code, t.total\_qty

FROM (

SELECT category\_code, SUM(qty\_on\_hand) AS total\_qty

FROM product

GROUP BY category\_code

) AS t

WHERE t.total\_qty NOT IN (

SELECT u.total\_qty

FROM (

SELECT category\_code, SUM(qty\_on\_hand) AS total\_qty

FROM product

GROUP BY category\_code

) AS u

WHERE u.total\_qty > t.total\_qty

);

Using a plain subquery

SELECT category\_code, SUM(qty\_on\_hand) AS total\_qty

FROM product

GROUP BY category\_code

HAVING SUM(qty\_on\_hand) = (

SELECT MAX(total\_qty)

FROM (

SELECT SUM(qty\_on\_hand) AS total\_qty

FROM product

GROUP BY category\_code

) AS v

);

Using JOIN

SELECT agg.category\_code, agg.total\_qty

FROM (

SELECT category\_code, SUM(qty\_on\_hand) AS total\_qty

FROM product

GROUP BY category\_code

) AS agg

JOIN (

SELECT MAX(z.total\_qty) AS max\_qty

FROM (

SELECT SUM(qty\_on\_hand) AS total\_qty

FROM product

GROUP BY category\_code

) AS z

) AS m

ON agg.total\_qty = m.max\_qty;

Using a correlated subquery

SELECT a.category\_code, a.total\_qty

FROM (

SELECT category\_code, SUM(qty\_on\_hand) AS total\_qty

FROM product

GROUP BY category\_code

) AS a

WHERE NOT EXISTS (

SELECT 1

FROM (

SELECT category\_code, SUM(qty\_on\_hand) AS total\_qty

FROM product

GROUP BY category\_code

) AS b

WHERE b.total\_qty > a.total\_qty

);

Using set operators

SELECT \*

FROM (

SELECT category\_code, SUM(qty\_on\_hand) AS total\_qty

FROM product

GROUP BY category\_code

) AS counts

EXCEPT use MINUS in Oracle

SELECT dom.\*

FROM (

SELECT category\_code, SUM(qty\_on\_hand) AS total\_qty

FROM product

GROUP BY category\_code

) AS dom

JOIN (

SELECT category\_code, SUM(qty\_on\_hand) AS total\_qty

FROM product

GROUP BY category\_code

) AS win

ON win.total\_qty > dom.total\_qty;

7. Category wise display the costliest products?

Using RANK function

SELECT category\_code, product\_code, prod\_desc, price

FROM (

SELECT category\_code, product\_code, prod\_desc, price,

RANK() OVER (PARTITION BY category\_code ORDER BY price DESC) as rnk

FROM product

) ranked

WHERE rnk = 1;

Using NOT IN

SELECT p1.category\_code, p1.product\_code, p1.prod\_desc, p1.price

FROM product p1

WHERE p1.price NOT IN (

SELECT p2.price

FROM product p2

WHERE p2.category\_code = p1.category\_code AND p2.price > p1.price

);

Using a plain subquery

SELECT p.\*

FROM product p

WHERE p.price = (

SELECT MAX(price)

FROM product q

WHERE q.category\_code = p.category\_code

);

Using JOIN

SELECT p.category\_code, p.product\_code, p.prod\_desc, p.price

FROM product p

JOIN (

SELECT category\_code, MAX(price) AS max\_price

FROM product

GROUP BY category\_code

) AS mx

ON p.category\_code = mx.category\_code AND p.price = mx.max\_price;

Using a correlated subquery

SELECT p1.category\_code, p1.product\_code, p1.prod\_desc, p1.price

FROM product p1

WHERE NOT EXISTS (

SELECT 1

FROM product p2

WHERE p2.category\_code = p1.category\_code AND p2.price > p1.price

);

Using set operators

SELECT category\_code, product\_code, prod\_desc, price

FROM product

EXCEPT replace with MINUS in Oracle

SELECT p1.category\_code, p1.product\_code, p1.prod\_desc, p1.price

FROM product p1

JOIN product p2

ON p2.category\_code = p1.category\_code AND p2.price > p1.price;

8. Category wise display the product whose quantity on hand is minimum?

Using RANK function

SELECT category\_code, product\_code, prod\_desc, qty\_on\_hand

FROM (

SELECT category\_code, product\_code, prod\_desc, qty\_on\_hand,

RANK() OVER (PARTITION BY category\_code ORDER BY qty\_on\_hand ASC) AS rnk

FROM product

) ranked

WHERE rnk = 1;

SELECT x.\*

FROM product x

WHERE qty\_on\_hand = (

SELECT MIN(qty\_on\_hand)

FROM product y

WHERE y.category\_code = x.category\_code

);

Using NOT IN

SELECT p1.category\_code, p1.product\_code, p1.prod\_desc, p1.qty\_on\_hand

FROM product p1

WHERE p1.qty\_on\_hand NOT IN (

SELECT p2.qty\_on\_hand

FROM product p2

WHERE p2.category\_code = p1.category\_code AND p2.qty\_on\_hand < p1.qty\_on\_hand

);

Using a plain subquery

SELECT p.\*

FROM product p

WHERE p.qty\_on\_hand = (

SELECT MIN(qty\_on\_hand)

FROM product q

WHERE q.category\_code = p.category\_code

);

Using JOIN

SELECT p.category\_code, p.product\_code, p.prod\_desc, p.qty\_on\_hand

FROM product p

JOIN (

SELECT category\_code, MIN(qty\_on\_hand) AS min\_qty

FROM product

GROUP BY category\_code

) AS m

ON p.category\_code = m.category\_code AND p.qty\_on\_hand = m.min\_qty;

Using a correlated subquery

SELECT p1.category\_code, p1.product\_code, p1.prod\_desc, p1.qty\_on\_hand

FROM product p1

WHERE NOT EXISTS (

SELECT 1

FROM product p2

WHERE p2.category\_code = p1.category\_code AND p2.qty\_on\_hand < p1.qty\_on\_hand

);

Using set operators

SELECT category\_code, product\_code, prod\_desc, qty\_on\_hand

FROM product

EXCEPT

SELECT p1.category\_code, p1.product\_code, p1.prod\_desc, p1.qty\_on\_hand

FROM product p1

JOIN product p2

ON p2.category\_code = p1.category\_code AND p2.qty\_on\_hand < p1.qty\_on\_hand;

9. Which order has maximum products?

Using RANK function

SELECT order\_code, total\_products

FROM (

SELECT order\_code, COUNT(DISTINCT product\_code) AS total\_products,

RANK() OVER (ORDER BY COUNT(DISTINCT product\_code) DESC) AS rnk

FROM order\_detail

GROUP BY order\_code

) ranked

WHERE rnk = 1;

Using NOT IN

SELECT a.order\_code, a.total\_products

FROM (

SELECT order\_code, COUNT(DISTINCT product\_code) AS total\_products

FROM order\_detail

GROUP BY order\_code

) AS a

WHERE a.total\_products NOT IN (

SELECT b.total\_products

FROM (

SELECT order\_code, COUNT(DISTINCT product\_code) AS total\_products

FROM order\_detail

GROUP BY order\_code

) AS b

WHERE b.total\_products > a.total\_products

);

Using a plain subquery

SELECT order\_code, COUNT(DISTINCT product\_code) AS total\_products

FROM order\_detail

GROUP BY order\_code

HAVING COUNT(DISTINCT product\_code) = (

SELECT MAX(cnt)

FROM (

SELECT COUNT(DISTINCT product\_code) AS cnt

FROM order\_detail

GROUP BY order\_code

) AS x

);

Using JOIN

SELECT o.order\_code, o.total\_products

FROM (

SELECT order\_code, COUNT(DISTINCT product\_code) AS total\_products

FROM order\_detail

GROUP BY order\_code

) AS o

JOIN (

SELECT MAX(tp) AS max\_tp

FROM (

SELECT COUNT(DISTINCT product\_code) AS tp

FROM order\_detail

GROUP BY order\_code

) AS y

) AS m

ON o.total\_products = m.max\_tp;

Using a correlated subquery

SELECT a.order\_code, a.total\_products

FROM (

SELECT order\_code, COUNT(DISTINCT product\_code) AS total\_products

FROM order\_detail

GROUP BY order\_code

) AS a

WHERE NOT EXISTS (

SELECT 1

FROM (

SELECT order\_code, COUNT(DISTINCT product\_code) AS total\_products

FROM order\_detail

GROUP BY order\_code

) AS b

WHERE b.total\_products > a.total\_products

);

Using set operators

SELECT order\_code, total\_products

FROM (

SELECT order\_code, COUNT(DISTINCT product\_code) AS total\_products

FROM order\_detail

GROUP BY order\_code

) AS s

EXCEPT

SELECT x.order\_code, x.total\_products

FROM (

SELECT order\_code, COUNT(DISTINCT product\_code) AS total\_products

FROM order\_detail

GROUP BY order\_code

) AS x

JOIN (

SELECT order\_code, COUNT(DISTINCT product\_code) AS total\_products

FROM order\_detail

GROUP BY order\_code

) AS y

ON y.total\_products > x.total\_products;

10. Which is the frequently ordered product?

Using RANK function

SELECT product\_code, total\_orders

FROM (

SELECT product\_code, COUNT(order\_code) AS total\_orders,

RANK() OVER (ORDER BY COUNT(order\_code) DESC) AS rnk

FROM order\_detail

GROUP BY product\_code

) ranked

WHERE rnk = 1;

Using NOT IN

SELECT a.product\_code, a.total\_orders

FROM (

SELECT product\_code, COUNT(order\_code) AS total\_orders

FROM order\_detail

GROUP BY product\_code ) AS a

WHERE a.total\_orders NOT IN (

SELECT b.total\_orders

FROM (

SELECT product\_code, COUNT(order\_code) AS total\_orders

FROM order\_detail

GROUP BY product\_code ) AS b

WHERE b.total\_orders > a.total\_orders

);

Using a plain subquery

SELECT product\_code, COUNT(order\_code) AS total\_orders

FROM order\_detail

GROUP BY product\_code

HAVING COUNT(order\_code) = (

SELECT MAX(cnt)

FROM (

SELECT COUNT(order\_code) AS cnt

FROM order\_detail

GROUP BY product\_code

) AS t

);

Using JOIN

SELECT d.product\_code, d.total\_orders

FROM (

SELECT product\_code, COUNT(order\_code) AS total\_orders

FROM order\_detail

GROUP BY product\_code

) AS d

JOIN (

SELECT MAX(cn) AS max\_orders

FROM (

SELECT COUNT(order\_code) AS cn

FROM order\_detail

GROUP BY product\_code

) AS z

) AS m

ON d.total\_orders = m.max\_orders;

Using a correlated subquery

SELECT d1.product\_code, d1.total\_orders

FROM (

SELECT product\_code, COUNT(order\_code) AS total\_orders

FROM order\_detail

GROUP BY product\_code

) AS d1

WHERE NOT EXISTS (

SELECT 1

FROM (

SELECT product\_code, COUNT(order\_code) AS total\_orders

FROM order\_detail

GROUP BY product\_code

) AS d2

WHERE d2.total\_orders > d1.total\_orders

);

Using set operators

SELECT product\_code, total\_orders

FROM (

SELECT product\_code, COUNT(order\_code) AS total\_orders

FROM order\_detail

GROUP BY product\_code

) AS a

EXCEPT

SELECT b.product\_code, b.total\_orders

FROM (

SELECT product\_code, COUNT(order\_code) AS total\_orders

FROM order\_detail

GROUP BY product\_code

) AS b

JOIN (

SELECT product\_code, COUNT(order\_code) AS total\_orders

FROM order\_detail

GROUP BY product\_code

) AS c

ON c.total\_orders > b.total\_orders;

11. Which product is least ordered product?

Using RANK function

SELECT product\_code, total\_orders

FROM (

SELECT product\_code, COUNT(order\_code) AS total\_orders,

RANK() OVER (ORDER BY COUNT(order\_code) ASC) AS rnk

FROM order\_detail

GROUP BY product\_code

) ranked

WHERE rnk = 1;

Using NOT IN

SELECT a.product\_code, a.total\_orders

FROM (

SELECT product\_code, COUNT(order\_code) AS total\_orders

FROM order\_detail

GROUP BY product\_code

) AS a

WHERE a.total\_orders NOT IN (

SELECT b.total\_orders

FROM (

SELECT product\_code, COUNT(order\_code) AS total\_orders

FROM order\_detail

GROUP BY product\_code

) AS b

WHERE b.total\_orders < a.total\_orders

);

Using a plain subquery

SELECT product\_code, COUNT(order\_code) AS total\_orders

FROM order\_detail

GROUP BY product\_code

HAVING COUNT(order\_code) = (

SELECT MIN(cnt) FROM (

SELECT COUNT(order\_code) AS cnt

FROM order\_detail

GROUP BY product\_code

) AS t

);

Using JOIN

SELECT d.product\_code, d.total\_orders

FROM (

SELECT product\_code, COUNT(order\_code) AS total\_orders

FROM order\_detail

GROUP BY product\_code ) AS d

JOIN (

SELECT MIN(cn) AS min\_orders

FROM (

SELECT COUNT(order\_code) AS cn

FROM order\_detail

GROUP BY product\_code

) AS z

) AS m

ON d.total\_orders = m.min\_orders;

Using a correlated subquery

SELECT d1.product\_code, d1.total\_orders

FROM (

SELECT product\_code, COUNT(order\_code) AS total\_orders

FROM order\_detail

GROUP BY product\_code

) AS d1

WHERE NOT EXISTS (

SELECT 1

FROM (

SELECT product\_code, COUNT(order\_code) AS total\_orders

FROM order\_detail

GROUP BY product\_code

) AS d2

WHERE d2.total\_orders < d1.total\_orders

);

Using set operators

SELECT product\_code, total\_orders

FROM (

SELECT product\_code, COUNT(order\_code) AS total\_orders

FROM order\_detail

GROUP BY product\_code

) AS a

EXCEPT

SELECT b.product\_code, b.total\_orders

FROM (

SELECT product\_code, COUNT(order\_code) AS total\_orders

FROM order\_detail

GROUP BY product\_code

) AS b

JOIN (

SELECT product\_code, COUNT(order\_code) AS total\_orders

FROM order\_detail

GROUP BY product\_code

) AS c

ON c.total\_orders < b.total\_orders;

12. What product is not at all ordered?

Using NOT IN

SELECT product\_code, prod\_desc

FROM product

WHERE product\_code NOT IN (

SELECT DISTINCT product\_code

FROM order\_detail

);

Using a plain subquery

SELECT p.product\_code, p.prod\_desc

FROM product p

WHERE NOT EXISTS (

SELECT 1

FROM order\_detail od

WHERE od.product\_code = p.product\_code

);

Using Left Join

SELECT product.product\_code, product.prod\_desc

FROM product

LEFT JOIN order\_detail

ON product.product\_code = order\_detail.product\_code

WHERE order\_detail.product\_code IS NULL;

Using set operators

SELECT product\_code, prod\_desc

FROM product

EXCEPT

SELECT DISTINCT p.product\_code, p.prod\_desc

FROM product AS p

JOIN order\_detail AS od

ON p.product\_code = od.product\_code;

Using JOIN

SELECT p.product\_code, p.prod\_desc

FROM product p

LEFT JOIN (

SELECT DISTINCT product\_code

FROM order\_detail

) AS o

ON p.product\_code = o.product\_code

WHERE o.product\_code IS NULL;

Using a correlated subquery

SELECT p.product\_code, p.prod\_desc

FROM product p

WHERE NOT EXISTS (

SELECT 1

FROM order\_detail od

WHERE od.product\_code = p.product\_code

);

13. Which is the costliest order? (Calculate the bill)

Using RANK function

SELECT order\_code, total\_cost

FROM (

SELECT order\_code, SUM(p.price \* o.qty\_ordered) AS total\_cost,

RANK() OVER(ORDER BY SUM(p.price \* o.qty\_ordered) DESC) AS rnk

FROM order\_detail o

JOIN product p ON o.product\_code = p.product\_code

GROUP BY order\_code

) rnked

WHERE rnk = 1;

Using NOT IN

SELECT t.order\_code, t.total\_cost

FROM (

SELECT od.order\_code,

SUM(p.price \* od.qty\_ordered) AS total\_cost

FROM order\_detail od

JOIN product p ON p.product\_code = od.product\_code

GROUP BY od.order\_code

) AS t

WHERE t.total\_cost NOT IN (

SELECT u.total\_cost

FROM (

SELECT od2.order\_code,

SUM(p2.price \* od2.qty\_ordered) AS total\_cost

FROM order\_detail od2

JOIN product p2 ON p2.product\_code = od2.product\_code

GROUP BY od2.order\_code

) AS u

WHERE u.total\_cost > t.total\_cost

);

Using a plain subquery

SELECT od.order\_code,

SUM(p.price \* od.qty\_ordered) AS total\_cost

FROM order\_detail od

JOIN product p ON p.product\_code = od.product\_code

GROUP BY od.order\_code

HAVING SUM(p.price \* od.qty\_ordered) = (

SELECT MAX(tc)

FROM (

SELECT SUM(p2.price \* od2.qty\_ordered) AS tc

FROM order\_detail od2

JOIN product p2 ON p2.product\_code = od2.product\_code

GROUP BY od2.order\_code

) AS x

);

Using JOIN

SELECT c.order\_code, c.total\_cost

FROM (

SELECT od.order\_code,

SUM(p.price \* od.qty\_ordered) AS total\_cost

FROM order\_detail od

JOIN product p ON p.product\_code = od.product\_code

GROUP BY od.order\_code

) AS c

JOIN (

SELECT MAX(t.total\_cost) AS max\_cost

FROM (

SELECT SUM(p2.price \* od2.qty\_ordered) AS total\_cost

FROM order\_detail od2

JOIN product p2 ON p2.product\_code = od2.product\_code

GROUP BY od2.order\_code

) AS t

) AS m

ON c.total\_cost = m.max\_cost;

Using a correlated subquery

SELECT a.order\_code, a.total\_cost

FROM (

SELECT od.order\_code,

SUM(p.price \* od.qty\_ordered) AS total\_cost

FROM order\_detail od

JOIN product p ON p.product\_code = od.product\_code

GROUP BY od.order\_code

) AS a

WHERE NOT EXISTS (

SELECT 1

FROM (

SELECT od2.order\_code,

SUM(p2.price \* od2.qty\_ordered) AS total\_cost

FROM order\_detail od2

JOIN product p2 ON p2.product\_code = od2.product\_code

GROUP BY od2.order\_code

) AS b

WHERE b.total\_cost > a.total\_cost

);

Using set operators

SELECT order\_code, total\_cost

FROM (

SELECT od.order\_code,

SUM(p.price \* od.qty\_ordered) AS total\_cost

FROM order\_detail od

JOIN product p ON p.product\_code = od.product\_code

GROUP BY od.order\_code

) AS s

EXCEPT use MINUS in Oracle

SELECT x.order\_code, x.total\_cost

FROM (

SELECT od1.order\_code,

SUM(p1.price \* od1.qty\_ordered) AS total\_cost

FROM order\_detail od1

JOIN product p1 ON p1.product\_code = od1.product\_code

GROUP BY od1.order\_code

) AS x

JOIN (

SELECT od2.order\_code,

SUM(p2.price \* od2.qty\_ordered) AS total\_cost

FROM order\_detail od2

JOIN product p2 ON p2.product\_code = od2.product\_code

GROUP BY od2.order\_code

) AS y

ON y.total\_cost > x.total\_cost;

14. In which date the costliest order made?

Using RANK function

SELECT order\_code, order\_date, total\_cost

FROM (

SELECT od.order\_code, om.order\_date,

SUM(p.price \* od.qty\_ordered) AS total\_cost,

RANK() OVER (ORDER BY SUM(p.price \* od.qty\_ordered) DESC) AS rnk

FROM order\_detail od

JOIN product p ON p.product\_code = od.product\_code

JOIN order\_master om ON om.order\_code = od.order\_code

GROUP BY od.order\_code, om.order\_date

) AS ranked

WHERE rnk = 1;

Using NOT IN

SELECT t.order\_code, t.order\_date, t.total\_cost

FROM (

SELECT od.order\_code, om.order\_date, SUM(p.price \* od.qty\_ordered) AS total\_cost

FROM order\_detail od

JOIN product p ON p.product\_code = od.product\_code

JOIN order\_master om ON om.order\_code = od.order\_code

GROUP BY od.order\_code, om.order\_date

) AS t

WHERE t.total\_cost NOT IN (

SELECT u.total\_cost

FROM (

SELECT od2.order\_code,

SUM(p2.price \* od2.qty\_ordered) AS total\_cost

FROM order\_detail od2

JOIN product p2 ON p2.product\_code = od2.product\_code

GROUP BY od2.order\_code

) AS u

WHERE u.total\_cost > t.total\_cost

);

Using a plain subquery

SELECT od.order\_code, om.order\_date, SUM(p.price \* od.qty\_ordered) AS total\_cost

FROM order\_detail od

JOIN product p ON p.product\_code = od.product\_code

JOIN order\_master om ON om.order\_code = od.order\_code

GROUP BY od.order\_code, om.order\_date

HAVING SUM(p.price \* od.qty\_ordered) = (

SELECT MAX(tc)

FROM (

SELECT SUM(p2.price \* od2.qty\_ordered) AS tc

FROM order\_detail od2

JOIN product p2 ON p2.product\_code = od2.product\_code

GROUP BY od2.order\_code

) AS x

);

Using JOIN

SELECT a.order\_code,

a.order\_date,

a.total\_cost

FROM (

SELECT od.order\_code,

om.order\_date,

SUM(p.price \* od.qty\_ordered) AS total\_cost

FROM order\_detail od

JOIN product p ON p.product\_code = od.product\_code

JOIN order\_master om ON om.order\_code = od.order\_code

GROUP BY od.order\_code, om.order\_date

) AS a

JOIN (

SELECT MAX(total\_cost) AS max\_cost

FROM (

SELECT SUM(p2.price \* od2.qty\_ordered) AS total\_cost

FROM order\_detail od2

JOIN product p2 ON p2.product\_code = od2.product\_code

GROUP BY od2.order\_code

) AS z

) AS m

ON a.total\_cost = m.max\_cost;

Using a correlated subquery

SELECT a.order\_code,

a.order\_date,

a.total\_cost

FROM (

SELECT od.order\_code,

om.order\_date,

SUM(p.price \* od.qty\_ordered) AS total\_cost

FROM order\_detail od

JOIN product p ON p.product\_code = od.product\_code

JOIN order\_master om ON om.order\_code = od.order\_code

GROUP BY od.order\_code, om.order\_date

) AS a

WHERE NOT EXISTS (

SELECT 1

FROM (

SELECT od2.order\_code,

SUM(p2.price \* od2.qty\_ordered) AS total\_cost

FROM order\_detail od2

JOIN product p2 ON p2.product\_code = od2.product\_code

GROUP BY od2.order\_code

) AS b

WHERE b.total\_cost > a.total\_cost

);

Using set operators

SELECT order\_code,

order\_date,

total\_cost

FROM (

SELECT od.order\_code,

om.order\_date,

SUM(p.price \* od.qty\_ordered) AS total\_cost

FROM order\_detail od

JOIN product p ON p.product\_code = od.product\_code

JOIN order\_master om ON om.order\_code = od.order\_code

GROUP BY od.order\_code, om.order\_date

) AS s

EXCEPT use MINUS in Oracle

SELECT x.order\_code,

x.order\_date,

x.total\_cost

FROM (

SELECT od1.order\_code,

om1.order\_date,

SUM(p1.price \* od1.qty\_ordered) AS total\_cost

FROM order\_detail od1

JOIN product p1 ON p1.product\_code = od1.product\_code

JOIN order\_master om1 ON om1.order\_code = od1.order\_code

GROUP BY od1.order\_code, om1.order\_date

) AS x

JOIN (

SELECT od2.order\_code,

SUM(p2.price \* od2.qty\_ordered) AS total\_cost

FROM order\_detail od2

JOIN product p2 ON p2.product\_code = od2.product\_code

GROUP BY od2.order\_code

) AS y

ON y.total\_cost > x.total\_cost;

15. Which customer made the costliest order?

Using RANK function

SELECT CUSTOMER\_CODE, CUSTOMER\_NAME, TOTAL\_COST

FROM (

SELECT OM.CUSTOMER\_CODE, C.CUSTOMER\_NAME,

SUM(P.PRICE \* O.QTY\_ORDERED) AS TOTAL\_COST,

RANK() OVER(ORDER BY SUM(P.PRICE \* O.QTY\_ORDERED) DESC) AS rnk

FROM ORDER\_DETAIL O

JOIN PRODUCT P ON O.PRODUCT\_CODE = P.PRODUCT\_CODE

JOIN ORDER\_MASTER OM ON O.ORDER\_CODE = OM.ORDER\_CODE

JOIN CUSTOMER C ON OM.CUSTOMER\_CODE = C.CUSTOMER\_CODE

GROUP BY OM.CUSTOMER\_CODE, C.CUSTOMER\_NAME

) rnked

WHERE rnk = 1;

Using NOT IN

SELECT OM.CUSTOMER\_CODE, C.CUSTOMER\_NAME, SUM(P.PRICE \* OD.QTY\_ORDERED) AS TOTAL\_COST

FROM ORDER\_MASTER OM

JOIN CUSTOMER C ON OM.CUSTOMER\_CODE = C.CUSTOMER\_CODE

JOIN ORDER\_DETAIL OD ON OM.ORDER\_CODE = OD.ORDER\_CODE

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

GROUP BY OM.CUSTOMER\_CODE, C.CUSTOMER\_NAME

HAVING SUM(P.PRICE \* OD.QTY\_ORDERED) NOT IN (

SELECT SUM(P.PRICE \* O.QTY\_ORDERED)

FROM ORDER\_DETAIL O

JOIN PRODUCT P ON O.PRODUCT\_CODE = P.PRODUCT\_CODE

GROUP BY O.ORDER\_CODE

HAVING SUM(P.PRICE \* O.QTY\_ORDERED) < (

SELECT MAX(sub.total)

FROM (

SELECT SUM(P.PRICE \* O.QTY\_ORDERED) AS total

FROM ORDER\_DETAIL O

JOIN PRODUCT P ON O.PRODUCT\_CODE = P.PRODUCT\_CODE

GROUP BY O.ORDER\_CODE

) sub

)

);

Using a plain subquery

SELECT OM.CUSTOMER\_CODE, C.CUSTOMER\_NAME, SUM(P.PRICE \* OD.QTY\_ORDERED) AS TOTAL\_COST

FROM ORDER\_MASTER OM

JOIN CUSTOMER C ON OM.CUSTOMER\_CODE = C.CUSTOMER\_CODE

JOIN ORDER\_DETAIL OD ON OM.ORDER\_CODE = OD.ORDER\_CODE

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

GROUP BY OM.CUSTOMER\_CODE, C.CUSTOMER\_NAME

HAVING SUM(P.PRICE \* OD.QTY\_ORDERED) = (

SELECT MAX(total)

FROM (

SELECT SUM(P.PRICE \* OD.QTY\_ORDERED) AS total

FROM ORDER\_DETAIL OD

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

JOIN ORDER\_MASTER OM ON OD.ORDER\_CODE = OM.ORDER\_CODE

GROUP BY OM.CUSTOMER\_CODE

) AS cost

);

Using a correlated subquery

SELECT DISTINCT C.CUSTOMER\_CODE, C.CUSTOMER\_NAME

FROM CUSTOMER C

WHERE (

SELECT SUM(P.PRICE \* OD.QTY\_ORDERED)

FROM ORDER\_MASTER OM

JOIN ORDER\_DETAIL OD ON OM.ORDER\_CODE = OD.ORDER\_CODE

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

WHERE OM.CUSTOMER\_CODE = C.CUSTOMER\_CODE

) = (

SELECT MAX(TOTAL)

FROM (

SELECT OM.CUSTOMER\_CODE, SUM(P.PRICE \* OD.QTY\_ORDERED) AS TOTAL

FROM ORDER\_MASTER OM

JOIN ORDER\_DETAIL OD ON OM.ORDER\_CODE = OD.ORDER\_CODE

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

GROUP BY OM.CUSTOMER\_CODE

) AS subquery

);

Using set operators

SELECT OM.CUSTOMER\_CODE, C.CUSTOMER\_NAME, SUM(P.PRICE \* OD.QTY\_ORDERED) AS TOTAL\_COST

FROM ORDER\_MASTER OM

JOIN CUSTOMER C ON OM.CUSTOMER\_CODE = C.CUSTOMER\_CODE

JOIN ORDER\_DETAIL OD ON OM.ORDER\_CODE = OD.ORDER\_CODE

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

GROUP BY OM.CUSTOMER\_CODE, C.CUSTOMER\_NAME

INTERSECT

SELECT OM.CUSTOMER\_CODE, C.CUSTOMER\_NAME, MAX(TOTAL\_COST)

FROM (

SELECT OM.CUSTOMER\_CODE, SUM(P.PRICE \* OD.QTY\_ORDERED) AS TOTAL\_COST

FROM ORDER\_MASTER OM

JOIN ORDER\_DETAIL OD ON OM.ORDER\_CODE = OD.ORDER\_CODE

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

GROUP BY OM.CUSTOMER\_CODE

) T

JOIN CUSTOMER C ON T.CUSTOMER\_CODE = C.CUSTOMER\_CODE;

16. Which customer made the costliest order today?

Using RANK function

SELECT CUSTOMER\_CODE, CUSTOMER\_NAME, TOTAL\_COST

FROM (

SELECT OM.CUSTOMER\_CODE, C.CUSTOMER\_NAME,

SUM(P.PRICE \* OD.QTY\_ORDERED) AS TOTAL\_COST,

RANK() OVER (ORDER BY SUM(P.PRICE \* OD.QTY\_ORDERED) DESC) AS rnk

FROM ORDER\_DETAIL OD

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

JOIN ORDER\_MASTER OM ON OD.ORDER\_CODE = OM.ORDER\_CODE

JOIN CUSTOMER C ON OM.CUSTOMER\_CODE = C.CUSTOMER\_CODE

WHERE OM.ORDER\_DATE = CURDATE()

GROUP BY OM.CUSTOMER\_CODE, C.CUSTOMER\_NAME

) ranked

WHERE rnk = 1;

Using NOT IN

SELECT OM.CUSTOMER\_CODE, C.CUSTOMER\_NAME, SUM(P.PRICE \* OD.QTY\_ORDERED) AS TOTAL\_COST

FROM ORDER\_DETAIL OD

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

JOIN ORDER\_MASTER OM ON OD.ORDER\_CODE = OM.ORDER\_CODE

JOIN CUSTOMER C ON OM.CUSTOMER\_CODE = C.CUSTOMER\_CODE

WHERE OM.ORDER\_DATE = CURDATE()

GROUP BY OM.CUSTOMER\_CODE, C.CUSTOMER\_NAME

HAVING SUM(P.PRICE \* OD.QTY\_ORDERED) NOT IN (

SELECT SUM(P.PRICE \* OD.QTY\_ORDERED)

FROM ORDER\_DETAIL OD

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

JOIN ORDER\_MASTER OM ON OD.ORDER\_CODE = OM.ORDER\_CODE

WHERE OM.ORDER\_DATE = CURDATE()

GROUP BY OM.CUSTOMER\_CODE

HAVING SUM(P.PRICE \* OD.QTY\_ORDERED) < (

SELECT MAX(TOTAL\_COST)

FROM (

SELECT SUM(P.PRICE \* OD.QTY\_ORDERED) AS TOTAL\_COST

FROM ORDER\_DETAIL OD

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

JOIN ORDER\_MASTER OM ON OD.ORDER\_CODE = OM.ORDER\_CODE

WHERE OM.ORDER\_DATE = CURDATE()

GROUP BY OM.CUSTOMER\_CODE

) x

)

);

Using a plain subquery

SELECT OM.CUSTOMER\_CODE, C.CUSTOMER\_NAME, SUM(P.PRICE \* OD.QTY\_ORDERED) AS TOTAL\_COST

FROM ORDER\_DETAIL OD

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

JOIN ORDER\_MASTER OM ON OD.ORDER\_CODE = OM.ORDER\_CODE

JOIN CUSTOMER C ON OM.CUSTOMER\_CODE = C.CUSTOMER\_CODE

WHERE OM.ORDER\_DATE = CURDATE()

GROUP BY OM.CUSTOMER\_CODE, C.CUSTOMER\_NAME

HAVING SUM(P.PRICE \* OD.QTY\_ORDERED) = (

SELECT MAX(total)

FROM (

SELECT SUM(P.PRICE \* OD.QTY\_ORDERED) AS total

FROM ORDER\_DETAIL OD

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

JOIN ORDER\_MASTER OM ON OD.ORDER\_CODE = OM.ORDER\_CODE

WHERE OM.ORDER\_DATE = CURDATE()

GROUP BY OM.CUSTOMER\_CODE

) x

);

Using a correlated subquery

SELECT OM.CUSTOMER\_CODE, C.CUSTOMER\_NAME, SUM(P.PRICE \* OD.QTY\_ORDERED) AS TOTAL\_COST

FROM ORDER\_DETAIL OD

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

JOIN ORDER\_MASTER OM ON OD.ORDER\_CODE = OM.ORDER\_CODE

JOIN CUSTOMER C ON OM.CUSTOMER\_CODE = C.CUSTOMER\_CODE

WHERE OM.ORDER\_DATE = CURDATE()

GROUP BY OM.CUSTOMER\_CODE, C.CUSTOMER\_NAME

HAVING SUM(P.PRICE \* OD.QTY\_ORDERED) >= ALL (

SELECT SUM(P2.PRICE \* OD2.QTY\_ORDERED)

FROM ORDER\_DETAIL OD2

JOIN PRODUCT P2 ON OD2.PRODUCT\_CODE = P2.PRODUCT\_CODE

JOIN ORDER\_MASTER OM2 ON OD2.ORDER\_CODE = OM2.ORDER\_CODE

WHERE OM2.ORDER\_DATE = CURDATE()

GROUP BY OM2.CUSTOMER\_CODE

);

Using set operators

SELECT OM.CUSTOMER\_CODE, C.CUSTOMER\_NAME, SUM(P.PRICE \* OD.QTY\_ORDERED) AS TOTAL\_COST

FROM ORDER\_DETAIL OD

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

JOIN ORDER\_MASTER OM ON OD.ORDER\_CODE = OM.ORDER\_CODE

JOIN CUSTOMER C ON OM.CUSTOMER\_CODE = C.CUSTOMER\_CODE

WHERE OM.ORDER\_DATE = CURDATE()

GROUP BY OM.CUSTOMER\_CODE, C.CUSTOMER\_NAME

INTERSECT

SELECT OM.CUSTOMER\_CODE, C.CUSTOMER\_NAME, MAX(TOTAL\_COST)

FROM (

SELECT OM.CUSTOMER\_CODE, C.CUSTOMER\_NAME, SUM(P.PRICE \* OD.QTY\_ORDERED) AS TOTAL\_COST

FROM ORDER\_DETAIL OD

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

JOIN ORDER\_MASTER OM ON OD.ORDER\_CODE = OM.ORDER\_CODE

JOIN CUSTOMER C ON OM.CUSTOMER\_CODE = C.CUSTOMER\_CODE

WHERE OM.ORDER\_DATE = CURDATE()

GROUP BY OM.CUSTOMER\_CODE, C.CUSTOMER\_NAME

) ranked;

17. Generate the report like

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Customer name | Cust\_addr | Order code | Order date | No\_of\_products\_ordered | Total Bill |

Using JOIN

SELECT

C.CUSTOMER\_NAME,

OM.ORDER\_CODE,

OM.ORDER\_DATE,

COUNT(OD.PRODUCT\_CODE) AS NO\_OF\_PRODUCTS\_ORDERED,

SUM(P.PRICE \* OD.QTY\_ORDERED) AS TOTAL\_BILL

FROM CUSTOMER C

JOIN ORDER\_MASTER OM ON C.CUSTOMER\_CODE = OM.CUSTOMER\_CODE

JOIN ORDER\_DETAIL OD ON OM.ORDER\_CODE = OD.ORDER\_CODE

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

GROUP BY C.CUSTOMER\_NAME, OM.ORDER\_CODE, OM.ORDER\_DATE;

Using NOT IN (to get customer details where bill not among top few, logical extension)

SELECT C.CUSTOMER\_NAME, OM.ORDER\_CODE, OM.ORDER\_DATE,

COUNT(OD.PRODUCT\_CODE) AS NO\_OF\_PRODUCTS\_ORDERED,

SUM(P.PRICE \* OD.QTY\_ORDERED) AS TOTAL\_BILL

FROM CUSTOMER C

JOIN ORDER\_MASTER OM ON C.CUSTOMER\_CODE = OM.CUSTOMER\_CODE

JOIN ORDER\_DETAIL OD ON OM.ORDER\_CODE = OD.ORDER\_CODE

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

WHERE OM.ORDER\_CODE NOT IN (

SELECT ORDER\_CODE

FROM ORDER\_DETAIL

GROUP BY ORDER\_CODE

HAVING COUNT(PRODUCT\_CODE) < 1

)

GROUP BY C.CUSTOMER\_NAME, OM.ORDER\_CODE, OM.ORDER\_DATE;

Using a plain subquery (nested select for total\_bill)

SELECT C.CUSTOMER\_NAME, OM.ORDER\_CODE, OM.ORDER\_DATE,

COUNT(OD.PRODUCT\_CODE) AS NO\_OF\_PRODUCTS\_ORDERED,

(SELECT SUM(P2.PRICE \* OD2.QTY\_ORDERED)

FROM ORDER\_DETAIL OD2

JOIN PRODUCT P2 ON OD2.PRODUCT\_CODE = P2.PRODUCT\_CODE

WHERE OD2.ORDER\_CODE = OM.ORDER\_CODE) AS TOTAL\_BILL

FROM CUSTOMER C

JOIN ORDER\_MASTER OM ON C.CUSTOMER\_CODE = OM.CUSTOMER\_CODE

JOIN ORDER\_DETAIL OD ON OM.ORDER\_CODE = OD.ORDER\_CODE

GROUP BY C.CUSTOMER\_NAME, OM.ORDER\_CODE, OM.ORDER\_DATE;

Using a correlated subquery

SELECT C.CUSTOMER\_NAME, OM.ORDER\_CODE, OM.ORDER\_DATE,

(SELECT COUNT(\*)

FROM ORDER\_DETAIL OD2

WHERE OD2.ORDER\_CODE = OM.ORDER\_CODE) AS NO\_OF\_PRODUCTS\_ORDERED,

(SELECT SUM(P2.PRICE \* OD2.QTY\_ORDERED)

FROM ORDER\_DETAIL OD2

JOIN PRODUCT P2 ON OD2.PRODUCT\_CODE = P2.PRODUCT\_CODE

WHERE OD2.ORDER\_CODE = OM.ORDER\_CODE) AS TOTAL\_BILL

FROM CUSTOMER C

JOIN ORDER\_MASTER OM ON C.CUSTOMER\_CODE = OM.CUSTOMER\_CODE;

Using set operators (UNION for multiple rows if splitting is needed)

SELECT C.CUSTOMER\_NAME, OM.ORDER\_CODE, OM.ORDER\_DATE,

COUNT(OD.PRODUCT\_CODE) AS NO\_OF\_PRODUCTS\_ORDERED,

SUM(P.PRICE \* OD.QTY\_ORDERED) AS TOTAL\_BILL

FROM CUSTOMER C

JOIN ORDER\_MASTER OM ON C.CUSTOMER\_CODE = OM.CUSTOMER\_CODE

JOIN ORDER\_DETAIL OD ON OM.ORDER\_CODE = OD.ORDER\_CODE

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

GROUP BY C.CUSTOMER\_NAME, OM.ORDER\_CODE, OM.ORDER\_DATE

UNION

SELECT CUSTOMER\_NAME, NULL, NULL, 0, 0

FROM CUSTOMER

WHERE CUSTOMER\_CODE NOT IN (

SELECT CUSTOMER\_CODE FROM ORDER\_MASTER

);

18. Write query the customers who are eligible for home delivery (if tot\_bill>=5000)?

Using JOIN

SELECT

C.CUSTOMER\_NAME,

OM.ORDER\_CODE,

OM.ORDER\_DATE,

SUM(P.PRICE \* OD.QTY\_ORDERED) AS TOTAL\_BILL

FROM CUSTOMER C

JOIN ORDER\_MASTER OM ON C.CUSTOMER\_CODE = OM.CUSTOMER\_CODE

JOIN ORDER\_DETAIL OD ON OM.ORDER\_CODE = OD.ORDER\_CODE

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

GROUP BY C.CUSTOMER\_NAME, OM.ORDER\_CODE, OM.ORDER\_DATE

HAVING TOTAL\_BILL >= 5000;

Using NOT IN

SELECT CUSTOMER\_NAME, ORDER\_CODE, ORDER\_DATE, TOTAL\_BILL

FROM (

SELECT

C.CUSTOMER\_NAME,

OM.ORDER\_CODE,

OM.ORDER\_DATE,

SUM(P.PRICE \* OD.QTY\_ORDERED) AS TOTAL\_BILL

FROM CUSTOMER C

JOIN ORDER\_MASTER OM ON C.CUSTOMER\_CODE = OM.CUSTOMER\_CODE

JOIN ORDER\_DETAIL OD ON OM.ORDER\_CODE = OD.ORDER\_CODE

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

GROUP BY C.CUSTOMER\_NAME, OM.ORDER\_CODE, OM.ORDER\_DATE

) AS sub

WHERE TOTAL\_BILL NOT IN (

SELECT TOTAL\_BILL

FROM (

SELECT SUM(P.PRICE \* OD.QTY\_ORDERED) AS TOTAL\_BILL

FROM ORDER\_DETAIL OD

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

GROUP BY OD.ORDER\_CODE

) x

WHERE TOTAL\_BILL < 5000

);

Using correlated subquery

SELECT C.CUSTOMER\_NAME, OM.ORDER\_CODE, OM.ORDER\_DATE,

(SELECT SUM(P.PRICE \* OD2.QTY\_ORDERED)

FROM ORDER\_DETAIL OD2 JOIN PRODUCT P ON OD2.PRODUCT\_CODE = P.PRODUCT\_CODE

WHERE OD2.ORDER\_CODE = OM.ORDER\_CODE) AS TOTAL\_BILL

FROM CUSTOMER C

JOIN ORDER\_MASTER OM ON C.CUSTOMER\_CODE = OM.CUSTOMER\_CODE

WHERE (SELECT SUM(P.PRICE \* OD2.QTY\_ORDERED)

FROM ORDER\_DETAIL OD2 JOIN PRODUCT P ON OD2.PRODUCT\_CODE = P.PRODUCT\_CODE

WHERE OD2.ORDER\_CODE = OM.ORDER\_CODE) >= 5000;

Using set operators

SELECT CUSTOMER\_NAME, ORDER\_CODE, ORDER\_DATE, TOTAL\_BILL

FROM (

SELECT

C.CUSTOMER\_NAME,

OM.ORDER\_CODE,

OM.ORDER\_DATE,

SUM(P.PRICE \* OD.QTY\_ORDERED) AS TOTAL\_BILL

FROM CUSTOMER C

JOIN ORDER\_MASTER OM ON C.CUSTOMER\_CODE = OM.CUSTOMER\_CODE

JOIN ORDER\_DETAIL OD ON OM.ORDER\_CODE = OD.ORDER\_CODE

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

GROUP BY C.CUSTOMER\_NAME, OM.ORDER\_CODE, OM.ORDER\_DATE

) all\_bills

INTERSECT

SELECT CUSTOMER\_NAME, ORDER\_CODE, ORDER\_DATE, TOTAL\_BILL

FROM (

SELECT

C.CUSTOMER\_NAME,

OM.ORDER\_CODE,

OM.ORDER\_DATE,

SUM(P.PRICE \* OD.QTY\_ORDERED) AS TOTAL\_BILL

FROM CUSTOMER C

JOIN ORDER\_MASTER OM ON C.CUSTOMER\_CODE = OM.CUSTOMER\_CODE

JOIN ORDER\_DETAIL OD ON OM.ORDER\_CODE = OD.ORDER\_CODE

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

GROUP BY C.CUSTOMER\_NAME, OM.ORDER\_CODE, OM.ORDER\_DATE

HAVING SUM(P.PRICE \* OD.QTY\_ORDERED) >= 5000

);

19. Which country has maximum branches?

Using RANK function

SELECT COUNTRY\_NAME, TOTAL\_BRANCHES

FROM (

SELECT CN.COUNTRY\_NAME,

COUNT(B.BRANCH\_CODE) AS TOTAL\_BRANCHES,

RANK() OVER (ORDER BY COUNT(B.BRANCH\_CODE) DESC) AS rnk

FROM COUNTRY CN

JOIN BRANCH B ON B.COUNTRY\_CODE = CN.COUNTRY\_CODE

GROUP BY CN.COUNTRY\_NAME

) ranked

WHERE rnk = 1;

Using plain subquery

SELECT CN.COUNTRY\_NAME, COUNT(B.BRANCH\_CODE) AS TOTAL\_BRANCHES

FROM COUNTRY CN

JOIN BRANCH B ON B.COUNTRY\_CODE = CN.COUNTRY\_CODE

GROUP BY CN.COUNTRY\_NAME

HAVING COUNT(B.BRANCH\_CODE) = (

SELECT MAX(CNT)

FROM (

SELECT COUNT(B2.BRANCH\_CODE) AS CNT

FROM COUNTRY C2

JOIN BRANCH B2 ON B2.COUNTRY\_CODE = C2.COUNTRY\_CODE

GROUP BY C2.COUNTRY\_NAME

) y

);

Using NOT IN

SELECT CN.COUNTRY\_NAME, COUNT(B.BRANCH\_CODE) AS TOTAL\_BRANCHES

FROM COUNTRY CN

JOIN BRANCH B ON B.COUNTRY\_CODE = CN.COUNTRY\_CODE

GROUP BY CN.COUNTRY\_NAME

HAVING COUNT(B.BRANCH\_CODE) NOT IN (

SELECT CNT

FROM (

SELECT COUNT(B2.BRANCH\_CODE) AS CNT

FROM COUNTRY C2

JOIN BRANCH B2 ON B2.COUNTRY\_CODE = C2.COUNTRY\_CODE

GROUP BY C2.COUNTRY\_NAME

) x

WHERE CNT > COUNT(B.BRANCH\_CODE)

);

Using JOIN

SELECT a.COUNTRY\_NAME, a.TOTAL\_BRANCHES

FROM (

SELECT CN.COUNTRY\_NAME,

COUNT(B.BRANCH\_CODE) AS TOTAL\_BRANCHES

FROM COUNTRY CN

JOIN BRANCH B ON B.COUNTRY\_CODE = CN.COUNTRY\_CODE

GROUP BY CN.COUNTRY\_NAME

) a

JOIN (

SELECT MAX(cnt) AS MX

FROM (

SELECT COUNT(B2.BRANCH\_CODE) AS cnt

FROM COUNTRY C2

JOIN BRANCH B2 ON B2.COUNTRY\_CODE = C2.COUNTRY\_CODE

GROUP BY C2.COUNTRY\_NAME

) z

) m ON a.TOTAL\_BRANCHES = m.MX;

Using correlated subquery

SELECT CN.COUNTRY\_NAME

FROM COUNTRY CN

WHERE ( SELECT COUNT(\*)

FROM BRANCH B

WHERE B.COUNTRY\_CODE = CN.COUNTRY\_CODE ) >= ALL (

SELECT COUNT(\*)

FROM BRANCH B2

GROUP BY B2.COUNTRY\_CODE );

Using set operator

SELECT COUNTRY\_NAME, TOTAL\_BRANCHES

FROM (

SELECT CN.COUNTRY\_NAME, COUNT(B.BRANCH\_CODE) AS TOTAL\_BRANCHES

FROM COUNTRY CN

JOIN BRANCH B ON B.COUNTRY\_CODE = CN.COUNTRY\_CODE

GROUP BY CN.COUNTRY\_NAME

) s

EXCEPT

SELECT d.COUNTRY\_NAME, d.TOTAL\_BRANCHES

FROM (

SELECT CN1.COUNTRY\_NAME, COUNT(B1.BRANCH\_CODE) AS TOTAL\_BRANCHES

FROM COUNTRY CN1

JOIN BRANCH B1 ON B1.COUNTRY\_CODE = CN1.COUNTRY\_CODE

GROUP BY CN1.COUNTRY\_NAME

) d

JOIN (

SELECT CN2.COUNTRY\_NAME, COUNT(B2.BRANCH\_CODE) AS TOTAL\_BRANCHES

FROM COUNTRY CN2

JOIN BRANCH B2 ON B2.COUNTRY\_CODE = CN2.COUNTRY\_CODE

GROUP BY CN2.COUNTRY\_NAME

) e ON e.TOTAL\_BRANCHES > d.TOTAL\_BRANCHES;

20. Which country has minimum branches?

Using RANK function

SELECT COUNTRY\_NAME, TOTAL\_BRANCHES

FROM (

SELECT CN.COUNTRY\_NAME,

COUNT(B.BRANCH\_CODE) AS TOTAL\_BRANCHES,

RANK() OVER (ORDER BY COUNT(B.BRANCH\_CODE) ASC) AS rnk

FROM COUNTRY CN

LEFT JOIN BRANCH B ON B.COUNTRY\_CODE = CN.COUNTRY\_CODE

GROUP BY CN.COUNTRY\_NAME

) ranked

WHERE rnk = 1;

Using NOT IN

SELECT CN.COUNTRY\_NAME, COUNT(B.BRANCH\_CODE) AS TOTAL\_BRANCHES

FROM COUNTRY CN

LEFT JOIN BRANCH B ON B.COUNTRY\_CODE = CN.COUNTRY\_CODE

GROUP BY CN.COUNTRY\_NAME

HAVING COUNT(B.BRANCH\_CODE) NOT IN (

SELECT CNT

FROM (

SELECT COUNT(B2.BRANCH\_CODE) AS CNT

FROM COUNTRY C2

LEFT JOIN BRANCH B2 ON B2.COUNTRY\_CODE = C2.COUNTRY\_CODE

GROUP BY C2.COUNTRY\_NAME

) x

WHERE CNT < COUNT(B.BRANCH\_CODE)

);

Using a plain subquery

SELECT CN.COUNTRY\_NAME, COUNT(B.BRANCH\_CODE) AS TOTAL\_BRANCHES

FROM COUNTRY CN

LEFT JOIN BRANCH B ON CN.COUNTRY\_CODE = B.COUNTRY\_CODE

GROUP BY CN.COUNTRY\_NAME

HAVING COUNT(B.BRANCH\_CODE) = (

SELECT MIN(BRANCH\_COUNT) FROM (

SELECT COUNT(BR.BRANCH\_CODE) AS BRANCH\_COUNT

FROM COUNTRY C

LEFT JOIN BRANCH BR ON C.COUNTRY\_CODE = BR.COUNTRY\_CODE

GROUP BY C.COUNTRY\_NAME

) AS counts

);

Using JOIN

SELECT CN.COUNTRY\_NAME, COUNT(B.BRANCH\_CODE) AS TOTAL\_BRANCHES

FROM COUNTRY CN

LEFT JOIN BRANCH B ON CN.COUNTRY\_CODE = B.COUNTRY\_CODE

GROUP BY CN.COUNTRY\_NAME

JOIN (

SELECT MIN(BRANCH\_COUNT) AS MIN\_BRANCHES

FROM (

SELECT COUNT(BR.BRANCH\_CODE) AS BRANCH\_COUNT

FROM COUNTRY C

LEFT JOIN BRANCH BR ON C.COUNTRY\_CODE = BR.COUNTRY\_CODE

GROUP BY C.COUNTRY\_NAME

) AS counts

) min\_counts ON COUNT(B.BRANCH\_CODE) = min\_counts.MIN\_BRANCHES;

SELECT CN.COUNTRY\_NAME, COUNT(B.BRANCH\_CODE) AS TOTAL\_BRANCHES

FROM COUNTRY CN

LEFT JOIN BRANCH B ON CN.COUNTRY\_CODE = B.COUNTRY\_CODE

GROUP BY CN.COUNTRY\_NAME

HAVING COUNT(B.BRANCH\_CODE) = (

SELECT MIN(branch\_count) FROM (

SELECT COUNT(\*) AS branch\_count

FROM BRANCH

GROUP BY COUNTRY\_CODE

) AS sub

);

Using a correlated subquery

SELECT CN.COUNTRY\_NAME, COUNT(B.BRANCH\_CODE) AS TOTAL\_BRANCHES

FROM COUNTRY CN

LEFT JOIN BRANCH B ON CN.COUNTRY\_CODE = B.COUNTRY\_CODE

GROUP BY CN.COUNTRY\_NAME

HAVING COUNT(B.BRANCH\_CODE) = (

SELECT MIN(

(SELECT COUNT(\*)

FROM BRANCH BR2

WHERE BR2.COUNTRY\_CODE = C.COUNTRY\_CODE)

)

FROM COUNTRY C

);

Using set operators

SELECT CN.COUNTRY\_NAME, COUNT(B.BRANCH\_CODE) AS TOTAL\_BRANCHES

FROM COUNTRY CN

LEFT JOIN BRANCH B ON CN.COUNTRY\_CODE = B.COUNTRY\_CODE

GROUP BY CN.COUNTRY\_NAME

EXCEPT

SELECT CN2.COUNTRY\_NAME, COUNT(B2.BRANCH\_CODE) AS TOTAL\_BRANCHES

FROM COUNTRY CN2

LEFT JOIN BRANCH B2 ON CN2.COUNTRY\_CODE = B2.COUNTRY\_CODE

GROUP BY CN2.COUNTRY\_NAME

HAVING COUNT(B2.BRANCH\_CODE) > (

SELECT MIN(COUNT(B3.BRANCH\_CODE))

FROM COUNTRY C3

LEFT JOIN BRANCH B3 ON C3.COUNTRY\_CODE = B3.COUNTRY\_CODE

GROUP BY C3.COUNTRY\_NAME

);

21. Which country has no branches?  
 Using NOT IN with subquery  
SELECT COUNTRY\_CODE, COUNTRY\_NAME

FROM COUNTRY

WHERE COUNTRY\_CODE NOT IN (

SELECT DISTINCT COUNTRY\_CODE

FROM BRANCH

);

Using LEFT JOIN

SELECT c.COUNTRY\_CODE, c.COUNTRY\_NAME

FROM COUNTRY c

LEFT JOIN BRANCH b

ON c.COUNTRY\_CODE = b.COUNTRY\_CODE

WHERE b.BRANCH\_CODE IS NULL;

Using a correlated subquery

SELECT c.COUNTRY\_CODE, c.COUNTRY\_NAME

FROM COUNTRY c

WHERE NOT EXISTS (

SELECT 1

FROM BRANCH b

WHERE b.COUNTRY\_CODE = c.COUNTRY\_CODE

);

Using set operators

SELECT COUNTRY\_CODE, COUNTRY\_NAME

FROM COUNTRY

EXCEPT

SELECT c.COUNTRY\_CODE, c.COUNTRY\_NAME

FROM COUNTRY c

JOIN BRANCH b

ON c.COUNTRY\_CODE = b.COUNTRY\_CODE;

22. Which branch has more sales representatives?

Subquery with RANK()

SELECT BRANCH\_NAME, TOTAL\_SALESREPS

FROM (

SELECT B.BRANCH\_NAME, COUNT(SR.SALESREP\_ID) AS TOTAL\_SALESREPS,

RANK() OVER (ORDER BY COUNT(SR.SALESREP\_ID) DESC) AS rnk

FROM BRANCH B

LEFT JOIN SALESREP SR

ON B.BRANCH\_CODE = SR.BRANCH\_CODE

GROUP BY B.BRANCH\_NAME

) ranked

WHERE rnk = 1;

Using DENSE\_RANK()

SELECT BRANCH\_NAME, TOTAL\_SALESREPS

FROM (

SELECT B.BRANCH\_NAME, COUNT(SR.SALESREP\_ID) AS TOTAL\_SALESREPS,

DENSE\_RANK() OVER (ORDER BY COUNT(SR.SALESREP\_ID) DESC) AS rnk

FROM BRANCH B

LEFT JOIN SALESREP SR ON B.BRANCH\_CODE = SR.BRANCH\_CODE

GROUP BY B.BRANCH\_NAME

) ranked

WHERE rnk = 1;

22. Which branch has more sales representatives?

Subquery with RANK()

SELECT BRANCH\_NAME, TOTAL\_SALESREPS

FROM (

SELECT B.BRANCH\_NAME, COUNT(SR.SALESREP\_ID) AS TOTAL\_SALESREPS,

RANK() OVER (ORDER BY COUNT(SR.SALESREP\_ID) DESC) AS rnk

FROM BRANCH B

LEFT JOIN SALESREP SR ON B.BRANCH\_CODE = SR.BRANCH\_CODE

GROUP BY B.BRANCH\_NAME

) ranked

WHERE rnk = 1;

Alternative 1: Using DENSE\_RANK()

SELECT BRANCH\_NAME, TOTAL\_SALESREPS

FROM (

SELECT B.BRANCH\_NAME, COUNT(SR.SALESREP\_ID) AS TOTAL\_SALESREPS,

DENSE\_RANK() OVER (ORDER BY COUNT(SR.SALESREP\_ID) DESC) AS rnk

FROM BRANCH B

LEFT JOIN SALESREP SR ON B.BRANCH\_CODE = SR.BRANCH\_CODE

GROUP BY B.BRANCH\_NAME

) ranked

WHERE rnk = 1;

23. Which branch has less sales representatives?

Original Method: Subquery with RANK() ASC

SELECT BRANCH\_NAME, TOTAL\_SALESREPS

FROM (

SELECT B.BRANCH\_NAME, COUNT(SR.SALESREP\_ID) AS TOTAL\_SALESREPS,

RANK() OVER (ORDER BY COUNT(SR.SALESREP\_ID) ASC) AS rnk

FROM BRANCH B

LEFT JOIN SALESREP SR ON B.BRANCH\_CODE = SR.BRANCH\_CODE

GROUP BY B.BRANCH\_NAME

) ranked

WHERE rnk = 1;

Alternative 1: Using DENSE\_RANK()

SELECT BRANCH\_NAME, TOTAL\_SALESREPS

FROM (

SELECT B.BRANCH\_NAME, COUNT(SR.SALESREP\_ID) AS TOTAL\_SALESREPS,

DENSE\_RANK() OVER (ORDER BY COUNT(SR.SALESREP\_ID) ASC) AS rnk

FROM BRANCH B

LEFT JOIN SALESREP SR ON B.BRANCH\_CODE = SR.BRANCH\_CODE

GROUP BY B.BRANCH\_NAME

) ranked

WHERE rnk = 1;

24. Which branch has no sales representatives?

Original Method: LEFT JOIN with NULL check

SELECT B.BRANCH\_NAME

FROM BRANCH B

LEFT JOIN SALESREP SR ON B.BRANCH\_CODE = SR.BRANCH\_CODE

WHERE SR.SALESREP\_ID IS NULL;

Alternative 1: Using NOT EXISTS

SELECT B.BRANCH\_NAME

FROM BRANCH B

WHERE NOT EXISTS (SELECT 1 FROM SALESREP SR WHERE SR.BRANCH\_CODE = B.BRANCH\_CODE);

Alternative 2: Using NOT IN

SELECT B.BRANCH\_NAME

FROM BRANCH B

WHERE B.BRANCH\_CODE NOT IN (SELECT DISTINCT BRANCH\_CODE FROM SALESREP WHERE BRANCH\_CODE IS NOT NULL);

25. Who is the active sales rep?

Original Method: JOIN with ORDER\_MASTER

SELECT DISTINCT SR.SALESREP\_ID, SR.SALESREP\_NAME

FROM SALESREP SR

JOIN ORDER\_MASTER OM ON SR.SALESREP\_ID = OM.SALESREP\_ID;

Alternative 1: Using EXISTS

SELECT SR.SALESREP\_ID, SR.SALESREP\_NAME

FROM SALESREP SR

WHERE EXISTS (SELECT 1 FROM ORDER\_MASTER OM WHERE OM.SALESREP\_ID = SR.SALESREP\_ID);

Alternative 2: Using IN

SELECT SALESREP\_ID, SALESREP\_NAME

FROM SALESREP

WHERE SALESREP\_ID IN (SELECT DISTINCT SALESREP\_ID FROM ORDER\_MASTER);

26. Display all the manager's names?

Original Method: Selfjoin

SELECT DISTINCT S.SALESREP\_ID, S.SALESREP\_NAME AS Manager\_name

FROM SALESREP S

JOIN SALESREP M ON S.SALESREP\_ID = M.MGR;

Alternative 1: Using IN

SELECT SALESREP\_ID, SALESREP\_NAME AS Manager\_name

FROM SALESREP

WHERE SALESREP\_ID IN (SELECT DISTINCT MGR FROM SALESREP WHERE MGR IS NOT NULL);

Alternative 2: Using EXISTS

SELECT SALESREP\_ID, SALESREP\_NAME AS Manager\_name

FROM SALESREP S

WHERE EXISTS (SELECT 1 FROM SALESREP M WHERE M.MGR = S.SALESREP\_ID);

27. Display the sales rep that who has joined before their manager?

Original Method: Selfjoin with date comparison

SELECT E.SALESREP\_ID, E.SALESREP\_NAME, E.SALESREP\_DOJ,

M.SALESREP\_NAME AS Manager\_name, M.SALESREP\_DOJ AS MANAGER\_DOJ

FROM SALESREP E

JOIN SALESREP M ON E.MGR = M.SALESREP\_ID

WHERE E.SALESREP\_DOJ < M.SALESREP\_DOJ;

Alternative 1: Using EXISTS

SELECT E.SALESREP\_ID, E.SALESREP\_NAME, E.SALESREP\_DOJ

FROM SALESREP E

WHERE EXISTS (

SELECT 1 FROM SALESREP M

WHERE M.SALESREP\_ID = E.MGR AND E.SALESREP\_DOJ < M.SALESREP\_DOJ

);

28. Display the sales rep that who earns more than their manager?

Original Method: Selfjoin with salary comparison

SELECT E.SALESREP\_ID, E.SALESREP\_NAME, E.SALARY,

M.SALESREP\_NAME AS MANAGER\_NAME, M.SALARY AS MANAGER\_SALARY

FROM SALESREP E

JOIN SALESREP M ON E.MGR = M.SALESREP\_ID

WHERE E.SALARY > M.SALARY;

Alternative 1: Using EXISTS

SELECT E.SALESREP\_ID, E.SALESREP\_NAME, E.SALARY

FROM SALESREP E

WHERE EXISTS (

SELECT 1 FROM SALESREP M

WHERE M.SALESREP\_ID = E.MGR AND E.SALARY > M.SALARY

);

29. Display the manager who earns more than average salary of the sales rep working under them.

Original Method: Selfjoin with GROUP BY and HAVING

SELECT M.SALESREP\_ID, M.SALESREP\_NAME, M.SALARY, AVG(E.SALARY) AS AVG\_TEAM\_SALARY

FROM SALESREP M

JOIN SALESREP E ON M.SALESREP\_ID = E.MGR

GROUP BY M.SALESREP\_ID, M.SALESREP\_NAME, M.SALARY

HAVING M.SALARY > AVG(E.SALARY);

Alternative 1: Using subquery in HAVING

SELECT M.SALESREP\_ID, M.SALESREP\_NAME, M.SALARY,

(SELECT AVG(SALARY) FROM SALESREP E WHERE E.MGR = M.SALESREP\_ID) AS AVG\_TEAM\_SALARY

FROM SALESREP M

WHERE M.SALESREP\_ID IN (SELECT DISTINCT MGR FROM SALESREP WHERE MGR IS NOT NULL)

AND M.SALARY > (SELECT AVG(SALARY) FROM SALESREP E WHERE E.MGR = M.SALESREP\_ID);

30. Display the sales rep details that who started their work immediately?

Original Method: Subquery with RANK()

SELECT SALESREP\_ID, SALESREP\_NAME, SALESREP\_DOJ

FROM (

SELECT \*, RANK() OVER(ORDER BY SALESREP\_DOJ ASC) AS rnk

FROM SALESREP

) AS RankedReps

WHERE rnk = 1;

Alternative 1: Using DENSE\_RANK()

SELECT SALESREP\_ID, SALESREP\_NAME, SALESREP\_DOJ

FROM (

SELECT \*, DENSE\_RANK() OVER(ORDER BY SALESREP\_DOJ ASC) AS rnk

FROM SALESREP

) AS RankedReps

WHERE rnk = 1;

Alternative 2: Using subquery with MIN()

SELECT SALESREP\_ID, SALESREP\_NAME, SALESREP\_DOJ

FROM SALESREP

WHERE SALESREP\_DOJ = (SELECT MIN(SALESREP\_DOJ) FROM SALESREP);

31. Display the sales rep that who is in bench for a long time?

Original Method: RANK() with LEFT JOIN

SELECT SALESREP\_ID, SALESREP\_NAME, SALESREP\_DOJ

FROM (

SELECT S.SALESREP\_ID, S.SALESREP\_NAME, S.SALESREP\_DOJ,

RANK() OVER(ORDER BY (SALESREP\_DOJ) ASC) AS rnk

FROM SALESREP S

LEFT JOIN ORDER\_MASTER OM ON S.SALESREP\_ID = OM.SALESREP\_ID

WHERE OM.ORDER\_CODE IS NULL

) rnked

WHERE rnk = 1;

Alternative Methods:

1. Using NOT EXISTS:

SELECT SALESREP\_ID, SALESREP\_NAME, SALESREP\_DOJ

FROM SALESREP S

WHERE NOT EXISTS (SELECT 1 FROM ORDER\_MASTER OM WHERE OM.SALESREP\_ID = S.SALESREP\_ID)

AND SALESREP\_DOJ = (SELECT MIN(SALESREP\_DOJ) FROM SALESREP

WHERE SALESREP\_ID NOT IN (SELECT DISTINCT SALESREP\_ID FROM ORDER\_MASTER));

2. Using DENSE\_RANK():

SELECT SALESREP\_ID, SALESREP\_NAME, SALESREP\_DOJ

FROM (

SELECT S.SALESREP\_ID, S.SALESREP\_NAME, S.SALESREP\_DOJ,

DENSE\_RANK() OVER(ORDER BY (SALESREP\_DOJ) ASC) AS rnk

FROM SALESREP S

LEFT JOIN ORDER\_MASTER OM ON S.SALESREP\_ID = OM.SALESREP\_ID

WHERE OM.ORDER\_CODE IS NULL

) rnked

WHERE rnk = 1;

32. Display the details of inactive sales representatives.

Original Method: LEFT JOIN with NULL check

SELECT S.SALESREP\_ID, S.SALESREP\_NAME, S.SALESREP\_DOJ

FROM SALESREP S

LEFT JOIN ORDER\_MASTER OM ON S.SALESREP\_ID = OM.SALESREP\_ID

WHERE OM.ORDER\_CODE IS NULL;

Alternative Methods:

1. Using NOT EXISTS:

SELECT SALESREP\_ID, SALESREP\_NAME, SALESREP\_DOJ

FROM SALESREP S

WHERE NOT EXISTS (SELECT 1 FROM ORDER\_MASTER OM WHERE OM.SALESREP\_ID = S.SALESREP\_ID);

2. Using EXCEPT:

SELECT SALESREP\_ID, SALESREP\_NAME, SALESREP\_DOJ FROM SALESREP

EXCEPT

SELECT S.SALESREP\_ID, S.SALESREP\_NAME, S.SALESREP\_DOJ

FROM SALESREP S JOIN ORDER\_MASTER OM ON S.SALESREP\_ID = OM.SALESREP\_ID;

33. Generate the report

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sales\_rep\_id | Sales\_rep\_name | Salary | Total\_order\_value | Commission(5% of bill) |

Original Method: Multiple LEFT JOINs with COALESCE

SELECT S.SALESREP\_ID, S.SALESREP\_NAME, S.SALARY,

COALESCE(SUM(P.PRICE \* O.QTY\_ORDERED), 0) AS Total\_order\_value,

COALESCE(SUM(P.PRICE \* O.QTY\_ORDERED), 0) \* 0.05 AS Commision

FROM SALESREP S

LEFT JOIN ORDER\_MASTER OM ON S.SALESREP\_ID = OM.SALESREP\_ID

LEFT JOIN ORDER\_DETAIL O ON OM.ORDER\_CODE = O.ORDER\_CODE

LEFT JOIN PRODUCT P ON O.PRODUCT\_CODE = P.PRODUCT\_CODE

GROUP BY S.SALESREP\_ID, S.SALESREP\_NAME, S.SALARY;

Alternative Methods:

1. Using subqueries:

SELECT S.SALESREP\_ID, S.SALESREP\_NAME, S.SALARY,

(SELECT COALESCE(SUM(P.PRICE \* OD.QTY\_ORDERED), 0)

FROM ORDER\_MASTER OM

JOIN ORDER\_DETAIL OD ON OM.ORDER\_CODE = OD.ORDER\_CODE

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

WHERE OM.SALESREP\_ID = S.SALESREP\_ID) AS Total\_order\_value,

(SELECT COALESCE(SUM(P.PRICE \* OD.QTY\_ORDERED), 0) \* 0.05

FROM ORDER\_MASTER OM

JOIN ORDER\_DETAIL OD ON OM.ORDER\_CODE = OD.ORDER\_CODE

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

WHERE OM.SALESREP\_ID = S.SALESREP\_ID) AS Commision

FROM SALESREP S;

34. Generate the report like

|  |  |  |
| --- | --- | --- |
| Sales\_rep\_name | Order\_code | Bill\_value |

Original Method: Multiple JOINs

SELECT S.SALESREP\_NAME, O.ORDER\_CODE,

SUM(P.PRICE \* O.QTY\_ORDERED) AS BILL\_VALUE

FROM SALESREP S

JOIN ORDER\_MASTER OM ON S.SALESREP\_ID = OM.SALESREP\_ID

JOIN ORDER\_DETAIL O ON OM.ORDER\_CODE = O.ORDER\_CODE

JOIN PRODUCT P ON O.PRODUCT\_CODE = P.PRODUCT\_CODE

GROUP BY S.SALESREP\_NAME, O.ORDER\_CODE;

Alternative Methods:

1. Using CTE:

WITH OrderValues AS (

SELECT OM.SALESREP\_ID, OD.ORDER\_CODE, SUM(P.PRICE \* OD.QTY\_ORDERED) AS BILL\_VALUE

FROM ORDER\_MASTER OM

JOIN ORDER\_DETAIL OD ON OM.ORDER\_CODE = OD.ORDER\_CODE

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

GROUP BY OM.SALESREP\_ID, OD.ORDER\_CODE

)

SELECT S.SALESREP\_NAME, OV.ORDER\_CODE, OV.BILL\_VALUE

FROM OrderValues OV

JOIN SALESREP S ON OV.SALESREP\_ID = S.SALESREP\_ID;

35. Generate the report

|  |  |  |
| --- | --- | --- |
| Sales\_rep\_name | Order\_code | No\_of\_products(in order) |

Original Method: Multiple JOINs

SELECT S.SALESREP\_NAME, OM.ORDER\_CODE,

SUM(OD.QTY\_ORDERED) AS NO\_OF\_PRODUCTS

FROM ORDER\_MASTER OM

JOIN SALESREP S ON OM.SALESREP\_ID = S.SALESREP\_ID

JOIN ORDER\_DETAIL OD ON OM.ORDER\_CODE = OD.ORDER\_CODE

GROUP BY S.SALESREP\_NAME, OM.ORDER\_CODE;

Alternative Methods:

1. Using subquery:

SELECT S.SALESREP\_NAME, OM.ORDER\_CODE,

(SELECT SUM(QTY\_ORDERED) FROM ORDER\_DETAIL WHERE ORDER\_CODE = OM.ORDER\_CODE) AS NO\_OF\_PRODUCTS

FROM ORDER\_MASTER OM

JOIN SALESREP S ON OM.SALESREP\_ID = S.SALESREP\_ID;

36. Generate the report like

|  |  |  |  |
| --- | --- | --- | --- |
| Sales\_rep\_name | Order\_code | Bill\_value | No\_of\_products(in order) |

A. For orders placed in 2010:

Original method:

SELECT S.SALESREP\_NAME, OM.ORDER\_CODE,

SUM(P.PRICE \* OD.QTY\_ORDERED) AS BILL\_VALUE,

SUM(OD.QTY\_ORDERED) AS NO\_OF\_PRODUCTS

FROM ORDER\_MASTER OM

JOIN SALESREP S ON OM.SALESREP\_ID = S.SALESREP\_ID

JOIN ORDER\_DETAIL OD ON OM.ORDER\_CODE = OD.ORDER\_CODE

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

WHERE YEAR(OM.ORDER\_DATE) = 2010

GROUP BY S.SALESREP\_NAME, OM.ORDER\_CODE;

Alternative using BETWEEN:

SELECT S.SALESREP\_NAME, OM.ORDER\_CODE,

SUM(P.PRICE \* OD.QTY\_ORDERED) AS BILL\_VALUE,

SUM(OD.QTY\_ORDERED) AS NO\_OF\_PRODUCTS

FROM ORDER\_MASTER OM

JOIN SALESREP S ON OM.SALESREP\_ID = S.SALESREP\_ID

JOIN ORDER\_DETAIL OD ON OM.ORDER\_CODE = OD.ORDER\_CODE

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

WHERE OM.ORDER\_DATE BETWEEN '20100101' AND '20101231'

GROUP BY S.SALESREP\_NAME, OM.ORDER\_CODE;

B. For sales reps with ≤10 orders:

Original method:

SELECT S.SALESREP\_NAME, OM.ORDER\_CODE,

SUM(P.PRICE \* OD.QTY\_ORDERED) AS BILL\_VALUE,

SUM(OD.QTY\_ORDERED) AS NO\_OF\_PRODUCTS

FROM ORDER\_MASTER OM

JOIN SALESREP S ON OM.SALESREP\_ID = S.SALESREP\_ID

JOIN ORDER\_DETAIL OD ON OM.ORDER\_CODE = OD.ORDER\_CODE

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

GROUP BY S.SALESREP\_NAME, OM.ORDER\_CODE

HAVING COUNT(OM.ORDER\_CODE) <= 10;

Alternative with CTE:

WITH RepOrderCount AS (

SELECT SALESREP\_ID, COUNT(ORDER\_CODE) AS order\_count

FROM ORDER\_MASTER

GROUP BY SALESREP\_ID

HAVING COUNT(ORDER\_CODE) <= 10

)

SELECT S.SALESREP\_NAME, OM.ORDER\_CODE,

SUM(P.PRICE \* OD.QTY\_ORDERED) AS BILL\_VALUE,

SUM(OD.QTY\_ORDERED) AS NO\_OF\_PRODUCTS

FROM ORDER\_MASTER OM

JOIN SALESREP S ON OM.SALESREP\_ID = S.SALESREP\_ID

JOIN ORDER\_DETAIL OD ON OM.ORDER\_CODE = OD.ORDER\_CODE

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

JOIN RepOrderCount ROC ON OM.SALESREP\_ID = ROC.SALESREP\_ID

GROUP BY S.SALESREP\_NAME, OM.ORDER\_CODE;

C. For sales reps with salary >45000:

Original method:

SELECT S.SALESREP\_NAME, OM.ORDER\_CODE,

SUM(P.PRICE \* OD.QTY\_ORDERED) AS BILL\_VALUE,

SUM(OD.QTY\_ORDERED) AS NO\_OF\_PRODUCTS

FROM ORDER\_MASTER OM

JOIN SALESREP S ON OM.SALESREP\_ID = S.SALESREP\_ID

JOIN ORDER\_DETAIL OD ON OM.ORDER\_CODE = OD.ORDER\_CODE

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

WHERE S.SALARY > 45000

GROUP BY S.SALESREP\_NAME, OM.ORDER\_CODE;

Alternative with subquery:

SELECT S.SALESREP\_NAME, OM.ORDER\_CODE,

SUM(P.PRICE \* OD.QTY\_ORDERED) AS BILL\_VALUE,

SUM(OD.QTY\_ORDERED) AS NO\_OF\_PRODUCTS

FROM ORDER\_MASTER OM

JOIN (SELECT SALESREP\_ID, SALESREP\_NAME FROM SALESREP WHERE SALARY > 45000) S

ON OM.SALESREP\_ID = S.SALESREP\_ID

JOIN ORDER\_DETAIL OD ON OM.ORDER\_CODE = OD.ORDER\_CODE

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

GROUP BY S.SALESREP\_NAME, OM.ORDER\_CODE;

37. Who is the senior most sales rep?

Original Method: RANK()

SELECT SALESREP\_ID, SALESREP\_NAME, SALESREP\_DOJ

FROM (

SELECT \*, RANK() OVER(ORDER BY (SALESREP\_DOJ) ASC) AS rnk

FROM SALESREP

) rnked

WHERE rnk = 1;

Alternative Methods:

1. Using subquery with MIN():

SELECT SALESREP\_ID, SALESREP\_NAME, SALESREP\_DOJ

FROM SALESREP

WHERE SALESREP\_DOJ = (SELECT MIN(SALESREP\_DOJ) FROM SALESREP);

2. Using TOP 1 WITH TIES (SQL Server):

SELECT TOP 1 WITH TIES SALESREP\_ID, SALESREP\_NAME, SALESREP\_DOJ

FROM SALESREP

ORDER BY SALESREP\_DOJ ASC;

38. Find the no of orders processed by junior most sales rep

Original Method: RANK() with LEFT JOIN

SELECT SALESREP\_ID, SALESREP\_NAME, NO\_OF\_ORDERS

FROM (

SELECT S.SALESREP\_ID, S.SALESREP\_NAME, S.SALESREP\_DOJ,

COUNT(O.ORDER\_CODE) AS NO\_OF\_ORDERS,

RANK() OVER (ORDER BY S.SALESREP\_DOJ DESC) AS rnk

FROM SALESREP S

LEFT JOIN ORDER\_MASTER O ON S.SALESREP\_ID = O.SALESREP\_ID

GROUP BY S.SALESREP\_ID, S.SALESREP\_NAME, S.SALESREP\_DOJ

) AS ranked\_salesreps

WHERE rnk = 1;

Alternative Methods:

1. Using DENSE\_RANK():

SELECT SALESREP\_ID, SALESREP\_NAME, NO\_OF\_ORDERS

FROM (

SELECT S.SALESREP\_ID, S.SALESREP\_NAME, S.SALESREP\_DOJ,

COUNT(O.ORDER\_CODE) AS NO\_OF\_ORDERS,

DENSE\_RANK() OVER (ORDER BY S.SALESREP\_DOJ DESC) AS rnk

FROM SALESREP S

LEFT JOIN ORDER\_MASTER O ON S.SALESREP\_ID = O.SALESREP\_ID

GROUP BY S.SALESREP\_ID, S.SALESREP\_NAME, S.SALESREP\_DOJ

) AS ranked\_salesreps

WHERE rnk = 1;

39. Find the no of orders processed by each month in 2010

Original Method: GROUP BY with MONTH()

SELECT COUNT(ORDER\_CODE) AS NO\_OF\_ORDERS,

MONTH(ORDER\_DATE) AS MONTHLY\_PROCESSING

FROM ORDER\_MASTER

WHERE YEAR(ORDER\_DATE) = 2010

GROUP BY MONTH(ORDER\_DATE);

Alternative Methods:

1. Using DATEPART (more standard SQL):

SELECT COUNT(ORDER\_CODE) AS NO\_OF\_ORDERS,

DATEPART(month, ORDER\_DATE) AS MONTHLY\_PROCESSING

FROM ORDER\_MASTER

WHERE DATEPART(year, ORDER\_DATE) = 2010

GROUP BY DATEPART(month, ORDER\_DATE)

ORDER BY MONTHLY\_PROCESSING;

2. Using EXTRACT (ANSI SQL):

SELECT COUNT(ORDER\_CODE) AS NO\_OF\_ORDERS,

EXTRACT(month FROM ORDER\_DATE) AS MONTHLY\_PROCESSING

FROM ORDER\_MASTER

WHERE EXTRACT(year FROM ORDER\_DATE) = 2010

GROUP BY EXTRACT(month FROM ORDER\_DATE)

ORDER BY MONTHLY\_PROCESSING;

40. Which region is having inactive sales rep?

Original Method: Multiple JOINs with LEFT JOIN and NULL check

SELECT S.SALESREP\_ID, S.SALESREP\_NAME, C.REGION

FROM COUNTRY C

JOIN BRANCH B ON C.COUNTRY\_CODE = B.COUNTRY\_CODE

JOIN SALESREP S ON S.BRANCH\_CODE = B.BRANCH\_CODE

LEFT JOIN ORDER\_MASTER OM ON S.SALESREP\_ID = OM.SALESREP\_ID

WHERE OM.ORDER\_CODE IS NULL;

Alternative Methods:

1. Using NOT EXISTS:

SELECT S.SALESREP\_ID, S.SALESREP\_NAME, C.REGION

FROM SALESREP S

JOIN BRANCH B ON S.BRANCH\_CODE = B.BRANCH\_CODE

JOIN COUNTRY C ON B.COUNTRY\_CODE = C.COUNTRY\_CODE

WHERE NOT EXISTS (SELECT 1 FROM ORDER\_MASTER OM WHERE OM.SALESREP\_ID = S.SALESREP\_ID);

2. Using EXCEPT:

SELECT S.SALESREP\_ID, S.SALESREP\_NAME, C.REGION

FROM SALESREP S

JOIN BRANCH B ON S.BRANCH\_CODE = B.BRANCH\_CODE

JOIN COUNTRY C ON B.COUNTRY\_CODE = C.COUNTRY\_CODE

EXCEPT

SELECT S.SALESREP\_ID, S.SALESREP\_NAME, C.REGION

FROM SALESREP S

JOIN BRANCH B ON S.BRANCH\_CODE = B.BRANCH\_CODE

JOIN COUNTRY C ON B.COUNTRY\_CODE = C.COUNTRY\_CODE

JOIN ORDER\_MASTER OM ON S.SALESREP\_ID = OM.SALESREP\_ID;

41. Which region is having junior most sales rep?

Original Method: RANK()

SELECT REGION, SALESREP\_ID, SALESREP\_NAME, SALESREP\_DOJ

FROM (

SELECT C.REGION, S.SALESREP\_ID, S.SALESREP\_NAME, S.SALESREP\_DOJ,

RANK() OVER(ORDER BY(S.SALESREP\_DOJ) DESC) AS rnk

FROM SALESREP S

JOIN BRANCH B ON S.BRANCH\_CODE = B.BRANCH\_CODE

JOIN COUNTRY C ON B.COUNTRY\_CODE = C.COUNTRY\_CODE

) rnked

WHERE rnk = 1;

Alternative Methods:

1. Using subquery with MAX():

SELECT C.REGION, S.SALESREP\_ID, S.SALESREP\_NAME, S.SALESREP\_DOJ

FROM SALESREP S

JOIN BRANCH B ON S.BRANCH\_CODE = B.BRANCH\_CODE

JOIN COUNTRY C ON B.COUNTRY\_CODE = C.COUNTRY\_CODE

WHERE S.SALESREP\_DOJ = (SELECT MAX(SALESREP\_DOJ) FROM SALESREP);

42. Find the no of orders whose bill is less than 500

Original Method: JOIN with HAVING

SELECT O.ORDER\_CODE, SUM(P.PRICE \* O.QTY\_ORDERED) AS BILL,

COUNT(O.ORDER\_CODE) AS NO\_OF\_ORDERS

FROM ORDER\_DETAIL O

JOIN ORDER\_MASTER OM ON O.ORDER\_CODE = OM.ORDER\_CODE

JOIN PRODUCT P ON O.PRODUCT\_CODE = P.PRODUCT\_CODE

GROUP BY O.ORDER\_CODE

HAVING SUM(P.PRICE \* O.QTY\_ORDERED) < 500;

Alternative Methods:

1. Using CTE:

WITH OrderBills AS (

SELECT O.ORDER\_CODE, SUM(P.PRICE \* O.QTY\_ORDERED) AS BILL

FROM ORDER\_DETAIL O

JOIN PRODUCT P ON O.PRODUCT\_CODE = P.PRODUCT\_CODE

GROUP BY O.ORDER\_CODE

HAVING SUM(P.PRICE \* O.QTY\_ORDERED) < 500

)

SELECT OB.ORDER\_CODE, OB.BILL, COUNT(OM.ORDER\_CODE) AS NO\_OF\_ORDERS

FROM OrderBills OB

JOIN ORDER\_MASTER OM ON OB.ORDER\_CODE = OM.ORDER\_CODE

GROUP BY OB.ORDER\_CODE, OB.BILL;

43. Frequently ordered product by most active customer

Original Method: Nested RANK() subqueries

SELECT PRODUCT\_CODE, ORDER\_COUNT

FROM (

SELECT OD.PRODUCT\_CODE,

COUNT(OD.ORDER\_CODE) AS ORDER\_COUNT,

RANK() OVER (ORDER BY COUNT(OD.ORDER\_CODE) DESC) AS rnk

FROM ORDER\_DETAIL OD

JOIN ORDER\_MASTER OM ON OD.ORDER\_CODE = OM.ORDER\_CODE

WHERE OM.CUSTOMER\_CODE = (

SELECT CUSTOMER\_CODE

FROM (

SELECT CUSTOMER\_CODE,

RANK() OVER (ORDER BY COUNT(ORDER\_CODE) DESC) AS rnk

FROM ORDER\_MASTER

GROUP BY CUSTOMER\_CODE

) ranked\_customers

WHERE rnk = 1

)

GROUP BY OD.PRODUCT\_CODE

) ranked\_products

WHERE rnk = 1;

Alternative Methods:

1. Using CTEs for better readability:

WITH MostActiveCustomer AS (

SELECT CUSTOMER\_CODE

FROM ORDER\_MASTER

GROUP BY CUSTOMER\_CODE

ORDER BY COUNT(ORDER\_CODE) DESC

LIMIT 1

),

ProductOrderCounts AS (

SELECT OD.PRODUCT\_CODE, COUNT(\*) AS ORDER\_COUNT

FROM ORDER\_DETAIL OD

JOIN ORDER\_MASTER OM ON OD.ORDER\_CODE = OM.ORDER\_CODE

WHERE OM.CUSTOMER\_CODE = (SELECT CUSTOMER\_CODE FROM MostActiveCustomer)

GROUP BY OD.PRODUCT\_CODE

)

SELECT PRODUCT\_CODE, ORDER\_COUNT

FROM (

SELECT PRODUCT\_CODE, ORDER\_COUNT,

RANK() OVER (ORDER BY ORDER\_COUNT DESC) AS rnk

FROM ProductOrderCounts

) ranked

WHERE rnk = 1;

44. Products not ordered by active customer

Original Method: NOT IN with subquery

SELECT P.PRODUCT\_CODE

FROM PRODUCT P

WHERE P.PRODUCT\_CODE NOT IN (

SELECT OD.PRODUCT\_CODE

FROM ORDER\_DETAIL OD

JOIN ORDER\_MASTER OM ON OD.ORDER\_CODE = OM.ORDER\_CODE

WHERE OM.CUSTOMER\_CODE = (

SELECT CUSTOMER\_CODE

FROM (

SELECT CUSTOMER\_CODE,

RANK() OVER (ORDER BY COUNT(ORDER\_CODE) DESC) AS rnk

FROM ORDER\_MASTER

GROUP BY CUSTOMER\_CODE

) ranked\_customers

WHERE rnk = 1

)

);

Alternative Methods:

1. Using NOT EXISTS:

SELECT P.PRODUCT\_CODE

FROM PRODUCT P

WHERE NOT EXISTS (

SELECT 1

FROM ORDER\_DETAIL OD

JOIN ORDER\_MASTER OM ON OD.ORDER\_CODE = OM.ORDER\_CODE

WHERE OM.CUSTOMER\_CODE = (

SELECT CUSTOMER\_CODE

FROM ORDER\_MASTER

GROUP BY CUSTOMER\_CODE

ORDER BY COUNT(ORDER\_CODE) DESC

LIMIT 1

)

AND OD.PRODUCT\_CODE = P.PRODUCT\_CODE

);

45. Orders which can't be processed immediately (QTY\_ORDERED > QTY\_ON\_HAND)

Original Method: Simple JOIN with filter

SELECT OD.ORDER\_CODE, P.PRODUCT\_CODE, OD.QTY\_ORDERED, P.QTY\_ON\_HAND

FROM ORDER\_DETAIL OD

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

WHERE OD.QTY\_ORDERED > P.QTY\_ON\_HAND;

Alternative Methods:

1. Using EXISTS:

SELECT OD.ORDER\_CODE, OD.PRODUCT\_CODE, OD.QTY\_ORDERED, P.QTY\_ON\_HAND

FROM ORDER\_DETAIL OD

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

WHERE EXISTS (

SELECT 1

WHERE OD.QTY\_ORDERED > P.QTY\_ON\_HAND

);

46. Complete order report

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Cust\_name | Cust\_address | Order\_code | Product\_id | Product\_name | Order\_date |

Original Method: Multiple JOINs

SELECT C.CUSTOMER\_NAME, C.CUSTOMER\_ADDRESS, OM.ORDER\_CODE,

P.PRODUCT\_CODE, P.PRODUCT\_NAME, OM.ORDER\_DATE

FROM CUSTOMER C

JOIN ORDER\_MASTER OM ON C.CUSTOMER\_CODE = OM.CUSTOMER\_CODE

JOIN ORDER\_DETAIL OD ON OM.ORDER\_CODE = OD.ORDER\_CODE

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE;

Alternative Methods:

1. Using CTE for better organization:

WITH OrderDetails AS (

SELECT OM.CUSTOMER\_CODE, OM.ORDER\_CODE, OM.ORDER\_DATE,

OD.PRODUCT\_CODE, P.PRODUCT\_NAME

FROM ORDER\_MASTER OM

JOIN ORDER\_DETAIL OD ON OM.ORDER\_CODE = OD.ORDER\_CODE

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

)

SELECT C.CUSTOMER\_NAME, C.CUSTOMER\_ADDRESS,

OD.ORDER\_CODE, OD.PRODUCT\_CODE, OD.PRODUCT\_NAME, OD.ORDER\_DATE

FROM CUSTOMER C

JOIN OrderDetails OD ON C.CUSTOMER\_CODE = OD.CUSTOMER\_CODE;

48. Department wise maximum salary earner

Original Method: RANK() with PARTITION BY

SELECT BRANCH\_CODE, SALESREP\_ID, SALESREP\_NAME, SALARY

FROM (

SELECT B.BRANCH\_CODE, S.SALESREP\_ID, S.SALESREP\_NAME, S.SALARY,

RANK() OVER(PARTITION BY BRANCH\_CODE ORDER BY (S.SALARY)DESC) AS rnk

FROM BRANCH B

JOIN SALESREP S ON B.BRANCH\_CODE = S.BRANCH\_CODE

GROUP BY B.BRANCH\_CODE, S.SALESREP\_ID, S.SALESREP\_NAME

) AS rnked

WHERE rnk = 1;

Alternative Methods:

1. Using correlated subquery:

SELECT S1.BRANCH\_CODE, S1.SALESREP\_ID, S1.SALESREP\_NAME, S1.SALARY

FROM SALESREP S1

WHERE S1.SALARY = (

SELECT MAX(S2.SALARY)

FROM SALESREP S2

WHERE S2.BRANCH\_CODE = S1.BRANCH\_CODE

);

49. Top10 active customers based on no. of orders

Original Method: RANK() with subquery

SELECT CUSTOMER\_CODE, ORDER\_COUNT

FROM (

SELECT OM.CUSTOMER\_CODE, COUNT(OD.ORDER\_CODE) AS ORDER\_COUNT,

RANK() OVER(ORDER BY COUNT(OD.ORDER\_CODE) DESC) AS rnk

FROM ORDER\_DETAIL OD

JOIN ORDER\_MASTER OM ON OD.ORDER\_CODE = OM.ORDER\_CODE

GROUP BY OM.CUSTOMER\_CODE

) AS rnked

WHERE rnk <= 10;

Alternative Methods:

1. Using LIMIT (MySQL/PostgreSQL):

SELECT OM.CUSTOMER\_CODE, COUNT(OD.ORDER\_CODE) AS ORDER\_COUNT

FROM ORDER\_DETAIL OD

JOIN ORDER\_MASTER OM ON OD.ORDER\_CODE = OM.ORDER\_CODE

GROUP BY OM.CUSTOMER\_CODE

ORDER BY ORDER\_COUNT DESC

LIMIT 10;

2. Using TOP (SQL Server):

SELECT TOP 10 OM.CUSTOMER\_CODE, COUNT(OD.ORDER\_CODE) AS ORDER\_COUNT

FROM ORDER\_DETAIL OD

JOIN ORDER\_MASTER OM ON OD.ORDER\_CODE = OM.ORDER\_CODE

GROUP BY OM.CUSTOMER\_CODE

ORDER BY ORDER\_COUNT DESC;

50. Top5 orders based on bill value

Original Method: RANK() with subquery

SELECT ORDER\_CODE, BILL\_VALUE

FROM (

SELECT OD.ORDER\_CODE, SUM(P.PRICE \* OD.QTY\_ORDERED) AS BILL\_VALUE,

RANK() OVER(ORDER BY SUM(P.PRICE \* OD.QTY\_ORDERED) DESC) AS rnk

FROM ORDER\_DETAIL OD

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

GROUP BY OD.ORDER\_CODE

) rnked

WHERE rnk <= 5;

Alternative Methods:

1. Using LIMIT:

SELECT OD.ORDER\_CODE, SUM(P.PRICE \* OD.QTY\_ORDERED) AS BILL\_VALUE

FROM ORDER\_DETAIL OD

JOIN PRODUCT P ON OD.PRODUCT\_CODE = P.PRODUCT\_CODE

GROUP BY OD.ORDER\_CODE

ORDER BY BILL\_VALUE DESC

LIMIT 5;

51. Sales rep who processed ORD11 and ORD15

Original Method: IN with subquery

SELECT SALESREP\_ID, SALESREP\_NAME

FROM SALESREP

WHERE SALESREP\_ID IN (

SELECT SALESREP\_ID

FROM ORDER\_MASTER

WHERE ORDER\_CODE IN ('ORD11', 'ORD15')

GROUP BY SALESREP\_ID

HAVING COUNT(DISTINCT ORDER\_CODE) = 2

);

Alternative Methods:

1. Using INTERSECT:

SELECT S.SALESREP\_ID, S.SALESREP\_NAME

FROM SALESREP S

WHERE EXISTS (

SELECT 1 FROM ORDER\_MASTER WHERE SALESREP\_ID = S.SALESREP\_ID AND ORDER\_CODE = 'ORD11'

)

INTERSECT

SELECT S.SALESREP\_ID, S.SALESREP\_NAME

FROM SALESREP S

WHERE EXISTS (

SELECT 1 FROM ORDER\_MASTER WHERE SALESREP\_ID = S.SALESREP\_ID AND ORDER\_CODE = 'ORD15'

);

52. Generate report "worker works for manager"

Original Method: CONCAT with selfjoin

SELECT CONCAT(sr.SALESREP\_NAME, ' works for ', m.SALESREP\_NAME) AS relationship

FROM SALESREP sr

JOIN SALESREP m ON sr.MGR = m.SALESREP\_ID;

Alternative Methods:

1. Using string concatenation with + (SQL Server):

SELECT sr.SALESREP\_NAME + ' works for ' + m.SALESREP\_NAME AS relationship

FROM SALESREP sr

JOIN SALESREP m ON sr.MGR = m.SALESREP\_ID;