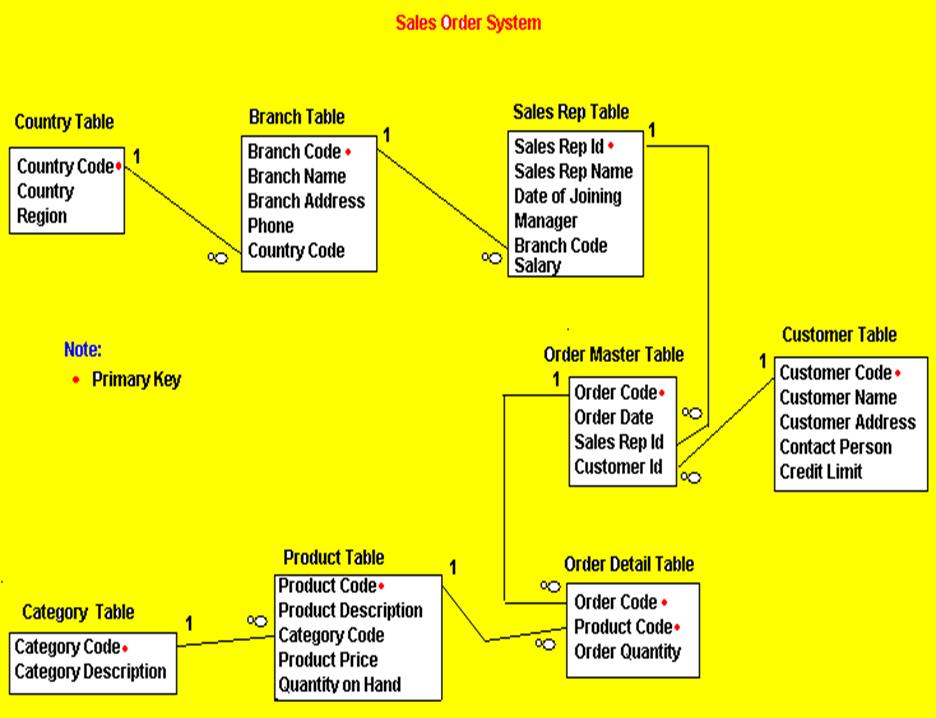
Sales Order Management



USE salesordermanagement;

CREATE TABLE COUNTRY (

COUNTRY\_CODE CHAR(4) PRIMARY KEY,

COUNTRY\_NAME VARCHAR(15),

REGION VARCHAR(15)

);

CREATE TABLE CATEGORY (

CATEGORY\_CODE CHAR(4) PRIMARY KEY,

CATEGORY\_DESC VARCHAR(20)

);

CREATE TABLE CUSTOMER (

CUSTOMER\_CODE CHAR(4) PRIMARY KEY,

CUSTOMER\_NAME VARCHAR(20),

CREDIT\_LIMIT DECIMAL(10,2)

);

CREATE TABLE BRANCH (

BRANCH\_CODE CHAR(4) PRIMARY KEY,

BRANCH\_NAME VARCHAR(15),

COUNTRY\_CODE CHAR(4),

FOREIGN KEY (COUNTRY\_CODE) REFERENCES COUNTRY(COUNTRY\_CODE)

);

CREATE TABLE SALESREP (

SALESREP\_ID CHAR(4) PRIMARY KEY,

SALESREP\_NAME VARCHAR(20),

SALESREP\_DOJ DATE,

MGR CHAR(4),

BRANCH\_CODE CHAR(4),

SALARY DECIMAL(10,2),

FOREIGN KEY (BRANCH\_CODE) REFERENCES BRANCH(BRANCH\_CODE)

);

CREATE TABLE PRODUCT (

PRODUCT\_CODE CHAR(4) PRIMARY KEY,

PROD\_DESC VARCHAR(30),

CATEGORY\_CODE CHAR(4),

PRICE DECIMAL(10,2),

QTY\_ON\_HAND INT,

FOREIGN KEY (CATEGORY\_CODE) REFERENCES CATEGORY(CATEGORY\_CODE)

);

CREATE TABLE ORDER\_MASTER (

ORDER\_CODE CHAR(4) PRIMARY KEY,

ORDER\_DATE DATE,

SALESREP\_ID CHAR(4),

CUSTOMER\_CODE CHAR(4),

FOREIGN KEY (SALESREP\_ID) REFERENCES SALESREP(SALESREP\_ID),

FOREIGN KEY (CUSTOMER\_CODE) REFERENCES CUSTOMER(CUSTOMER\_CODE)

);

CREATE TABLE ORDER\_DETAIL (

ORDER\_CODE CHAR(4),

PRODUCT\_CODE CHAR(4),

QTY\_ORDERED INT,

PRIMARY KEY (ORDER\_CODE, PRODUCT\_CODE),

FOREIGN KEY (ORDER\_CODE) REFERENCES ORDER\_MASTER(ORDER\_CODE),

FOREIGN KEY (PRODUCT\_CODE) REFERENCES PRODUCT(PRODUCT\_CODE)

);

INSERT INTO COUNTRY VALUES ('CY01', 'INDIA', 'SOUTHERN');

INSERT INTO COUNTRY VALUES ('CY02', 'FRANCE', 'EUROPEAN');

INSERT INTO COUNTRY VALUES ('CY03', 'GERMANY', 'EUROPEAN');

INSERT INTO COUNTRY VALUES ('CY04', 'PAKISTAN', 'SOUTHERN');

INSERT INTO COUNTRY VALUES ('CY05', 'NETHERLANDS', 'EUROPEAN');

INSERT INTO CATEGORY VALUES ('CT01', 'COMPUTER PERIPHERALS');

INSERT INTO CATEGORY VALUES ('CT02', 'MOBILE');

INSERT INTO CATEGORY VALUES ('CT03', 'HOUSEHOLD APPLIANCES');

INSERT INTO CATEGORY VALUES ('CT04', 'ELECTRONIC GADGETS');

INSERT INTO CUSTOMER VALUES ('CU01', 'VIVEK&SONS', 2000000.00);

INSERT INTO CUSTOMER VALUES ('CU02', 'CITIZEN', 100000.00);

INSERT INTO CUSTOMER VALUES ('CU03', 'CHAND CO', 500000.00);

INSERT INTO CUSTOMER VALUES ('CU04', 'SPENCER', 1000000.00);

INSERT INTO CUSTOMER VALUES ('CU05', 'RAMCO RETAILS', 400000.00);

INSERT INTO BRANCH VALUES ('BR01', 'CHENNAI', 'CY01');

INSERT INTO BRANCH VALUES ('BR02', 'PARIS', 'CY02');

INSERT INTO BRANCH VALUES ('BR03', 'HAMBURG', 'CY03');

INSERT INTO SALESREP VALUES ('SP01', 'PRIYA', '2005-03-22', 'SP03', 'BR01', 25000.00);

INSERT INTO SALESREP VALUES ('SP02', 'MADHAN', '2003-05-12', 'SP03', 'BR01', 35000.00);

INSERT INTO SALESREP VALUES ('SP03', 'ACHU', '2000-06-04', NULL, 'BR02', 50000.00);

INSERT INTO PRODUCT VALUES ('PR01', 'INTEL PROCESSOR', 'CT01', 10000.00, 150);

INSERT INTO PRODUCT VALUES ('PR02', 'HP MONITOR', 'CT01', 15000.00, 200);

INSERT INTO PRODUCT VALUES ('PR03', 'IBM MONITOR', 'CT01', 12000.00, 200);

INSERT INTO ORDER\_MASTER VALUES ('OR01', '2008-03-06', 'SP01', 'CU01');

INSERT INTO ORDER\_MASTER VALUES ('OR02', '2009-04-12', 'SP01', 'CU02');

INSERT INTO ORDER\_MASTER VALUES ('OR03', '2010-03-16', 'SP02', 'CU01');

INSERT INTO ORDER\_DETAIL VALUES ('OR01', 'PR01', 20);

INSERT INTO ORDER\_DETAIL VALUES ('OR01', 'PR02', 10);

INSERT INTO ORDER\_DETAIL VALUES ('OR01', 'PR03', 20);

INSERT INTO ORDER\_DETAIL VALUES ('OR02', 'PR01', 10);

INSERT INTO ORDER\_DETAIL VALUES ('OR03', 'PR04', 50);

-- 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

);

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

-- All categories

SELECT category\_code

FROM category

EXCEPT

-- Categories that have products

SELECT DISTINCT category\_code

FROM product;

-- Method 7: Using INTERSECT (SET operators)

-- All categories

SELECT category\_code

FROM category

INTERSECT

-- Categories not in the product table

SELECT c.category\_code

FROM category c

WHERE NOT EXISTS (

SELECT 1

FROM product p

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

);

\*/

-- 4. Which is the costliest product?

-- Method 1: Using RANK function

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

FROM (

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

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

FROM product

) AS ranked\_products

WHERE rnk = 1;

-- Method 2: 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

);

-- Method 3: Using Subquery

SELECT product\_code, prod\_desc, price

FROM product

WHERE price = (

SELECT MAX(price)

FROM product

);

-- Method 4: 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;

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

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

FROM product AS p1

WHERE NOT EXISTS (

SELECT 1

FROM product AS p2

WHERE p2.price > p1.price

);

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

-- Get products with maximum price

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

FROM product p

WHERE p.price = (

SELECT MAX(price)

FROM product

)

UNION ALL

-- Also include products not dominated by others (handles NULL prices)

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

FROM product p

WHERE NOT EXISTS (

SELECT 1

FROM product p2

WHERE p2.price > p.price

)

AND p.price IS NOT NULL;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

-- All products

SELECT product\_code, prod\_desc, price

FROM product

EXCEPT

-- Products that have any other product with higher price

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

FROM product p1

JOIN product p2

ON p2.price > p1.price;

-- Method 8: Using INTERSECT (SET operators)

-- All products

SELECT product\_code, prod\_desc, price

FROM product

INTERSECT

-- Products with maximum price

SELECT product\_code, prod\_desc, price

FROM product

WHERE price = (

SELECT MAX(price)

FROM product

);

\*/

-- 5. Which category has costliest product?

-- Method 1: Using RANK function

SELECT category\_code

FROM (

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

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

FROM product

) AS ranked\_products

WHERE rnk = 1;

-- Method 2: 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

);

-- Method 3: Using 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

);

-- Method 4: 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;

-- Method 5: Using NOT EXISTS (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

);

-- Method 6: Using UNION (SET operators)

-- Direct MAX approach

SELECT c.category\_code, MAX(p.price) AS price

FROM category c

JOIN product p

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

GROUP BY c.category\_code

HAVING MAX(p.price) = (

SELECT MAX(price)

FROM product

)

UNION

-- NOT EXISTS approach

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

);

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

-- All category-price combinations

SELECT c.category\_code, p.price

FROM category c

JOIN product p

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

EXCEPT

-- Categories with products that have cheaper alternatives

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;

-- Method 8: Using INTERSECT (SET operators)

-- All category-price combinations

SELECT c.category\_code, p.price

FROM category c

JOIN product p

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

INTERSECT

-- Only those with maximum price

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

);

\*/

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

-- Method 1: Using RANK function

SELECT category\_code, total\_quantity

FROM (

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

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

FROM product

GROUP BY category\_code

) AS ranked\_products

WHERE rnk = 1;

-- Method 2: 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

);

-- Method 3: Using HAVING

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

);

-- Method 4: 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;

-- Method 5: Using NOT EXISTS (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

);

-- Method 6: Using UNION (SET operators)

-- Direct MAX approach

SELECT c.category\_code, SUM(p.qty\_on\_hand) AS total\_quantity

FROM category c

JOIN product p

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

GROUP BY c.category\_code

HAVING SUM(p.qty\_on\_hand) = (

SELECT MAX(total) FROM (

SELECT SUM(qty\_on\_hand) AS total

FROM product

GROUP BY category\_code

) sums

)

UNION

-- NOT EXISTS approach

SELECT c.category\_code, SUM(p.qty\_on\_hand) AS total\_quantity

FROM category c

JOIN product p

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

WHERE NOT EXISTS (

SELECT 1

FROM category c2

JOIN product p2

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

GROUP BY c2.category\_code

HAVING SUM(p2.qty\_on\_hand) > SUM(p.qty\_on\_hand)

)

GROUP BY c.category\_code;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

-- All categories with their total quantities

SELECT c.category\_code, SUM(p.qty\_on\_hand) AS total\_quantity

FROM category c

JOIN product p

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

GROUP BY c.category\_code

EXCEPT

-- Remove categories that have less than some other category

SELECT c1.category\_code, SUM(p1.qty\_on\_hand)

FROM category c1

JOIN product p1

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

GROUP BY c1.category\_code

JOIN (

SELECT c2.category\_code, SUM(p2.qty\_on\_hand) AS cat\_sum

FROM category c2

JOIN product p2

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

GROUP BY c2.category\_code

) c2

ON c2.cat\_sum > SUM(p1.qty\_on\_hand);

-- Method 8: Using INTERSECT (SET operators)

-- All categories with their quantities

SELECT c.category\_code, SUM(p.qty\_on\_hand) AS total\_quantity

FROM category c

JOIN product p

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

GROUP BY c.category\_code

INTERSECT

-- Only those matching the maximum quantity

SELECT c.category\_code, SUM(p.qty\_on\_hand)

FROM category c

JOIN product p

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

GROUP BY c.category\_code

HAVING SUM(p.qty\_on\_hand) = (

SELECT MAX(total) FROM (

SELECT SUM(qty\_on\_hand) AS total

FROM product

GROUP BY category\_code

) sums

);

\*/

-- 7. Category wise display the cosliest product

-- Method 1: Using RANK function

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

FROM (

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

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

FROM product

) AS ranked\_products

WHERE rnk = 1;

-- Method 2: Using Subquery

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

FROM product a

WHERE price = (

SELECT MAX(price)

FROM product b

WHERE b.category\_code = a.category\_code

);

-- Method 3: 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

);

-- Method 4: 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;

-- Method 5: Using NOT EXISTS (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

);

-- Method 6: Using UNION (SET operators)

-- Direct MAX approach

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

FROM category c

JOIN product p

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

WHERE p.price = (

SELECT MAX(price)

FROM product p2

WHERE p2.category\_code = c.category\_code

)

UNION

-- NOT EXISTS approach

SELECT c.category\_code, p.product\_code, p.prod\_desc, 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.category\_code = p.category\_code AND p2.price > p.price

);

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

-- All category-product combinations

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

FROM category c

JOIN product p

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

EXCEPT

-- Remove products that aren't the costliest in their category

SELECT c1.category\_code, p1.product\_code, p1.prod\_desc, p1.price

FROM category c1

JOIN product p1

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

JOIN product p2

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

-- Method 8: Using INTERSECT (SET operators)

-- All products

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

FROM category c

JOIN product p

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

INTERSECT

-- Only products that rank highest in their category

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

FROM (

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

DENSE\_RANK() OVER (PARTITION BY c.category\_code ORDER BY p.price DESC) AS price\_rank

FROM category c

JOIN product p

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

) ranked

WHERE price\_rank = 1;

\*/

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

-- Method 1: Using RANK function

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

FROM (

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

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

FROM product

) AS ranked\_products

WHERE rnk = 1;

-- Method 2: 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

);

-- Method 3: Using 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

);

-- Method 4: 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;

-- Method 5: Using NOT EXISTS (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

);

-- Method 6: Using UNION (SET operators)

-- Direct MIN approach

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

FROM category c

JOIN product p

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

WHERE p.qty\_on\_hand = (

SELECT MIN(qty\_on\_hand)

FROM product p2

WHERE p2.category\_code = c.category\_code

)

UNION

-- NOT EXISTS approach

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

FROM category c

JOIN product p

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

WHERE NOT EXISTS (

SELECT 1

FROM product p2

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

);

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

-- All category-product-quantity combinations

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

FROM category c

JOIN product p

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

EXCEPT

-- Remove products that have higher quantities than others in same category

SELECT c1.category\_code, p1.product\_code, p1.prod\_desc, p1.qty\_on\_hand

FROM category c1

JOIN product p1

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

JOIN product p2

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

-- Method 8: Using INTERSECT (SET operators)

-- All products

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

FROM category c

JOIN product p

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

INTERSECT

-- Only products with minimum quantity in their category

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

FROM (

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

DENSE\_RANK() OVER (PARTITION BY c.category\_code ORDER BY p.qty\_on\_hand ASC) AS qty\_rank

FROM category c

JOIN product p

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

) ranked

WHERE qty\_rank = 1;

\*/

-- 9. Which order has maximum products?

-- Method 1: Using RANK function

SELECT order\_code, product\_count

FROM (

SELECT order\_code, COUNT(product\_code) AS product\_count,

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

FROM order\_detail

GROUP BY order\_code

) AS ranked\_products

WHERE rnk = 1;

-- Method 2: 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

);

-- Method 3: Using HAVING

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

);

-- Method 4: 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;

-- Method 5: Using NOT EXISTS (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

);

-- Method 6: Using UNION (SET operators)

-- Direct MAX approach

SELECT order\_code, COUNT(DISTINCT product\_code) AS product\_count

FROM order\_detail

GROUP BY order\_code

HAVING COUNT(DISTINCT product\_code) = (

SELECT MAX(pcount)

FROM (

SELECT COUNT(DISTINCT product\_code) AS pcount

FROM order\_detail

GROUP BY order\_code

) max\_counts

)

UNION

-- NOT EXISTS approach

SELECT o1.order\_code, COUNT(DISTINCT o1.product\_code) AS product\_count

FROM order\_detail o1

WHERE NOT EXISTS (

SELECT 1

FROM order\_detail o2

GROUP BY o2.order\_code

HAVING COUNT(DISTINCT o2.product\_code) > COUNT(DISTINCT o1.product\_code)

)

GROUP BY o1.order\_code;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

-- All orders with their product counts

SELECT order\_code, COUNT(DISTINCT product\_code) AS product\_count

FROM order\_detail

GROUP BY order\_code

EXCEPT

-- Remove orders that have fewer products than some other order

SELECT o1.order\_code, COUNT(DISTINCT o1.product\_code)

FROM order\_detail o1

GROUP BY o1.order\_code

JOIN (

SELECT o2.order\_code, COUNT(DISTINCT o2.product\_code) AS order\_count

FROM order\_detail o2

GROUP BY o2.order\_code

) o2

ON o2.order\_count > COUNT(DISTINCT o1.product\_code);

-- Method 8: Using INTERSECT (SET operators)

-- All orders with their product counts

SELECT order\_code, COUNT(DISTINCT product\_code) AS product\_count

FROM order\_detail

GROUP BY order\_code

INTERSECT

-- Only orders with the maximum count

SELECT order\_code, COUNT(DISTINCT product\_code)

FROM order\_detail

GROUP BY order\_code

HAVING COUNT(DISTINCT product\_code) = (

SELECT MAX(pcount)

FROM (

SELECT COUNT(DISTINCT product\_code) AS pcount

FROM order\_detail

GROUP BY order\_code

) max\_counts

);

\*/

-- 10. Which is the frequently ordered product?

-- Method 1: Using RANK function

SELECT product\_code, order\_count

FROM (

SELECT product\_code, COUNT(order\_code) AS order\_count,

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

FROM order\_detail

GROUP BY product\_code

) AS ranked\_products

WHERE rnk = 1;

-- Method 2: 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

);

-- Method 3: Using 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

);

-- Method 4: 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;

-- Method 5: Using NOT EXISTS (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

);

-- Method 6: Using UNION (SET operators)

-- Direct MAX approach

SELECT product\_code, COUNT(order\_code) AS order\_count

FROM order\_detail

GROUP BY product\_code

HAVING COUNT(order\_code) = (

SELECT MAX(ocount) FROM (

SELECT COUNT(order\_code) AS ocount

FROM order\_detail

GROUP BY product\_code

) max\_counts

)

UNION

-- NOT EXISTS approach

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

FROM order\_detail p1

WHERE NOT EXISTS (

SELECT 1

FROM order\_detail p2

GROUP BY p2.product\_code

HAVING COUNT(p2.order\_code) > COUNT(p1.order\_code)

)

GROUP BY p1.product\_code;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

-- All products with their order counts

SELECT product\_code, COUNT(order\_code) AS order\_count

FROM order\_detail

GROUP BY product\_code

EXCEPT

-- Remove products that are ordered less than some other product

SELECT p1.product\_code, COUNT(p1.order\_code)

FROM order\_detail p1

GROUP BY p1.product\_code

JOIN (

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

FROM order\_detail p2

GROUP BY p2.product\_code

) p2

ON p2.product\_count > COUNT(p1.order\_code);

-- Method 8: Using INTERSECT (SET operators)

-- All products with their order counts

SELECT product\_code, COUNT(order\_code) AS order\_count

FROM order\_detail

GROUP BY product\_code

INTERSECT

-- Only products with the maximum order count

SELECT product\_code, COUNT(order\_code)

FROM order\_detail

GROUP BY product\_code

HAVING COUNT(order\_code) = (

SELECT MAX(ocount)

FROM (

SELECT COUNT(order\_code) AS ocount

FROM order\_detail

GROUP BY product\_code

) max\_counts

);

\*/

-- 11. Which product is least ordered product?

-- Method 1: Using RANK function

SELECT product\_code, order\_count

FROM (

SELECT product\_code, COUNT(order\_code) AS order\_count,

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

FROM order\_detail

GROUP BY product\_code

) AS ranked\_products

WHERE rnk = 1;

-- Method 2: 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

);

-- Method 3: 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

);

-- Method 4: 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;

-- Method 5: Using NOT EXISTS (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

);

-- Method 6: Using UNION (SET operators)

-- Direct MIN approach

SELECT product\_code, COUNT(order\_code) AS order\_count

FROM order\_detail

GROUP BY product\_code

HAVING COUNT(order\_code) = (

SELECT MIN(ocount) FROM (

SELECT COUNT(order\_code) AS ocount

FROM order\_detail

GROUP BY product\_code

) min\_counts

)

UNION

-- Include never-ordered products

SELECT p.product\_code, 0

FROM product p

WHERE NOT EXISTS (

SELECT 1 FROM order\_detail od

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

);

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

-- All products with their order counts

SELECT product\_code, COUNT(order\_code) AS order\_count

FROM order\_detail

GROUP BY product\_code

EXCEPT

-- Remove products that have been ordered more than some other product

SELECT p1.product\_code, COUNT(p1.order\_code)

FROM order\_detail p1

GROUP BY p1.product\_code

JOIN (

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

FROM order\_detail p2

GROUP BY p2.product\_code

) p2

ON p2.product\_count < COUNT(p1.order\_code);

-- Method 8: Using INTERSECT (SET operators)

-- All products with their order counts

SELECT product\_code, COUNT(order\_code) AS order\_count

FROM order\_detail

GROUP BY product\_code

INTERSECT

-- Only products with the minimum order count

SELECT product\_code, COUNT(order\_code)

FROM order\_detail

GROUP BY product\_code

HAVING COUNT(order\_code) = (

SELECT MIN(ocount)

FROM (

SELECT COUNT(order\_code) AS ocount

FROM order\_detail

GROUP BY product\_code

) min\_counts

);

\*/

-- 12. What product is not at all ordered?

-- Method 1: Using LEFT JOIN

SELECT product.product\_code

FROM product

LEFT JOIN order\_detail

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

WHERE order\_detail.product\_code IS NULL;

-- Method 2: Using NOT IN

SELECT product\_code, prod\_desc

FROM product

WHERE product\_code NOT IN (

SELECT DISTINCT product\_code

FROM order\_detail

);

-- Method 3: Using HAVING

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

FROM product p

LEFT JOIN order\_detail od ON p.product\_code = od.product\_code

GROUP BY p.product\_code, p.prod\_desc

HAVING COUNT(od.order\_code) = 0;

-- Method 4: Using NOT EXISTS (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

);

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

-- Products with zero orders (explicit count)

SELECT p.product\_code

FROM product p

LEFT JOIN order\_detail od ON p.product\_code = od.product\_code

GROUP BY p.product\_code

HAVING COUNT(od.order\_code) = 0

UNION ALL

-- Alternative method to catch any edge cases

SELECT p.product\_code

FROM product p

WHERE p.product\_code NOT IN (

SELECT DISTINCT product\_code

FROM order\_detail

);

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

-- All products

SELECT product\_code

FROM product

EXCEPT

-- Products that appear in orders

SELECT DISTINCT product\_code

FROM order\_detail;

-- Method 7: Using INTERSECT (SET operators)

-- All products

SELECT product\_code FROM product

INTERSECT

-- Products with no matching orders

SELECT p.product\_code

FROM product p

LEFT JOIN order\_detail od

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

WHERE od.product\_code IS NULL;

\*/

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

-- Method 1: Using RANK function

SELECT order\_code, total\_cost

FROM (

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

RANK() OVER(ORDER BY SUM(product.price \* order\_detail.qty\_ordered) DESC) AS rnk

FROM order\_detail

JOIN product

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

GROUP BY order\_code

) AS ranked

WHERE rnk = 1;

-- Method 2: Using NOT IN

-- Find orders that are NOT less expensive than some other order

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

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) NOT IN (

-- All order totals that are less than the maximum

SELECT total

FROM (

SELECT SUM(p2.price \* o2.qty\_ordered) AS total

FROM order\_detail o2

JOIN product p2

ON o2.product\_code = p2.product\_code

GROUP BY o2.order\_code

) order\_totals

WHERE total < (

SELECT MAX(total)

FROM (

SELECT SUM(p3.price \* o3.qty\_ordered) AS total

FROM order\_detail o3

JOIN product p3

ON o3.product\_code = p3.product\_code

GROUP BY o3.order\_code

) max\_totals

)

);

-- Method 3: Using HAVING

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

);

-- Method 4: 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;

-- Method 5: Using NOT EXISTS (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

);

-- Method 6: Using UNION (SET operators)

-- Direct MAX approach

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

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(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

) order\_totals

)

UNION

-- NOT EXISTS approach

SELECT o1.order\_code, SUM(p1.price \* o1.qty\_ordered) AS total\_cost

FROM order\_detail o1

JOIN product p1

ON o1.product\_code = p1.product\_code

WHERE NOT EXISTS (

SELECT 1

FROM order\_detail o2

JOIN product p2

ON o2.product\_code = p2.product\_code

GROUP BY o2.order\_code

HAVING SUM(p2.price \* o2.qty\_ordered) > SUM(p1.price \* o1.qty\_ordered)

)

GROUP BY o1.order\_code;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

-- All orders with their total costs

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

FROM order\_detail o

JOIN product p

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

GROUP BY o.order\_code

EXCEPT

-- Remove orders that cost less than some other order

SELECT o1.order\_code, SUM(p1.price \* o1.qty\_ordered)

FROM order\_detail o1

JOIN product p1

ON o1.product\_code = p1.product\_code

GROUP BY o1.order\_code

JOIN (

SELECT o2.order\_code, SUM(p2.price \* o2.qty\_ordered) AS order\_cost

FROM order\_detail o2

JOIN product p2

ON o2.product\_code = p2.product\_code

GROUP BY o2.order\_code

) o2

ON o2.order\_cost > SUM(p1.price \* o1.qty\_ordered);

-- Method 8: Using INTERSECT (SET operators)

-- All orders with their total costs

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

FROM order\_detail o

JOIN product p

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

GROUP BY o.order\_code

INTERSECT

-- Only orders with the maximum total cost

SELECT o.order\_code, 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(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

) order\_totals

);

\*/

-- 14. In which date is the costliest order made?

-- Method 1: Using RANK function

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

FROM (

SELECT order\_detail.order\_code, order\_master.order\_date, SUM(product.price \* order\_detail.qty\_ordered) AS total\_cost,

RANK() OVER(ORDER BY order\_detail.order\_code, SUM(product.price \* order\_detail.qty\_ordered) DESC) AS rnk

FROM order\_detail

JOIN product

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

JOIN order\_master

ON order\_detail.order\_code = order\_master.order\_code

GROUP BY order\_detail.order\_code, order\_master.order\_date

) AS ranked

WHERE rnk = 1;

-- Method 2: 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

);

-- Method 3: Using HAVING

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

);

-- Method 4: 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;

-- Method 5: Using NOT EXISTS (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

);

-- Method 6: Using UNION (SET operators)

-- Direct MAX approach

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

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

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

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

SELECT MAX(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

) order\_totals

)

UNION

-- NOT EXISTS approach

SELECT o1.order\_code, om1.order\_date, SUM(p1.price \* o1.qty\_ordered) AS total\_cost

FROM order\_detail o1

JOIN product p1

ON o1.product\_code = p1.product\_code

JOIN order\_master om1

ON o1.order\_code = om1.order\_code

WHERE NOT EXISTS (

SELECT 1

FROM order\_detail o2

JOIN product p2 ON o2.product\_code = p2.product\_code

GROUP BY o2.order\_code

HAVING SUM(p2.price \* o2.qty\_ordered) > SUM(p1.price \* o1.qty\_ordered)

)

GROUP BY o1.order\_code, om1.order\_date;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

-- All orders with their dates and total costs

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

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

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

EXCEPT

-- Remove orders that cost less than some other order

SELECT o1.order\_code, om1.order\_date, SUM(p1.price \* o1.qty\_ordered)

FROM order\_detail o1

JOIN product p1

ON o1.product\_code = p1.product\_code

JOIN order\_master om1

ON o1.order\_code = om1.order\_code

GROUP BY o1.order\_code, om1.order\_date

JOIN (

SELECT o2.order\_code, SUM(p2.price \* o2.qty\_ordered) AS order\_cost

FROM order\_detail o2

JOIN product p2

ON o2.product\_code = p2.product\_code

GROUP BY o2.order\_code

) o2

ON o2.order\_cost > SUM(p1.price \* o1.qty\_ordered);

-- Method 8: Using INTERSECT (SET operators)

-- All orders with their dates and costs

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

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

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

INTERSECT

-- Only orders with the maximum total cost

SELECT o.order\_code, om.order\_date, SUM(p.price \* o.qty\_ordered)

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

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

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

SELECT MAX(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

) order\_totals

);

\*/

-- 15. Which customer made the costliest order?

-- Method 1: 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

GROUP BY om.customer\_code, c.customer\_name

) AS ranked

WHERE rnk = 1;

-- Method 2: 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

)

);

-- Method 3: Using HAVING

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

);

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

/\* Step 1: pre-aggregate what each customer has spent in total \*/

SELECT t.customer\_code,

t.customer\_name,

t.total\_cost

FROM (

SELECT om.customer\_code, c.customer\_name, SUM(p.price \* od.qty\_ordered) AS total\_cost

FROM order\_master om

JOIN customer c

ON c.customer\_code = om.customer\_code

JOIN order\_detail od

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

JOIN product p

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

GROUP BY om.customer\_code, c.customer\_name

) AS t

/\* Step 2: keep only those rows for which no \*other\* customer total is greater \*/

WHERE NOT EXISTS (

SELECT 1

FROM (

SELECT om2.customer\_code, SUM(p2.price \* od2.qty\_ordered) AS cust\_total

FROM order\_master om2

JOIN order\_detail od2

ON od2.order\_code = om2.order\_code

JOIN product p2

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

GROUP BY om2.customer\_code

) AS u

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

);

-- Method 6: Using UNION (SET operators)

-- Customers with orders matching the maximum order total

SELECT c.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

GROUP BY c.customer\_code, c.customer\_name

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

SELECT MAX(order\_total)

FROM (

SELECT SUM(p.price \* od.qty\_ordered) AS order\_total

FROM order\_detail od

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

GROUP BY od.order\_code

) order\_totals

)

UNION

-- Alternative approach using NOT EXISTS

SELECT c.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 NOT EXISTS (

SELECT 1

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

GROUP BY om2.order\_code

HAVING SUM(p2.price \* od2.qty\_ordered) > SUM(p.price \* od.qty\_ordered)

)

AND EXISTS (

SELECT 1 FROM order\_detail od3

WHERE od3.order\_code = od.order\_code

)

GROUP BY c.customer\_code, c.customer\_name;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

-- All customers with their maximum order totals

SELECT c.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

GROUP BY c.customer\_code, c.customer\_name

EXCEPT

-- Remove customers whose maximum order is less than someone else's

SELECT c1.customer\_code, c1.customer\_name, SUM(p1.price \* od1.qty\_ordered)

FROM order\_detail od1

JOIN product p1

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

JOIN order\_master om1

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

JOIN customer c1

ON om1.customer\_code = c1.customer\_code

GROUP BY c1.customer\_code, c1.customer\_name

HAVING SUM(p1.price \* od1.qty\_ordered) < (

SELECT MAX(order\_total) FROM (

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

FROM order\_detail od2

JOIN product p2

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

GROUP BY od2.order\_code

) order\_totals

);

-- Method 8: Using INTERSECT (SET operators)

-- All customers with their order totals

SELECT c.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

GROUP BY c.customer\_code, c.customer\_name

INTERSECT

-- Only customers who made orders matching the maximum total

SELECT c.customer\_code, c.customer\_name, 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

JOIN customer c

ON om.customer\_code = c.customer\_code

GROUP BY c.customer\_code, c.customer\_name

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

SELECT MAX(order\_total) FROM (

SELECT SUM(p.price \* od.qty\_ordered) AS order\_total

FROM order\_detail od

JOIN product p

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

GROUP BY od.order\_code

) order\_totals

);

\*/

-- 16. Which customer made the costliest order today?

-- Method 1: Using RANK function

SELECT customer\_code, customer\_name, total\_cost

FROM (

SELECT order\_master.customer\_code, customer.customer\_name, SUM(product.price \* order\_detail.qty\_ordered) AS total\_cost,

RANK() OVER(ORDER BY SUM(product.price \* order\_detail.qty\_ordered) DESC) AS rnk

FROM order\_detail

JOIN product

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

JOIN order\_master

ON order\_detail.order\_code = order\_master.order\_code

JOIN customer

ON order\_master.customer\_code = customer.customer\_code

WHERE order\_master.order\_date = curdate()

GROUP BY order\_master.customer\_code, customer.customer\_name

) AS ranked

WHERE rnk = 1;

-- Method 2: 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

)

);

-- Method 3: Using HAVING

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

);

-- Method 4: Using ALL

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

);

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

SELECT c.customer\_code, c.customer\_name, om.order\_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

JOIN customer c

ON om.customer\_code = c.customer\_code

WHERE om.order\_date = '2025-05-24'

GROUP BY c.customer\_code, c.customer\_name, om.order\_code

HAVING NOT EXISTS (

SELECT 1

FROM order\_master om2

JOIN order\_detail od2

ON om2.order\_code = od2.order\_code

JOIN product p2

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

WHERE om2.order\_date = '2025-05-24'

GROUP BY om2.order\_code

HAVING SUM(p2.price \* od2.qty\_ordered) > SUM(p.price \* od.qty\_ordered)

);

-- Method 6: Using UNION (SET operators)

SELECT om.customer\_code, c.customer\_name, om.order\_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

JOIN customer c

ON om.customer\_code = c.customer\_code

WHERE om.order\_date = '2025-05-24'

GROUP BY om.customer\_code, c.customer\_name, om.order\_code

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

SELECT MAX(order\_total)

FROM (

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

FROM order\_master om2

JOIN order\_detail od2

ON om2.order\_code = od2.order\_code

JOIN product p2

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

WHERE om2.order\_date = '2025-05-24'

GROUP BY om2.order\_code

) AS daily\_totals

)

UNION

SELECT NULL, NULL, NULL, NULL;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

-- All orders today minus those that are not the max

SELECT om.customer\_code, c.customer\_name, om.order\_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

JOIN customer c

ON om.customer\_code = c.customer\_code

WHERE om.order\_date = '2025-05-24'

GROUP BY om.customer\_code, c.customer\_name, om.order\_code

EXCEPT

SELECT om.customer\_code, c.customer\_name, om.order\_code, 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

JOIN customer c

ON om.customer\_code = c.customer\_code

WHERE om.order\_date = '2025-05-24'

GROUP BY om.customer\_code, c.customer\_name, om.order\_code

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

SELECT MAX(order\_total)

FROM (

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

FROM order\_master om2

JOIN order\_detail od2

ON om2.order\_code = od2.order\_code

JOIN product p2

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

WHERE om2.order\_date = '2025-05-24'

GROUP BY om2.order\_code

) AS max\_total

);

-- Method 8: Using INTERSECT (SET operators)

-- Get costliest total first

SELECT om.customer\_code, c.customer\_name, om.order\_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

JOIN customer c

ON om.customer\_code = c.customer\_code

WHERE om.order\_date = '2025-05-24'

GROUP BY om.customer\_code, c.customer\_name, om.order\_code

INTERSECT

SELECT om.customer\_code, c.customer\_name, om.order\_code, max\_order.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

JOIN customer c

ON om.customer\_code = c.customer\_code

JOIN (

SELECT om2.order\_code, SUM(p2.price \* od2.qty\_ordered) AS total\_cost

FROM order\_master om2

JOIN order\_detail od2

ON om2.order\_code = od2.order\_code

JOIN product p2

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

WHERE om2.order\_date = '2025-05-24'

GROUP BY om2.order\_code

HAVING SUM(p2.price \* od2.qty\_ordered) = (

SELECT MAX(SUM(p3.price \* od3.qty\_ordered))

FROM order\_master om3

JOIN order\_detail od3

ON om3.order\_code = od3.order\_code

JOIN product p3

ON od3.product\_code = p3.product\_code

WHERE om3.order\_date = '2025-05-24'

GROUP BY om3.order\_code

)

) AS max\_order

ON om.order\_code = max\_order.order\_code;

\*/

-- 17. Generate the report like: Customer name | Cust\_addr | Order code | Order date | No\_of\_products\_ordered | Total Bill

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS generateCustomerOrderReport;

DELIMITER $$

CREATE PROCEDURE generateCustomerOrderReport()

BEGIN

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

ORDER BY om.order\_date;

END $$

DELIMITER ;

CALL generateCustomerOrderReport();

-- Method 2: Using JOIN

SELECT customer.customer\_name, order\_master.order\_code, order\_master.order\_date, COUNT(product.product\_code) AS No\_of\_products\_ordered, SUM(product.price \* order\_detail.qty\_ordered) AS total\_bill

FROM customer

JOIN order\_master

ON customer.customer\_code = order\_master.customer\_code

JOIN order\_detail

ON order\_master.order\_code = order\_detail.order\_code

JOIN product

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

GROUP BY customer.customer\_name, order\_master.order\_code, order\_master.order\_date;

-- Method 3: 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;

-- Method 4: Using 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;

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

SELECT DISTINCT c.customer\_code, c.customer\_name

FROM customer c

JOIN order\_master om ON c.customer\_code = om.customer\_code

WHERE om.order\_date = CURDATE() AND NOT EXISTS (

SELECT 1

FROM order\_master om2

JOIN order\_detail od2

ON om2.order\_code = od2.order\_code

JOIN product p2

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

WHERE om2.order\_date = CURDATE()

GROUP BY om2.order\_code

HAVING SUM(p2.price \* od2.qty\_ordered) > (

SELECT SUM(p.price \* od.qty\_ordered)

FROM order\_detail od

JOIN product p

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

WHERE od.order\_code = om.order\_code

GROUP BY od.order\_code

)

);

-- Method 6: Using UNION (SET operators)

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

);

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

-- First, get all customers who made orders today

SELECT c.customer\_code, c.customer\_name

FROM customer c

JOIN order\_master om

ON c.customer\_code = om.customer\_code

WHERE om.order\_date = CURDATE()

EXCEPT

-- Then exclude those who made orders with cost less than max

SELECT c.customer\_code, c.customer\_name

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\_date = CURDATE()

GROUP BY om.order\_code, c.customer\_code, c.customer\_name

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

SELECT MAX(total)

FROM (

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

FROM order\_master om2

JOIN order\_detail od2

ON om2.order\_code = od2.order\_code

JOIN product p2

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

WHERE om2.order\_date = CURDATE()

GROUP BY om2.order\_code

) AS totals

);

-- Method 7: Using INTERSECT (SET operators)

-- Customers with orders today

SELECT c.customer\_code, c.customer\_name

FROM customer c

JOIN order\_master om ON c.customer\_code = om.customer\_code

WHERE om.order\_date = CURDATE()

INTERSECT

-- Customers who made orders equal to the max order total today

SELECT c.customer\_code, c.customer\_name

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\_date = CURDATE()

GROUP BY om.order\_code, c.customer\_code, c.customer\_name

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

SELECT MAX(total)

FROM (

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

FROM order\_master om2

JOIN order\_detail od2

ON om2.order\_code = od2.order\_code

JOIN product p2

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

WHERE om2.order\_date = CURDATE()

GROUP BY om2.order\_code

) AS totals

);

\*/

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

-- Method 1: Using JOIN

SELECT customer.customer\_code, customer.customer\_name, order\_master.order\_code, order\_master.order\_date, SUM(product.price \* order\_detail.qty\_ordered) AS total\_cost

FROM customer

JOIN order\_master

ON customer.customer\_code = order\_master.customer\_code

JOIN order\_detail

ON order\_master.order\_code = order\_detail.order\_code

JOIN product

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

GROUP BY customer.customer\_code,customer.customer\_name, order\_master.order\_code, order\_master.order\_date

HAVING total\_cost >= 5000;

-- Method 2: Using NOT IN

SELECT customer\_code, customer\_name, order\_code, order\_date, total\_bill

FROM (

SELECT c.customer\_code, 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

);

-- Method 3: Using NOT EXISTS (Correlated Subquery)

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 NOT EXISTS (

SELECT 1

FROM order\_detail od2

JOIN product p2

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

WHERE od2.order\_code = om.order\_code

GROUP BY od2.order\_code

HAVING SUM(p2.price \* od2.qty\_ordered) < 5000

);

-- Method 4: Using UNION (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

HAVING total\_bill >= 5000

UNION

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 union\_all\_bills;

/\* Does not work in MySQL

-- Method 5: Using EXCEPT (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

EXCEPT

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

) bills\_less\_than\_5000;

-- Method 6: Using INTERSECT (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

) bills\_gte\_5000;

\*/

-- 19. Which country has maximum branches?

-- Method 1: 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;

-- Method 2: Using HAVING

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

);

-- Method 3: 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)

);

-- Method 4: 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;

-- Method 5: Using Subquery

SELECT cn.country\_name

FROM country cn

WHERE (

SELECT COUNT(b.branch\_code)

FROM branch b

WHERE b.country\_code = cn.country\_code

) >= ALL (

SELECT COUNT(b2.branch\_code)

FROM branch b2

GROUP BY b2.country\_code

);

-- Method 6: Using NOT EXISTS (Correlated 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 NOT EXISTS (

SELECT 1

FROM country c2

JOIN branch b2

ON b2.country\_code = c2.country\_code

GROUP BY c2.country\_name

HAVING COUNT(b2.branch\_code) > COUNT(b.branch\_code)

);

-- Method 7: Using UNION (SET operators)

/\* step-A: max-value derived once \*/

WITH max\_val AS (

SELECT MAX(br\_cnt) AS mx

FROM (

SELECT COUNT(branch\_code) AS br\_cnt

FROM branch

GROUP BY country\_code

) t

)

/\* step-B: union left side — only the rows matching that max \*/

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 mx FROM max\_val)

UNION

/\* union right side identical – shown only to illustrate UNION usage \*/

SELECT cn.country\_name, COUNT(b.branch\_code)

FROM country cn

JOIN branch b

ON b.country\_code = cn.country\_code

GROUP BY cn.country\_name

HAVING COUNT(b.branch\_code) = (SELECT mx FROM max\_val);

/\* Does not work in MySQL

-- Method 8: Using EXCEPT (SET operators)

SELECT country\_name, branch\_count

FROM (

SELECT cn.country\_name, COUNT(b.branch\_code) AS branch\_count

FROM country cn

JOIN branch b

ON b.country\_code = cn.country\_code

GROUP BY cn.country\_name

) AS all\_counts

EXCEPT

SELECT cn.country\_name, COUNT(b.branch\_code)

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(branch\_code) AS cnt

FROM branch

GROUP BY country\_code

) t

);

-- Method 9: Using INTERSECT (SET operators)

SELECT cn.country\_name, COUNT(b.branch\_code) AS branch\_count

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(branch\_code) AS cnt

FROM branch

GROUP BY country\_code

) t

)

INTERSECT

SELECT cn.country\_name, COUNT(b.branch\_code)

FROM country cn

JOIN branch b

ON b.country\_code = cn.country\_code

GROUP BY cn.country\_name;

\*/

-- 20. Which country has minimum branches?

-- Method 1: 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;

-- Method 2: 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)

);

-- Method 3: Using 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

);

-- Method 4: 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

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

GROUP BY cn.country\_name;

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

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 NOT EXISTS (

SELECT 1

FROM country c2

LEFT JOIN branch b2

ON b2.country\_code = c2.country\_code

GROUP BY c2.country\_name

HAVING COUNT(b2.branch\_code) < COUNT(b.branch\_code)

);

-- Method 6: Using UNION (SET operators)

/\* derive min once \*/

WITH min\_val AS (

SELECT MIN(br\_cnt) AS mn

FROM (

SELECT COUNT(branch\_code) AS br\_cnt

FROM branch

RIGHT JOIN country c

ON branch.country\_code = c.country\_code

GROUP BY c.country\_name

)

t )

/\* union of identical result sets just for demonstration \*/

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) = (SELECT mn FROM min\_val)

UNION

SELECT cn.country\_name, COUNT(b.branch\_code)

FROM country cn

LEFT JOIN branch b

ON b.country\_code = cn.country\_code

GROUP BY cn.country\_name

HAVING COUNT(b.branch\_code) = (SELECT mn FROM min\_val);

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT country\_name, branch\_count

FROM (

SELECT cn.country\_name, COUNT(b.branch\_code) AS branch\_count

FROM country cn

LEFT JOIN branch b ON b.country\_code = cn.country\_code

GROUP BY cn.country\_name

) AS all\_counts

EXCEPT

SELECT cn.country\_name, COUNT(b.branch\_code)

FROM country cn

LEFT JOIN branch b

ON b.country\_code = cn.country\_code

GROUP BY cn.country\_name

HAVING COUNT(b.branch\_code) > (

SELECT MIN(cnt)

FROM (

SELECT COUNT(branch\_code) AS cnt

FROM country c

LEFT JOIN branch b

ON c.country\_code = b.country\_code

GROUP BY c.country\_name

) t

);

-- Method 8: Using INTERSECT (SET operators)

SELECT cn.country\_name, COUNT(b.branch\_code) AS branch\_count

FROM country cn

LEFT JOIN branch b ON b.country\_code = cn.country\_code

GROUP BY cn.country\_name

HAVING COUNT(b.branch\_code) = (

SELECT MIN(cnt)

FROM (

SELECT COUNT(branch\_code) AS cnt

FROM country c

LEFT JOIN branch b

ON c.country\_code = b.country\_code

GROUP BY c.country\_name

) t

)

INTERSECT

SELECT cn.country\_name, COUNT(b.branch\_code)

FROM country cn

LEFT JOIN branch b

ON b.country\_code = cn.country\_code

GROUP BY cn.country\_name;

\*/

-- 21. Which country has no branches?

-- Method 1: Using LEFT JOIN

SELECT country.country\_code

FROM country

LEFT JOIN branch ON country.country\_code = branch.country\_code

WHERE branch.country\_code IS NULL;

-- Method 2: Using NOT IN with subquery

SELECT COUNTRY\_CODE, COUNTRY\_NAME

FROM COUNTRY

WHERE COUNTRY\_CODE NOT IN (

SELECT DISTINCT COUNTRY\_CODE

FROM BRANCH

);

-- Method 3: Using NOT EXISTS (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

);

-- Method 4: Using UNION (SET operators)

/\* A: via LEFT JOIN … IS NULL \*/

SELECT c.country\_code, c.country\_name

FROM country c

LEFT JOIN branch b

ON b.country\_code = c.country\_code

WHERE b.country\_code IS NULL

UNION

/\* B: via GROUP BY HAVING COUNT = 0 \*/

SELECT c.country\_code, c.country\_name

FROM country c

LEFT JOIN branch b

ON b.country\_code = c.country\_code

GROUP BY c.country\_code, c.country\_name

HAVING COUNT(b.branch\_code) = 0;

/\* Does not work in MySQL

-- Method 5: Using EXCEPT (SET operators)

SELECT country\_code, country\_name

FROM country

EXCEPT

SELECT DISTINCT c.country\_code, c.country\_name

FROM country c

JOIN branch b

ON b.country\_code = c.country\_code;

-- Method 6: Using INTERSECT (SET operators)

SELECT country\_code, country\_name

FROM country

INTERSECT

SELECT c.country\_code, c.country\_name

FROM country c

LEFT JOIN branch b

ON b.country\_code = c.country\_code

GROUP BY c.country\_code, c.country\_name

HAVING COUNT(b.branch\_code) = 0;

\*/

-- 22. Which branch has more sales representatives?

-- Method 1: Using RANK function

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;

-- Method 2: Using DENSE\_RANK function

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;

-- Method 3: Using NOT IN

SELECT b.branch\_name, COUNT(sr.salesrep\_id) AS total\_salesreps

FROM branch b

LEFT JOIN salesrep sr

ON sr.branch\_code = b.branch\_code

GROUP BY b.branch\_name

HAVING COUNT(sr.salesrep\_id) NOT IN (

SELECT cnt

FROM (

SELECT COUNT(sr2.salesrep\_id) AS cnt

FROM branch b2

LEFT JOIN salesrep sr2

ON sr2.branch\_code = b2.branch\_code

GROUP BY b2.branch\_name

) x

WHERE cnt > COUNT(sr.salesrep\_id)

);

-- Method 4: Using HAVING

SELECT b.branch\_name, COUNT(sr.salesrep\_id) AS total\_salesreps

FROM branch b

LEFT JOIN salesrep sr

ON sr.branch\_code = b.branch\_code

GROUP BY b.branch\_name

HAVING COUNT(sr.salesrep\_id) = (

SELECT MAX(cnt)

FROM (

SELECT COUNT(sr2.salesrep\_id) AS cnt

FROM branch b2

LEFT JOIN salesrep sr2

ON sr2.branch\_code = b2.branch\_code

GROUP BY b2.branch\_name

) t

);

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

SELECT b.branch\_name, COUNT(sr.salesrep\_id) AS total\_salesreps

FROM branch b

LEFT JOIN salesrep sr

ON sr.branch\_code = b.branch\_code

GROUP BY b.branch\_name

HAVING NOT EXISTS (

SELECT 1

FROM branch b2

LEFT JOIN salesrep sr2

ON sr2.branch\_code = b2.branch\_code

GROUP BY b2.branch\_name

HAVING COUNT(sr2.salesrep\_id) > COUNT(sr.salesrep\_id)

);

-- Method 6: Using UNION (SET operators)

WITH max\_val AS (

SELECT MAX(cnt) AS mx

FROM (

SELECT COUNT(sr.salesrep\_id) AS cnt

FROM branch b

LEFT JOIN salesrep sr

ON sr.branch\_code = b.branch\_code

GROUP BY b.branch\_name

) t

)

SELECT b.branch\_name, COUNT(sr.salesrep\_id) AS total\_salesreps

FROM branch b

LEFT JOIN salesrep sr

ON sr.branch\_code = b.branch\_code

GROUP BY b.branch\_name

HAVING COUNT(sr.salesrep\_id) = (SELECT mx FROM max\_val)

UNION

SELECT b.branch\_name, COUNT(sr.salesrep\_id)

FROM branch b

LEFT JOIN salesrep sr

ON sr.branch\_code = b.branch\_code

GROUP BY b.branch\_name

HAVING COUNT(sr.salesrep\_id) = (SELECT mx FROM max\_val);

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT b.branch\_name, COUNT(sr.salesrep\_id) AS total\_salesreps

FROM branch b

LEFT JOIN salesrep sr

ON sr.branch\_code = b.branch\_code

GROUP BY b.branch\_name

EXCEPT

SELECT b2.branch\_name, COUNT(sr2.salesrep\_id)

FROM branch b2

LEFT JOIN salesrep sr2

ON sr2.branch\_code = b2.branch\_code

GROUP BY b2.branch\_name

HAVING COUNT(sr2.salesrep\_id) < (

SELECT MAX(cnt)

FROM (

SELECT COUNT(sr3.salesrep\_id) AS cnt

FROM branch b3

LEFT JOIN salesrep sr3

ON sr3.branch\_code = b3.branch\_code

GROUP BY b3.branch\_name

) AS max\_counts

);

-- Method 8: Using INTERSECT (SET operators)

SELECT b.branch\_name, COUNT(sr.salesrep\_id) AS total\_salesreps

FROM branch b

LEFT JOIN salesrep sr

ON sr.branch\_code = b.branch\_code

GROUP BY b.branch\_name

INTERSECT

SELECT b2.branch\_name, COUNT(sr2.salesrep\_id)

FROM branch b2

LEFT JOIN salesrep sr2

ON sr2.branch\_code = b2.branch\_code

GROUP BY b2.branch\_name

HAVING COUNT(sr2.salesrep\_id) = (

SELECT MAX(cnt)

FROM (

SELECT COUNT(sr3.salesrep\_id) AS cnt

FROM branch b3

LEFT JOIN salesrep sr3

ON sr3.branch\_code = b3.branch\_code

GROUP BY b3.branch\_name

) AS max\_counts

);

\*/

-- 23. Which branch has less sales representatives?

-- Method 1: Using RANK function

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;

-- Method 2: Using DENSE\_RANK function

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;

-- Method 3: Using NOT IN

SELECT b.branch\_name, COUNT(sr.salesrep\_id) AS total\_salesreps

FROM branch b

LEFT JOIN salesrep sr

ON sr.branch\_code = b.branch\_code

GROUP BY b.branch\_name

HAVING COUNT(sr.salesrep\_id) NOT IN (

SELECT cnt

FROM (

SELECT COUNT(sr2.salesrep\_id) AS cnt

FROM branch b2

LEFT JOIN salesrep sr2

ON sr2.branch\_code = b2.branch\_code

GROUP BY b2.branch\_name

) x

WHERE cnt < COUNT(sr.salesrep\_id)

);

-- Method 4: Using HAVING

SELECT b.branch\_name, COUNT(sr.salesrep\_id) AS total\_salesreps

FROM branch b LEFT JOIN salesrep sr

ON sr.branch\_code = b.branch\_code

GROUP BY b.branch\_name

HAVING COUNT(sr.salesrep\_id) = (

SELECT MIN(cnt)

FROM (

SELECT COUNT(sr2.salesrep\_id) AS cnt

FROM branch b2

LEFT JOIN salesrep sr2

ON sr2.branch\_code = b2.branch\_code

GROUP BY b2.branch\_name

) m

);

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

SELECT b.branch\_name, COUNT(sr.salesrep\_id) AS total\_salesreps

FROM branch b

LEFT JOIN salesrep sr

ON sr.branch\_code = b.branch\_code

GROUP BY b.branch\_name

HAVING NOT EXISTS (

SELECT 1

FROM branch b2

LEFT JOIN salesrep sr2

ON sr2.branch\_code = b2.branch\_code

GROUP BY b2.branch\_name

HAVING COUNT(sr2.salesrep\_id) < COUNT(sr.salesrep\_id)

);

-- Method 6: Using UNION (SET operators)

WITH min\_val AS (

SELECT MIN(cnt) AS mn

FROM (

SELECT COUNT(sr.salesrep\_id) AS cnt

FROM branch b

LEFT JOIN salesrep sr

ON sr.branch\_code = b.branch\_code

GROUP BY b.branch\_name

) t

)

SELECT b.branch\_name, COUNT(sr.salesrep\_id) AS total\_salesreps

FROM branch b

LEFT JOIN salesrep sr

ON sr.branch\_code = b.branch\_code

GROUP BY b.branch\_name

HAVING COUNT(sr.salesrep\_id) = (SELECT mn FROM min\_val)

UNION

SELECT b.branch\_name, COUNT(sr.salesrep\_id)

FROM branch b

LEFT JOIN salesrep sr

ON sr.branch\_code = b.branch\_code

GROUP BY b.branch\_name

HAVING COUNT(sr.salesrep\_id) = (SELECT mn FROM min\_val);

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT b.branch\_name, COUNT(sr.salesrep\_id) AS total\_salesreps

FROM branch b

LEFT JOIN salesrep sr

ON sr.branch\_code = b.branch\_code

GROUP BY b.branch\_name

EXCEPT

SELECT b2.branch\_name, COUNT(sr2.salesrep\_id)

FROM branch b2

LEFT JOIN salesrep sr2

ON sr2.branch\_code = b2.branch\_code

GROUP BY b2.branch\_name

HAVING COUNT(sr2.salesrep\_id) > (

SELECT MIN(cnt)

FROM (

SELECT COUNT(sr3.salesrep\_id) AS cnt

FROM branch b3

LEFT JOIN salesrep sr3

ON sr3.branch\_code = b3.branch\_code

GROUP BY b3.branch\_name

) AS min\_counts

);

-- Method 8: Using INTERSECT (SET operators)

SELECT b.branch\_name, COUNT(sr.salesrep\_id) AS total\_salesreps

FROM branch b

LEFT JOIN salesrep sr

ON sr.branch\_code = b.branch\_code

GROUP BY b.branch\_name

INTERSECT

SELECT b2.branch\_name, COUNT(sr2.salesrep\_id)

FROM branch b2

LEFT JOIN salesrep sr2

ON sr2.branch\_code = b2.branch\_code

GROUP BY b2.branch\_name

HAVING COUNT(sr2.salesrep\_id) = (

SELECT MIN(cnt)

FROM (

SELECT COUNT(sr3.salesrep\_id) AS cnt

FROM branch b3

LEFT JOIN salesrep sr3

ON sr3.branch\_code = b3.branch\_code

GROUP BY b3.branch\_name

) AS min\_counts

);

\*/

-- 24. Which branch has no sales representatives?

-- Method 1: Using LEFT JOIN

SELECT b.branch\_name

FROM branch b

LEFT JOIN salesrep sr

ON b.branch\_code = sr.branch\_code

WHERE sr.salesrep\_id IS NULL;

-- Method 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

);

-- Method 3: Using NOT EXISTS (Correlated Subquery)

SELECT b.branch\_name

FROM branch b

WHERE NOT EXISTS (

SELECT 1

FROM salesrep sr

WHERE sr.branch\_code = b.branch\_code

);

-- Method 4: Using HAVING

SELECT b.branch\_name

FROM branch b

LEFT JOIN salesrep sr

ON sr.branch\_code = b.branch\_code

GROUP BY b.branch\_name

HAVING COUNT(sr.salesrep\_id) = 0;

-- Method 5: Using UNION (SET operators)

SELECT b.branch\_name

FROM branch b

LEFT JOIN salesrep sr

ON sr.branch\_code = b.branch\_code

WHERE sr.salesrep\_id IS NULL

UNION

SELECT b.branch\_name

FROM branch b

LEFT JOIN salesrep sr

ON sr.branch\_code = b.branch\_code

GROUP BY b.branch\_name

HAVING COUNT(sr.salesrep\_id) = 0;

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT branch\_name

FROM branch

EXCEPT

SELECT DISTINCT b.branch\_name

FROM branch b

JOIN salesrep sr

ON sr.branch\_code = b.branch\_code;

-- Method 7: Using INTERSECT (SET operators)

SELECT b.branch\_name

FROM branch b

INTERSECT

SELECT b2.branch\_name

FROM branch b2

LEFT JOIN salesrep sr2

ON sr2.branch\_code = b2.branch\_code

GROUP BY b2.branch\_name

HAVING COUNT(sr2.salesrep\_id) = 0;

\*/

-- 25. Who is the active sales representatives?

-- Method 1: Using JOIN

SELECT DISTINCT sr.salesrep\_id, sr.salesrep\_name

FROM salesrep sr

JOIN order\_master om

ON sr.salesrep\_id = om.salesrep\_id;

-- Method 2: Using EXISTS (Correlated Subquery)

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

);

-- Method 3: Using IN

SELECT salesrep\_id, salesrep\_name

FROM salesrep

WHERE salesrep\_id IN (

SELECT DISTINCT salesrep\_id

FROM order\_master

);

-- Method 4: Using NOT IN

SELECT salesrep\_id, salesrep\_name

FROM salesrep

WHERE salesrep\_id NOT IN (

SELECT salesrep\_id

FROM salesrep

WHERE salesrep\_id NOT IN (

SELECT DISTINCT salesrep\_id

FROM order\_master

)

);

-- Method 5: Using Subquery

SELECT salesrep\_id, salesrep\_name

FROM salesrep

WHERE salesrep\_id IN (

SELECT salesrep\_id

FROM order\_master

GROUP BY salesrep\_id

);

-- Method 6: Using UNION (SET operators)

SELECT sr.salesrep\_id, sr.salesrep\_name

FROM salesrep sr

JOIN order\_master om

ON sr.salesrep\_id = om.salesrep\_id

UNION

SELECT salesrep\_id, salesrep\_name

FROM salesrep

WHERE salesrep\_id IN (

SELECT salesrep\_id

FROM order\_master

);

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT salesrep\_id, salesrep\_name

FROM salesrep

EXCEPT

SELECT sr.salesrep\_id, sr.salesrep\_name

FROM salesrep sr

LEFT JOIN order\_master om

ON sr.salesrep\_id = om.salesrep\_id

WHERE om.salesrep\_id IS NULL;

-- Method 8: Using INTERSECT (SET operators)

SELECT salesrep\_id, salesrep\_name

FROM salesrep

INTERSECT

SELECT sr.salesrep\_id, sr.salesrep\_name

FROM salesrep sr

JOIN order\_master om

ON sr.salesrep\_id = om.salesrep\_id;

\*/

-- 26. Display all the manager's names?

-- Method 1: Using JOIN

SELECT DISTINCT s.salesrep\_id, S.SALESREP\_NAME AS manager\_name

FROM salesrep s

JOIN SALESREP M

ON S.salesrep\_id = M.MGR;

-- Method 2: Using Subquery

SELECT salesrep\_id, salesrep\_name AS manager\_name

FROM salesrep

WHERE salesrep\_id IN (

SELECT DISTINCT mgr

FROM salesrep

WHERE mgr IS NOT NULL

);

-- Method 3: Using EXISTS (Correlated Subquery)

SELECT salesrep\_id, salesrep\_name AS manager\_name

FROM salesrep s

WHERE EXISTS (

SELECT 1

FROM salesrep m

WHERE m.mgr = s.salesrep\_id

);

-- Method 4: Using NOT IN

SELECT salesrep\_id, salesrep\_name AS manager\_name

FROM salesrep

WHERE salesrep\_id NOT IN (

SELECT salesrep\_id

FROM salesrep

WHERE mgr IS NOT NULL

);

-- Method 5: Using UNION (SET operators)

SELECT salesrep\_id, salesrep\_name AS manager\_name

FROM salesrep

WHERE salesrep\_id IN (

SELECT mgr

FROM salesrep

WHERE mgr IS NOT NULL

)

UNION

SELECT DISTINCT s.salesrep\_id, s.salesrep\_name

FROM salesrep s

JOIN salesrep m

ON s.salesrep\_id = m.mgr;

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT salesrep\_id, salesrep\_name

FROM salesrep

EXCEPT

SELECT salesrep\_id, salesrep\_name

FROM salesrep

WHERE salesrep\_id NOT IN (

SELECT mgr

FROM salesrep

WHERE mgr IS NOT NULL

);

-- Method 7: Using INTERSECT (SET operators)

SELECT salesrep\_id, salesrep\_name

FROM salesrep

INTERSECT

SELECT salesrep\_id, salesrep\_name

FROM salesrep

WHERE salesrep\_id IN (

SELECT mgr

FROM salesrep

WHERE mgr IS NOT NULL

);

\*/

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

-- Method 1: Using EXISTS (Correlated Subquery)

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

);

-- Method 2: Using JOIN

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;

-- Method 3: Using NOT IN

SELECT e.salesrep\_id, e.salesrep\_name, e.salesrep\_doj FROM salesrep e

WHERE e.mgr IS NOT NULL AND e.salesrep\_id NOT IN (

SELECT e2.salesrep\_id

FROM salesrep e2

JOIN salesrep m2

ON m2.salesrep\_id = e2.mgr

WHERE e2.salesrep\_doj >= m2.salesrep\_doj

);

-- Method 4: Using Subquery

SELECT e.salesrep\_id, e.salesrep\_name, e.salesrep\_doj

FROM salesrep e

WHERE e.mgr IS NOT NULL AND e.salesrep\_doj < (

SELECT m.salesrep\_doj

FROM salesrep m

WHERE m.salesrep\_id = e.mgr

);

-- Method 5: Using UNION (SET operators)

SELECT e.salesrep\_id, e.salesrep\_name, e.salesrep\_doj

FROM salesrep e

JOIN salesrep m

ON m.salesrep\_id = e.mgr

WHERE e.salesrep\_doj < m.salesrep\_doj

UNION

SELECT e.salesrep\_id, e.salesrep\_name, e.salesrep\_doj

FROM salesrep e

WHERE e.mgr IS NOT NULL AND e.salesrep\_doj < (

SELECT m.salesrep\_doj

FROM salesrep m

WHERE m.salesrep\_id = e.mgr

);

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT e.salesrep\_id, e.salesrep\_name, e.salesrep\_doj

FROM salesrep e

JOIN salesrep m

ON m.salesrep\_id = e.mgr

EXCEPT

SELECT e.salesrep\_id, e.salesrep\_name, e.salesrep\_doj

FROM salesrep e

JOIN salesrep m

ON m.salesrep\_id = e.mgr

WHERE e.salesrep\_doj >= m.salesrep\_doj;

-- Method 7: Using INTERSECT (SET operators)

SELECT e.salesrep\_id, e.salesrep\_name, e.salesrep\_doj

FROM salesrep e

JOIN salesrep m

ON m.salesrep\_id = e.mgr

INTERSECT

SELECT e.salesrep\_id, e.salesrep\_name, e.salesrep\_doj

FROM salesrep e

JOIN salesrep m

ON m.salesrep\_id = e.mgr

WHERE e.salesrep\_doj < m.salesrep\_doj;

\*/

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

-- Method 1: Using JOIN

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;

-- Method 2: Using EXISTS (Correlated Subquery)

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

);

-- Method 3: Using NOT IN

SELECT e.salesrep\_id, e.salesrep\_name, e.salary

FROM salesrep e

WHERE e.mgr IS NOT NULL AND e.salesrep\_id NOT IN (

SELECT e2.salesrep\_id

FROM salesrep e2

JOIN salesrep m2

ON m2.salesrep\_id = e2.mgr

WHERE e2.salary <= m2.salary

);

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

SELECT e.salesrep\_id, e.salesrep\_name, e.salary

FROM salesrep e

WHERE e.mgr IS NOT NULL AND NOT EXISTS (

SELECT 1

FROM salesrep m

WHERE m.salesrep\_id = e.mgr AND m.salary >= e.salary

);

-- Method 5: Using UNION (SET operators)

SELECT e.salesrep\_id, e.salesrep\_name, e.salary

FROM salesrep e

JOIN salesrep m

ON m.salesrep\_id = e.mgr

WHERE e.salary > m.salary

UNION

SELECT salesrep\_id, salesrep\_name, salary

FROM salesrep

WHERE salesrep\_id IN (

SELECT DISTINCT om.salesrep\_id

FROM order\_master om

);

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT e.salesrep\_id, e.salesrep\_name, e.salary

FROM salesrep e

JOIN salesrep m

ON m.salesrep\_id = e.mgr

EXCEPT

SELECT e.salesrep\_id, e.salesrep\_name, e.salary

FROM salesrep e

JOIN salesrep m

ON m.salesrep\_id = e.mgr

WHERE e.salary <= m.salary;

-- Method 7: Using INTERSECT (SET operators)

SELECT e.salesrep\_id, e.salesrep\_name, e.salary

FROM salesrep e

JOIN salesrep m

ON m.salesrep\_id = e.mgr

INTERSECT

SELECT e.salesrep\_id, e.salesrep\_name, e.salary

FROM salesrep e

JOIN salesrep m

ON m.salesrep\_id = e.mgr

WHERE e.salary > m.salary;

\*/

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

-- Method 1: Using Subquery

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

);

-- Method 2: Using 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);

-- Method 3: Using JOIN

SELECT m.salesrep\_id, m.salesrep\_name, m.salary, t.avg\_team\_salary

FROM salesrep m

JOIN (

SELECT e.mgr AS manager\_id, AVG(e.salary) AS avg\_team\_salary

FROM salesrep e

WHERE e.mgr IS NOT NULL

GROUP BY e.mgr

) t

ON t.manager\_id = m.salesrep\_id

WHERE m.salary > t.avg\_team\_salary;

-- Method 4: Using NOT IN

SELECT m.salesrep\_id, m.salesrep\_name, m.salary

FROM salesrep m

WHERE m.salesrep\_id IN (

SELECT DISTINCT mgr

FROM salesrep

WHERE mgr IS NOT NULL

) AND m.salesrep\_id NOT IN (

SELECT e.mgr

FROM salesrep e

GROUP BY e.mgr

HAVING AVG(e.salary) >= (

SELECT s.salary

FROM salesrep s

WHERE s.salesrep\_id = e.mgr

)

);

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

SELECT m.salesrep\_id, m.salesrep\_name, m.salary

FROM salesrep m

WHERE m.salesrep\_id IN (

SELECT DISTINCT mgr

FROM salesrep

WHERE mgr IS NOT NULL

) AND NOT EXISTS (

SELECT 1

FROM salesrep e

GROUP BY e.mgr

HAVING e.mgr = m.salesrep\_id AND AVG(e.salary) >= m.salary

);

-- Method 6: Using UNION (SET operators)

SELECT m.salesrep\_id AS mgr\_id, AVG(e.salary) AS avg\_team\_salary

FROM salesrep m

JOIN salesrep e ON m.salesrep\_id = e.mgr

GROUP BY m.salesrep\_id

UNION

SELECT m.salesrep\_id AS mgr\_id, m.salary AS avg\_team\_salary

FROM salesrep m

WHERE m.salesrep\_id IN (

SELECT e.mgr FROM salesrep e WHERE e.mgr IS NOT NULL

)

AND m.salary > (

SELECT AVG(e.salary)

FROM salesrep e

WHERE e.mgr = m.salesrep\_id

);

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT m.salesrep\_id, m.salesrep\_name, m.salary

FROM salesrep m

WHERE m.salesrep\_id IN (

SELECT DISTINCT mgr

FROM salesrep

WHERE mgr IS NOT NULL

)

EXCEPT

SELECT m.salesrep\_id, m.salesrep\_name, m.salary

FROM salesrep m

JOIN (

SELECT e.mgr AS manager\_id, AVG(e.salary) AS avg\_sal

FROM salesrep e

WHERE e.mgr IS NOT NULL

GROUP BY e.mgr

) a

ON a.manager\_id = m.salesrep\_id

WHERE m.salary <= a.avg\_sal;

-- Method 8: Using INTERSECT (SET operators)

SELECT m.salesrep\_id, m.salesrep\_name, m.salary

FROM salesrep m

WHERE m.salesrep\_id IN (

SELECT DISTINCT mgr

FROM salesrep

WHERE mgr IS NOT NULL

)

INTERSECT

SELECT m.salesrep\_id, m.salesrep\_name, m.salary

FROM salesrep m

JOIN (

SELECT e.mgr AS manager\_id, AVG(e.salary) AS avg\_sal

FROM salesrep e

WHERE e.mgr IS NOT NULL

GROUP BY e.mgr

) a

ON a.manager\_id = m.salesrep\_id

WHERE m.salary > a.avg\_sal;

\*/

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

-- Method 1: Using DENSE\_RANK function

SELECT salesrep\_id, salesrep\_name, salesrep\_doj

FROM (

SELECT \*,

DENSE\_RANK() OVER(ORDER BY salesrep\_doj ASC) AS rnk

FROM salesrep

) AS ranked\_reps

WHERE rnk = 1;

-- Method 2: Using RANK function

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;

-- Method 3: Using Subquery

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

FROM SALESREP

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

-- Method 4: Using JOIN

SELECT s.salesrep\_id, s.salesrep\_name, s.salesrep\_doj

FROM salesrep s

JOIN (

SELECT MIN(salesrep\_doj) AS min\_doj

FROM salesrep

) t

ON s.salesrep\_doj = t.min\_doj;

-- Method 5: Using NOT IN

SELECT s.salesrep\_id, s.salesrep\_name, s.salesrep\_doj

FROM salesrep s

WHERE s.salesrep\_doj NOT IN (

SELECT s2.salesrep\_doj

FROM salesrep s2

WHERE s2.salesrep\_doj > (

SELECT MIN(salesrep\_doj)

FROM salesrep

)

);

-- Method 6: Using NOT EXISTS (Correlated Subquery)

SELECT s1.salesrep\_id, s1.salesrep\_name, s1.salesrep\_doj

FROM salesrep s1

WHERE NOT EXISTS (

SELECT 1

FROM salesrep s2

WHERE s2.salesrep\_doj < s1.salesrep\_doj

);

-- Method 7: Using UNION (SET operators)

SELECT salesrep\_doj

FROM salesrep

WHERE salesrep\_doj = (

SELECT MIN(salesrep\_doj)

FROM salesrep

)

UNION

SELECT salesrep\_doj

FROM salesrep

WHERE salesrep\_doj = (

SELECT MIN(salesrep\_doj)

FROM salesrep

);

/\* Does not work in MySQL

-- Method 8: Using EXCEPT (SET operators)

SELECT salesrep\_id, salesrep\_name, salesrep\_doj

FROM salesrep

EXCEPT

SELECT salesrep\_id, salesrep\_name, salesrep\_doj

FROM salesrep

WHERE salesrep\_doj > (

SELECT MIN(salesrep\_doj)

FROM salesrep

);

-- Method 9: Using INTERSECT (SET operators)

SELECT salesrep\_id, salesrep\_name, salesrep\_doj

FROM salesrep

INTERSECT

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?

-- Method 1: Using RANK function

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;

-- Method 2: Using DENSE\_RANK function

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;

-- Method 3: Using NOT EXISTS (Correlated Subquery)

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

)

);

-- Method 4: Using JOIN

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.salesrep\_id IS NULL

AND s.salesrep\_doj = (

SELECT MIN(salesrep\_doj)

FROM salesrep

WHERE salesrep\_id NOT IN (

SELECT DISTINCT salesrep\_id

FROM order\_master

)

);

-- Method 5: Using NOT IN

SELECT salesrep\_id, salesrep\_name, salesrep\_doj

FROM salesrep

WHERE salesrep\_id NOT IN (

SELECT DISTINCT salesrep\_id

FROM order\_master

)

AND salesrep\_doj = (

SELECT MIN(salesrep\_doj)

FROM salesrep

WHERE salesrep\_id NOT IN (

SELECT DISTINCT salesrep\_id

FROM order\_master

)

);

-- Method 6: Using UNION (SET operators)

SELECT salesrep\_id, salesrep\_name, salesrep\_doj

FROM salesrep

WHERE salesrep\_id IN (

SELECT salesrep\_id FROM salesrep

WHERE salesrep\_id NOT IN (

SELECT DISTINCT salesrep\_id

FROM order\_master

) AND salesrep\_doj = (

SELECT MIN(salesrep\_doj)

FROM salesrep

WHERE salesrep\_id NOT IN (

SELECT DISTINCT salesrep\_id

FROM order\_master

)

)

)

UNION

SELECT salesrep\_id, salesrep\_name, salesrep\_doj

FROM salesrep

WHERE 1=0;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

-- All sales reps without orders

SELECT s.salesrep\_id, s.salesrep\_name, s.salesrep\_doj

FROM salesrep s

WHERE NOT EXISTS (

SELECT 1

FROM order\_master om

WHERE om.salesrep\_id = s.salesrep\_id

)

EXCEPT

-- Remove reps who haven't been on bench the longest

SELECT s1.salesrep\_id, s1.salesrep\_name, s1.salesrep\_doj

FROM salesrep s1

WHERE NOT EXISTS (

SELECT 1

FROM order\_master om

WHERE om.salesrep\_id = s1.salesrep\_id

)

AND s1.salesrep\_doj > (

SELECT MIN(s2.salesrep\_doj)

FROM salesrep s2

WHERE NOT EXISTS (

SELECT 1

FROM order\_master om

WHERE om.salesrep\_id = s2.salesrep\_id

)

);

-- Method 8: Using INTERSECT (SET operators)

-- All sales reps without orders

SELECT s.salesrep\_id, s.salesrep\_name, s.salesrep\_doj

FROM salesrep s

WHERE NOT EXISTS (

SELECT 1

FROM order\_master om

WHERE om.salesrep\_id = s.salesrep\_id

)

INTERSECT

-- Only those with the earliest join date

SELECT s.salesrep\_id, s.salesrep\_name, s.salesrep\_doj

FROM salesrep s

WHERE s.salesrep\_doj = (

SELECT MIN(s2.salesrep\_doj)

FROM salesrep s2

WHERE NOT EXISTS (

SELECT 1

FROM order\_master om

WHERE om.salesrep\_id = s2.salesrep\_id

)

);

\*/

-- 32. Display the details of in-active sales representatives.

-- Method 1: Using LEFT JOIN

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;

-- Method 2: Using NOT EXISTS (Correlated Subquery)

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

);

-- Method 3: Using NOT IN

SELECT salesrep\_id, salesrep\_name, salesrep\_doj

FROM salesrep

WHERE salesrep\_id NOT IN (

SELECT DISTINCT salesrep\_id

FROM order\_master

);

-- Method 4: Using Subquery

SELECT s.salesrep\_id, s.salesrep\_name, s.salesrep\_doj

FROM salesrep s

WHERE s.salesrep\_id IN (

SELECT s2.salesrep\_id

FROM salesrep s2

WHERE s2.salesrep\_id NOT IN (

SELECT salesrep\_id

FROM order\_master

)

);

-- Method 6: Using UNION (SET operators)

SELECT salesrep\_id, salesrep\_name, salesrep\_doj

FROM salesrep

WHERE salesrep\_id NOT IN (

SELECT DISTINCT salesrep\_id

FROM order\_master

)

UNION

SELECT salesrep\_id, salesrep\_name, salesrep\_doj

FROM salesrep

WHERE 1 = 0;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

-- All sales reps

SELECT salesrep\_id, salesrep\_name, salesrep\_doj

FROM salesrep

EXCEPT

-- Remove active sales reps (those with orders)

SELECT s.salesrep\_id, s.salesrep\_name, s.salesrep\_doj

FROM salesrep s

JOIN order\_master om

ON s.salesrep\_id = om.salesrep\_id;

-- Method 8: Using INTERSECT (SET operators)

-- All sales reps

SELECT salesrep\_id, salesrep\_name, salesrep\_doj

FROM salesrep

INTERSECT

-- Only those without orders

SELECT s.salesrep\_id, s.salesrep\_name, s.salesrep\_doj

FROM salesrep s

WHERE NOT EXISTS (

SELECT 1

FROM order\_master om

WHERE om.salesrep\_id = s.salesrep\_id

);

\*/

-- 33. Generate the report: Sales\_rep\_id | Sales\_rep\_name | Salary | Total\_order\_value | Commission(5% of bill)

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS GetSalesRepCommissionReport;

DELIMITER $$

CREATE PROCEDURE GetSalesRepCommissionReport()

BEGIN

SELECT sr.salesrep\_id, sr.salesrep\_name, sr.salary, IFNULL(SUM(p.price \* od.qty\_ordered), 0) AS total\_order\_value, ROUND(IFNULL(SUM(p.price \* od.qty\_ordered), 0) \* 0.05, 2) AS commission

FROM salesrep sr

LEFT JOIN order\_master om

ON sr.salesrep\_id = om.salesrep\_id

LEFT JOIN order\_detail od

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

LEFT JOIN product p

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

GROUP BY sr.salesrep\_id, sr.salesrep\_name, sr.salary;

END $$

DELIMITER ;

CALL GetSalesRepCommissionReport();

-- Method 2: Using LEFT JOIN

SELECT s.salesrep\_id, s.salesrep\_name, s.salary

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

HAVING COALESCE(SUM(p.price \* o.qty\_ordered), 0) = 0;

-- Method 3: 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 commission

FROM salesrep s;

-- Method 4: Using NOT IN

SELECT s.salesrep\_id, s.salesrep\_name, s.salary, COALESCE(SUM(p.price \* od.qty\_ordered), 0) AS total\_order\_value, COALESCE(SUM(p.price \* od.qty\_ordered), 0) \* 0.05 AS commission

FROM salesrep s

LEFT JOIN order\_master om

ON s.salesrep\_id = om.salesrep\_id

LEFT JOIN order\_detail od

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

LEFT JOIN product p

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

WHERE s.salesrep\_id NOT IN (

SELECT DISTINCT salesrep\_id

FROM order\_master

)

GROUP BY s.salesrep\_id, s.salesrep\_name, s.salary;

-- Method 5: Using EXISTS (Correlated Subquery)

SELECT s.salesrep\_id, s.salesrep\_name, s.salary, COALESCE(SUM(p.price \* od.qty\_ordered), 0) AS total\_order\_value, COALESCE(SUM(p.price \* od.qty\_ordered), 0) \* 0.05 AS commission

FROM salesrep s

LEFT JOIN order\_master om

ON s.salesrep\_id = om.salesrep\_id

LEFT JOIN order\_detail od

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

LEFT JOIN product p

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

WHERE NOT EXISTS (

SELECT 1

FROM order\_master om2

WHERE om2.salesrep\_id = s.salesrep\_id

)

GROUP BY s.salesrep\_id, s.salesrep\_name, s.salary;

-- Method 6: Using UNION (SET operators)

SELECT s.salesrep\_id, s.salesrep\_name, s.salary, COALESCE(SUM(p.price \* od.qty\_ordered), 0) AS total\_order\_value, COALESCE(SUM(p.price \* od.qty\_ordered), 0) \* 0.05 AS commission

FROM salesrep s

LEFT JOIN order\_master om

ON s.salesrep\_id = om.salesrep\_id

LEFT JOIN order\_detail od

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

LEFT JOIN product p

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

GROUP BY s.salesrep\_id, s.salesrep\_name, s.salary

HAVING total\_order\_value = 0

UNION

SELECT salesrep\_id, salesrep\_name, salary, 0, 0

FROM salesrep

WHERE salesrep\_id NOT IN (

SELECT DISTINCT salesrep\_id

FROM order\_master

);

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

-- All sales reps with their performance metrics

SELECT s.salesrep\_id, s.salesrep\_name, s.salary, COALESCE(SUM(p.price \* od.qty\_ordered), 0) AS total\_order\_value, COALESCE(SUM(p.price \* od.qty\_ordered) \* 0.05, 0) AS commission

FROM salesrep s

LEFT JOIN order\_master om

ON s.salesrep\_id = om.salesrep\_id

LEFT JOIN order\_detail od

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

LEFT JOIN product p

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

GROUP BY s.salesrep\_id, s.salesrep\_name, s.salary

EXCEPT

-- Remove sales reps with no orders (if needed)

SELECT s.salesrep\_id, s.salesrep\_name, s.salary, 0, 0

FROM salesrep s

WHERE NOT EXISTS (

SELECT 1

FROM order\_master om

WHERE om.salesrep\_id = s.salesrep\_id

);

-- Method 8: Using INTERSECT (SET operators)

-- All sales reps with their performance metrics

SELECT s.salesrep\_id, s.salesrep\_name, s.salary, COALESCE(SUM(p.price \* od.qty\_ordered), 0) AS total\_order\_value, COALESCE(SUM(p.price \* od.qty\_ordered) \* 0.05, 0) AS commission

FROM salesrep s

LEFT JOIN order\_master om

ON s.salesrep\_id = om.salesrep\_id

LEFT JOIN order\_detail od

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

LEFT JOIN product p

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

GROUP BY s.salesrep\_id, s.salesrep\_name, s.salary

INTERSECT

-- Only include sales reps with orders (if needed)

SELECT s.salesrep\_id, s.salesrep\_name, s.salary, COALESCE(SUM(p.price \* od.qty\_ordered), 0), COALESCE(SUM(p.price \* od.qty\_ordered) \* 0.05, 0)

FROM salesrep s

JOIN order\_master om

ON s.salesrep\_id = om.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\_id, s.salesrep\_name, s.salary;

\*/

-- 34. Generate the report like: Sales\_rep\_name | Order\_code | Bill\_value

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS GetSalesRepOrderBillValue;

DELIMITER $$

CREATE PROCEDURE GetSalesRepOrderBillValue()

BEGIN

SELECT sr.salesrep\_name, om.order\_code, SUM(p.price \* od.qty\_ordered) AS bill\_value

FROM order\_master om

JOIN salesrep sr

ON om.salesrep\_id = sr.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 sr.salesrep\_name, om.order\_code;

END $$

DELIMITER ;

CALL GetSalesRepOrderBillValue();

-- Method 2: Using JOIN

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;

-- Method 3: Using WITH CTE

WITH order\_values 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 order\_values ov

JOIN salesrep s

ON ov.salesrep\_id = s.salesrep\_id;

-- Method 4: Using NOT IN

SELECT s.salesrep\_name, om.order\_code, SUM(p.price \* od.qty\_ordered) AS bill\_value

FROM salesrep s

JOIN order\_master om

ON s.salesrep\_id = om.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\_code NOT IN (

SELECT order\_code

FROM order\_detail

WHERE qty\_ordered IS NULL

)

GROUP BY s.salesrep\_name, om.order\_code;

-- Method 5: Using EXISTS (Correlated Subquery)

SELECT s.salesrep\_name, om.order\_code, SUM(p.price \* od.qty\_ordered) AS bill\_value

FROM salesrep s

JOIN order\_master om

ON s.salesrep\_id = om.salesrep\_id

JOIN order\_detail od

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

JOIN product p

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

WHERE EXISTS (

SELECT 1

FROM order\_detail od2

WHERE od2.order\_code = om.order\_code

)

GROUP BY s.salesrep\_name, om.order\_code;

-- Method 6: Using UNION (SET operators)

SELECT s.salesrep\_name, om.order\_code, SUM(p.price \* od.qty\_ordered) AS bill\_value

FROM salesrep s

JOIN order\_master om

ON s.salesrep\_id = om.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

UNION

SELECT s.salesrep\_name, NULL AS order\_code, 0 AS bill\_value

FROM salesrep s

WHERE s.salesrep\_id NOT IN (

SELECT DISTINCT salesrep\_id

FROM order\_master

);

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

-- All possible sales rep - order combinations with bill values

SELECT e.emp\_name AS Sales\_rep\_name, om.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

JOIN order\_master om

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

JOIN employee e

ON om.sales\_rep\_id = e.emp\_id

GROUP BY e.emp\_name, om.order\_code

EXCEPT

-- Remove orders with zero bill value (though unlikely in this case)

SELECT e.emp\_name, om.order\_code, 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

JOIN employee e

ON om.sales\_rep\_id = e.emp\_id

GROUP BY e.emp\_name, om.order\_code

HAVING SUM(p.price \* od.qty\_ordered) <= 0;

-- Method 8: Using INTERSECT (SET operators)

-- All sales reps with their orders and bill values

SELECT e.emp\_name AS Sales\_rep\_name, om.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

JOIN order\_master om

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

JOIN employee e

ON om.sales\_rep\_id = e.emp\_id

GROUP BY e.emp\_name, om.order\_code

INTERSECT

-- Only include valid sales (positive bill values)

SELECT e.emp\_name, om.order\_code, 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

JOIN employee e

ON om.sales\_rep\_id = e.emp\_id

GROUP BY e.emp\_name, om.order\_code

HAVING SUM(p.price \* od.qty\_ordered) > 0;

\*/

-- 35. Generate the report: Sales\_rep\_name | Order\_code | No\_of\_products(in order)

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS GetSalesRepOrderProducts;

DELIMITER $$

CREATE PROCEDURE GetSalesRepOrderProducts()

BEGIN

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;

END $$

DELIMITER ;

CALL GetSalesRepOrderProducts();

-- Method 2: Using JOIN

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;

-- Method 3: Using subquery

SELECT s.salesrep\_name, om.order\_code, (

SELECT SUM(od.qty\_ordered)

FROM order\_detail od

WHERE od.order\_code = om.order\_code

) AS no\_of\_products

FROM order\_master om

JOIN salesrep s

ON om.salesrep\_id = s.salesrep\_id;

-- Method 4: Using NOT IN

SELECT s.salesrep\_name, om.order\_code, SUM(od.qty\_ordered) AS no\_of\_products

FROM salesrep s

JOIN order\_master om ON om.salesrep\_id = s.salesrep\_id

JOIN order\_detail od

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

WHERE om.order\_code NOT IN (

SELECT order\_code

FROM order\_detail

GROUP BY order\_code

HAVING SUM(qty\_ordered) = 0

)

GROUP BY s.salesrep\_name, om.order\_code;

-- Method 5: Using EXISTS (Correlated Subquery)

SELECT s.salesrep\_name, om.order\_code, SUM(od.qty\_ordered) AS no\_of\_products

FROM salesrep s

JOIN order\_master om

ON om.salesrep\_id = s.salesrep\_id

JOIN order\_detail od

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

WHERE EXISTS (

SELECT 1

FROM order\_detail od2

WHERE od2.order\_code = om.order\_code

)

GROUP BY s.salesrep\_name, om.order\_code;

-- Method 6: Using UNION (SET operators)

SELECT s.salesrep\_name, om.order\_code, SUM(od.qty\_ordered) AS no\_of\_products

FROM salesrep s

JOIN order\_master om

ON om.salesrep\_id = s.salesrep\_id

JOIN order\_detail od

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

GROUP BY s.salesrep\_name, om.order\_code

UNION

SELECT s.salesrep\_name, NULL AS order\_code, 0 AS no\_of\_products

FROM salesrep s

WHERE NOT EXISTS (

SELECT 1

FROM order\_master om

WHERE om.salesrep\_id = s.salesrep\_id

);

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT s.salesrep\_name, om.order\_code, SUM(od.qty\_ordered) AS no\_of\_products

FROM salesrep s

JOIN order\_master om

ON om.salesrep\_id = s.salesrep\_id

JOIN order\_detail od

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

GROUP BY s.salesrep\_name, om.order\_code

EXCEPT

-- Remove orders whose product count = 0

SELECT s2.salesrep\_name, om2.order\_code, 0

FROM salesrep s2

JOIN order\_master om2

ON om2.salesrep\_id = s2.salesrep\_id

LEFT JOIN order\_detail od2

ON od2.order\_code = om2.order\_code

GROUP BY s2.salesrep\_name, om2.order\_code

HAVING COALESCE(SUM(od2.qty\_ordered),0) = 0;

-- Method 8: Using INTERSECT (SET operators)

SELECT s.salesrep\_name, om.order\_code, SUM(od.qty\_ordered) AS no\_of\_products

FROM salesrep s

JOIN order\_master om

ON om.salesrep\_id = s.salesrep\_id

JOIN order\_detail od

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

GROUP BY s.salesrep\_name, om.order\_code

INTERSECT

-- Keep only orders whose product count > 0

SELECT s.salesrep\_name,

om.order\_code,

SUM(od.qty\_ordered)

FROM salesrep s

JOIN order\_master om

ON om.salesrep\_id = s.salesrep\_id

JOIN order\_detail od

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

GROUP BY s.salesrep\_name, om.order\_code

HAVING SUM(od.qty\_ordered) > 0;

\*/

-- 36. Generate the report like: Sales\_rep\_name | Order\_code | Bill\_value | No\_of\_products(in order)

-- A. For orders placed in 2010:

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS GetOrdersIn2010;

DELIMITER $$

CREATE PROCEDURE GetOrdersIn2010()

BEGIN

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;

END $$

DELIMITER ;

CALL GetOrdersIn2010();

-- Method 2: Using JOIN

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;

-- Method 3: 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;

-- Method 4: Using NOT IN

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

AND om.order\_code NOT IN (

SELECT order\_code

FROM order\_master

WHERE YEAR(order\_date) <> 2010

)

GROUP BY s.salesrep\_name, om.order\_code;

-- Method 5: Using EXISTS / NOT EXISTS (Correlated 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 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 EXISTS (

SELECT 1

FROM order\_master om2

WHERE om.order\_code = om2.order\_code AND YEAR(om2.order\_date) = 2010

)

GROUP BY s.salesrep\_name, om.order\_code;

-- Method 6: Using UNION (SET operators)

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

UNION

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;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

-- All sales data

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

EXCEPT

-- Remove orders not from 2010

SELECT s.salesrep\_name, om.order\_code, SUM(p.price \* od.qty\_ordered), SUM(od.qty\_ordered)

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;

-- Method 8: Using INTERSECT (SET operators)

-- All sales data

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

INTERSECT

-- Only orders from 2010

SELECT s.salesrep\_name, om.order\_code, SUM(p.price \* od.qty\_ordered), SUM(od.qty\_ordered)

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;

\*/

-- B. For sales reps with ≤ 10 orders:

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS GetOrdersFromLowVolumeReps;

DELIMITER $$

CREATE PROCEDURE GetOrdersFromLowVolumeReps()

BEGIN

WITH rep\_order\_count AS (

SELECT salesrep\_id

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 rep\_order\_count roc

ON om.salesrep\_id = roc.salesrep\_id

GROUP BY s.salesrep\_name, om.order\_code;

END $$

DELIMITER ;

CALL GetOrdersFromLowVolumeReps();

-- Method 2: Using JOIN

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;

-- Method 3: Using With CTE

WITH rep\_order\_count 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 rep\_order\_count roc

ON om.salesrep\_id = roc.salesrep\_id

GROUP BY s.salesrep\_name, om.order\_code;

-- Method 4: Using NOT IN

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.salesrep\_id NOT IN (

SELECT salesrep\_id

FROM order\_master

GROUP BY salesrep\_id

HAVING COUNT(order\_code) > 10

)

GROUP BY s.salesrep\_name, om.order\_code;

-- Method 5: Using EXISTS (Correlated 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 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 EXISTS (

SELECT 1

FROM (

SELECT salesrep\_id, COUNT(order\_code) AS order\_count

FROM order\_master

GROUP BY salesrep\_id

HAVING order\_count <= 10

) AS low\_orders

WHERE low\_orders.salesrep\_id = om.salesrep\_id

)

GROUP BY s.salesrep\_name, om.order\_code;

-- Method 6 Using UNION (SET operators)

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.salesrep\_id IN (

SELECT salesrep\_id

FROM order\_master

GROUP BY salesrep\_id

HAVING COUNT(order\_code) <= 10

)

GROUP BY s.salesrep\_name, om.order\_code

UNION

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 (

SELECT salesrep\_id

FROM order\_master

GROUP BY salesrep\_id

HAVING COUNT(order\_code) <= 10

) low\_volume\_reps

ON om.salesrep\_id = low\_volume\_reps.salesrep\_id

GROUP BY s.salesrep\_name, om.order\_code;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

-- All sales data

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

EXCEPT

-- Remove sales reps with >10 orders

SELECT s.salesrep\_name, om.order\_code, SUM(p.price \* od.qty\_ordered), SUM(od.qty\_ordered)

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.salesrep\_id IN (

SELECT salesrep\_id

FROM order\_master

GROUP BY salesrep\_id

HAVING COUNT(order\_code) > 10

)

GROUP BY s.salesrep\_name, om.order\_code;

-- Method 8: Using INTERSECT (SET operators)

-- All sales data

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

INTERSECT

-- Only sales reps with ≤10 orders

SELECT s.salesrep\_name, om.order\_code, SUM(p.price \* od.qty\_ordered), SUM(od.qty\_ordered)

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.salesrep\_id IN (

SELECT salesrep\_id

FROM order\_master

GROUP BY salesrep\_id

HAVING COUNT(order\_code) <= 10

)

GROUP BY s.salesrep\_name, om.order\_code;

\*/

-- C. For sales reps with salary > 45000:

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS GetHighSalarySalesRepOrders;

DELIMITER $$

CREATE PROCEDURE GetHighSalarySalesRepOrders()

BEGIN

SELECT sr.salesrep\_name, om.order\_code, IFNULL(SUM(p.price \* od.qty\_ordered), 0) AS bill\_value, COUNT(DISTINCT od.product\_code) AS no\_of\_products

FROM salesrep sr

JOIN order\_master om

ON sr.salesrep\_id = om.salesrep\_id

JOIN order\_detail od

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

JOIN product p

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

WHERE sr.salary > 45000

GROUP BY sr.salesrep\_name, om.order\_code;

END $$

DELIMITER ;

CALL GetHighSalarySalesRepOrders();

-- Method 2: Using JOIN

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;

-- Method 3: Using 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;

-- Method 4: Using NOT IN

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.salesrep\_id NOT IN (

SELECT salesrep\_id

FROM salesrep

WHERE salary <= 45000

)

GROUP BY s.salesrep\_name, om.order\_code;

-- Method 5: Using EXISTS (Correlated 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 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 EXISTS (

SELECT 1

FROM salesrep s2

WHERE s2.salesrep\_id = s.salesrep\_id AND s2.salary > 45000

)

GROUP BY s.salesrep\_name, om.order\_code;

-- Method 6: Using UNION (SET operators)

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

UNION

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;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

-- All sales data

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

EXCEPT

-- Remove sales reps with salary ≤ 45000

SELECT s.salesrep\_name, om.order\_code, SUM(p.price \* od.qty\_ordered), SUM(od.qty\_ordered)

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;

-- Method 8: Using INTERSECT (SET operators)

-- All sales data

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

INTERSECT

-- Only sales reps with salary > 45000

SELECT s.salesrep\_name, om.order\_code, SUM(p.price \* od.qty\_ordered), SUM(od.qty\_ordered)

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;

\*/

-- 37. Who is the senior-most sales rep?

-- Method 1: Using RANK function

SELECT salesrep\_id, salesrep\_name, salesrep\_doj

FROM (

SELECT salesrep\_id, salesrep\_name, salesrep\_doj,

RANK() OVER(ORDER BY salesrep\_doj ASC) AS rnk

FROM salesrep

) ranked

WHERE rnk = 1;

-- Method 2: Using Subquery

SELECT salesrep\_id, salesrep\_name, salesrep\_doj

FROM salesrep

WHERE salesrep\_doj = (

SELECT MIN(salesrep\_doj)

FROM salesrep

);

-- Method 3: Using NOT IN

SELECT salesrep\_id, salesrep\_name, salesrep\_doj

FROM salesrep

WHERE salesrep\_doj NOT IN (

SELECT salesrep\_doj

FROM salesrep

WHERE salesrep\_doj > (

SELECT MIN(salesrep\_doj)

FROM salesrep

)

);

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

SELECT s1.salesrep\_id, s1.salesrep\_name, s1.salesrep\_doj

FROM salesrep s1

WHERE NOT EXISTS (

SELECT 1

FROM salesrep s2

WHERE s2.salesrep\_doj < s1.salesrep\_doj

);

-- Method 5: Using JOIN

SELECT s.salesrep\_id, s.salesrep\_name, s.salesrep\_doj

FROM salesrep s

JOIN (

SELECT MIN(salesrep\_doj) AS min\_doj

FROM salesrep

) t

ON s.salesrep\_doj = t.min\_doj;

-- Method 6: Using UNION (SET operators)

-- Direct MIN approach

SELECT salesrep\_id, salesrep\_name, salesrep\_doj

FROM salesrep

WHERE salesrep\_doj = (

SELECT MIN(salesrep\_doj)

FROM salesrep

)

UNION

-- NOT EXISTS approach

SELECT s.salesrep\_id, s.salesrep\_name, s.salesrep\_doj

FROM salesrep s

WHERE NOT EXISTS (

SELECT 1

FROM salesrep s2

WHERE s2.salesrep\_doj < s.salesrep\_doj

);

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

-- All sales reps with their join dates

SELECT salesrep\_id, salesrep\_name, salesrep\_doj

FROM salesrep

EXCEPT

-- Remove sales reps who joined after someone else

SELECT s1.salesrep\_id, s1.salesrep\_name, s1.salesrep\_doj

FROM salesrep s1

JOIN salesrep s2

ON s2.salesrep\_doj < s1.salesrep\_doj;

-- Method 8: Using INTERSECT (SET operators)

-- All sales reps

SELECT salesrep\_id, salesrep\_name, salesrep\_doj

FROM salesrep

INTERSECT

-- Only those with the earliest join date

SELECT salesrep\_id, salesrep\_name, salesrep\_doj

FROM salesrep

WHERE salesrep\_doj = (

SELECT MIN(salesrep\_doj)

FROM salesrep

);

\*/

-- 38. Find the number of orders processed by the junior-most sales rep

-- Method 1: Using DENSE\_RANK function

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;

-- Method 2: Using RANK function

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;

-- Method 3: Using NOT IN

SELECT s.salesrep\_id, s.salesrep\_name, COUNT(o.order\_code) AS no\_of\_orders

FROM salesrep s

LEFT JOIN order\_master o

ON s.salesrep\_id = o.salesrep\_id

WHERE s.salesrep\_doj NOT IN (

SELECT s1.salesrep\_doj

FROM salesrep s1

WHERE s1.salesrep\_doj < (

SELECT MAX(s2.salesrep\_doj)

FROM salesrep s2

)

)

GROUP BY s.salesrep\_id, s.salesrep\_name;

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

SELECT s.salesrep\_id, s.salesrep\_name, COUNT(o.order\_code) AS no\_of\_orders

FROM salesrep s

LEFT JOIN order\_master o

ON s.salesrep\_id = o.salesrep\_id

WHERE NOT EXISTS (

SELECT 1

FROM salesrep s2

WHERE s2.salesrep\_doj > s.salesrep\_doj

)

GROUP BY s.salesrep\_id, s.salesrep\_name;

-- Method 5: Using JOIN

SELECT s.salesrep\_id, s.salesrep\_name, COUNT(o.order\_code) AS no\_of\_orders

FROM salesrep s

LEFT JOIN order\_master o

ON s.salesrep\_id = o.salesrep\_id

JOIN (

SELECT MAX(salesrep\_doj) AS max\_doj

FROM salesrep

) latest

ON s.salesrep\_doj = latest.max\_doj

GROUP BY s.salesrep\_id, s.salesrep\_name;

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

-- First get the maximum DOJ

WITH latest\_doj AS (

SELECT MAX(salesrep\_doj) AS max\_doj

FROM salesrep

)

-- Then find all sales reps with that DOJ and their order counts

SELECT s.salesrep\_id, s.salesrep\_name, COUNT(o.order\_code) AS no\_of\_orders

FROM salesrep s

LEFT JOIN order\_master o

ON s.salesrep\_id = o.salesrep\_id

JOIN latest\_doj

ON s.salesrep\_doj = latest\_doj.max\_doj

GROUP BY s.salesrep\_id, s.salesrep\_name

UNION ALL

-- Add an empty row to ensure the query always returns at least one row

SELECT NULL, NULL, 0

WHERE NOT EXISTS (

SELECT 1

FROM salesrep s

JOIN latest\_doj

ON s.salesrep\_doj = latest\_doj.max\_doj

);

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

-- Get all sales reps who are NOT junior-most (i.e., have someone with a later DOJ)

SELECT s.salesrep\_id, s.salesrep\_name, COUNT(o.order\_code) AS no\_of\_orders

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

GROUP BY s.salesrep\_id, s.salesrep\_name, s.salesrep\_doj

EXCEPT

-- Exclude them from the full list to get the junior-most rep(s)

SELECT s.salesrep\_id, s.salesrep\_name, 0

FROM salesrep s

WHERE EXISTS (

SELECT 1

FROM salesrep s2

WHERE s2.salesrep\_doj > s.salesrep\_doj

);

-- Method 8: Using INTERSECT (SET operators)

-- Get all sales reps with the latest DOJ

SELECT s.salesrep\_id, s.salesrep\_name, COUNT(o.order\_code) AS no\_of\_orders

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

INTERSECT

-- Intersect with the full list to filter only those reps

SELECT s.salesrep\_id, s.salesrep\_name, 0

FROM salesrep s

WHERE s.salesrep\_doj = (

SELECT MAX(salesrep\_doj)

FROM salesrep

);

\*/

-- 39. Find the number of orders processed by each month in 2010

-- Method 1: Using 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);

-- Method 2: Using EXTRACT

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;

-- Method 3: Using NOT IN

SELECT COUNT(order\_code) AS no\_of\_orders, MONTH(order\_date) AS monthly\_processing

FROM order\_master

WHERE YEAR(order\_date) = 2010

AND MONTH(order\_date) NOT IN (0)

GROUP BY MONTH(order\_date);

-- Method 4: Using EXISTS (Correlated Subquery)

SELECT COUNT(o.order\_code) AS no\_of\_orders, MONTH(o.order\_date) AS monthly\_processing

FROM order\_master o

WHERE EXISTS (

SELECT 1

FROM order\_master o2

WHERE YEAR(o2.order\_date) = 2010

AND MONTH(o2.order\_date) = MONTH(o.order\_date)

)

GROUP BY MONTH(o.order\_date);

-- Method 5: Using JOIN

SELECT COUNT(o.order\_code) AS no\_of\_orders, MONTH(o.order\_date) AS monthly\_processing

FROM order\_master o

JOIN (

SELECT DISTINCT MONTH(order\_date) AS month

FROM order\_master

WHERE YEAR(order\_date) = 2010

) m

ON MONTH(o.order\_date) = m.month

WHERE YEAR(o.order\_date) = 2010

GROUP BY MONTH(o.order\_date);

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

SELECT COUNT(order\_code) AS no\_of\_orders, 1 AS monthly\_processing

FROM order\_master

WHERE YEAR(order\_date) = 2010 AND MONTH(order\_date) = 1

UNION ALL

SELECT COUNT(order\_code), 2

FROM order\_master

WHERE YEAR(order\_date) = 2010 AND MONTH(order\_date) = 2

UNION ALL

SELECT COUNT(order\_code), 3

FROM order\_master

WHERE YEAR(order\_date) = 2010 AND MONTH(order\_date) = 3

UNION ALL

SELECT COUNT(order\_code), 4

FROM order\_master

WHERE YEAR(order\_date) = 2010 AND MONTH(order\_date) = 4

UNION ALL

SELECT COUNT(order\_code), 5

FROM order\_master

WHERE YEAR(order\_date) = 2010 AND MONTH(order\_date) = 5

UNION ALL

SELECT COUNT(order\_code), 6

FROM order\_master

WHERE YEAR(order\_date) = 2010 AND MONTH(order\_date) = 6

UNION ALL

SELECT COUNT(order\_code), 7

FROM order\_master

WHERE YEAR(order\_date) = 2010 AND MONTH(order\_date) = 7

UNION ALL

SELECT COUNT(order\_code), 8

FROM order\_master

WHERE YEAR(order\_date) = 2010 AND MONTH(order\_date) = 8

UNION ALL

SELECT COUNT(order\_code), 9

FROM order\_master

WHERE YEAR(order\_date) = 2010 AND MONTH(order\_date) = 9

UNION ALL

SELECT COUNT(order\_code), 10

FROM order\_master

WHERE YEAR(order\_date) = 2010 AND MONTH(order\_date) = 10

UNION ALL

SELECT COUNT(order\_code), 11

FROM order\_master

WHERE YEAR(order\_date) = 2010 AND MONTH(order\_date) = 11

UNION ALL

SELECT COUNT(order\_code), 12

FROM order\_master

WHERE YEAR(order\_date) = 2010 AND MONTH(order\_date) = 12;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

-- Orders from 2010 EXCEPT orders from other years (redundant but demonstrates EXCEPT)

SELECT MONTH(order\_date) AS monthly\_processing, COUNT(order\_code) AS no\_of\_orders

FROM order\_master

WHERE YEAR(order\_date) = 2010

GROUP BY MONTH(order\_date)

EXCEPT

-- Exclude any months that somehow appear in both sets

SELECT MONTH(order\_date), 0

FROM order\_master

WHERE YEAR(order\_date) != 2010

GROUP BY MONTH(order\_date)

ORDER BY monthly\_processing;

-- Method 8: Using INTERSECT (SET operators)

-- Months that have orders in 2010 INTERSECT with all possible months

SELECT month\_num AS monthly\_processing, COUNT(o.order\_code) AS no\_of\_orders

FROM (

SELECT 1 AS month\_num

UNION

SELECT 2

UNION

SELECT 3

UNION

SELECT 4

UNION

SELECT 5

UNION

SELECT 6

UNION

SELECT 7

UNION

SELECT 8

UNION

SELECT 9

UNION

SELECT 10

UNION

SELECT 11

UNION

SELECT 12

) months

LEFT JOIN order\_master o

ON MONTH(o.order\_date) = months.month\_num AND YEAR(o.order\_date) = 2010

GROUP BY month\_num

INTERSECT

SELECT MONTH(order\_date), COUNT(order\_code)

FROM order\_master

WHERE YEAR(order\_date) = 2010

GROUP BY MONTH(order\_date)

ORDER BY monthly\_processing;

\*/

-- 40. Which region has inactive sales reps?

-- Method 1: Using LEFT JOIN

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;

-- Method 2: Using NOT EXISTS (Correlated Subquery)

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

);

-- Method 3: Using NOT IN

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 s.salesrep\_id NOT IN (

SELECT DISTINCT salesrep\_id

FROM order\_master

WHERE om.salesrep\_id IS NOT NULL

);

-- Method 4: Using Subquery

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 (

SELECT COUNT(order\_code)

FROM order\_master om

WHERE om.salesrep\_id = s.salesrep\_id

) = 0;

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

-- Inactive reps (explicit column selection)

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

LEFT JOIN order\_master om ON s.salesrep\_id = om.salesrep\_id

WHERE om.order\_code IS NULL

UNION ALL

-- Placeholder with explicit NULLs

SELECT NULL AS salesrep\_id, NULL AS salesrep\_name, NULL AS region

WHERE 1=0;

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

-- Explicit column selection

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;

-- Method 7: Using INTERSECT (SET operators)

-- All sales reps (explicit columns)

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

INTERSECT

-- Force empty result set

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 1=0;

\*/

-- 41. Which region has the junior-most sales rep?

-- Method 1: Using RANK function

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

) ranked\_salesreps

WHERE rnk = 1;

-- Method 2: Using Subquery

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

);

-- Method 3: Using NOT IN

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 NOT IN (

SELECT s2.salesrep\_doj

FROM salesrep s2

WHERE s2.salesrep\_doj < (SELECT MAX(s3.salesrep\_doj) FROM salesrep s3)

);

-- Method 4: Using EXISTS (Correlated Subquery)

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 NOT EXISTS (

SELECT 1

FROM salesrep s2

WHERE s2.salesrep\_doj > s.salesrep\_doj

);

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

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

)

UNION ALL

SELECT NULL, NULL, NULL, NULL

WHERE 1=0;

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

-- All sales reps EXCEPT those who aren't junior-most

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

EXCEPT

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);

-- Method 7: Using INTERSECT (SET operators)

-- Intersect all sales reps with those having max DOJ

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

INTERSECT

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 number of orders whose bill is less than 500

-- Method 1: Using JOIN

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;

-- Method 2: Using CTE

WITH order\_bills 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 order\_bills ob

JOIN order\_master om

ON ob.order\_code = om.order\_code

GROUP BY ob.order\_code, ob.bill;

-- Method 3: Using NOT IN

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

WHERE o.order\_code NOT IN (

SELECT o2.order\_code

FROM order\_detail o2

JOIN product p2

ON o2.product\_code = p2.product\_code

GROUP BY o2.order\_code

HAVING SUM(p2.price \* o2.qty\_ordered) >= 500

)

GROUP BY o.order\_code

HAVING SUM(p.price \* o.qty\_ordered) < 500;

-- Method 4: Using EXISTS (Correlated Subquery)

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

WHERE EXISTS (

SELECT 1

FROM order\_detail o2

JOIN product p2

ON o2.product\_code = p2.product\_code

WHERE o2.order\_code = o.order\_code

GROUP BY o2.order\_code

HAVING SUM(p2.price \* o2.qty\_ordered) < 500

)

GROUP BY o.order\_code;

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

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

UNION ALL

SELECT NULL, NULL WHERE 1=0;

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

-- All orders EXCEPT those with bill >= 500

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

EXCEPT

SELECT o.order\_code, 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) >= 500;

-- Method 7: Using INTERSECT (SET operators)

-- Intersect all orders with those meeting bill < 500 condition

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

INTERSECT

SELECT o.order\_code, 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) < 500;

\*/

-- 43. Frequently ordered product by the most active customer

-- Method 1: Using RANK function

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;

-- Method 2: Using CTEs for better readability

WITH most\_active\_customer AS (

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

WHERE rnk = 1

),

product\_order\_counts 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 most\_active\_customer

)

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 product\_order\_counts

) ranked

WHERE rnk = 1;

-- Method 3: Using NOT IN

WITH most\_active\_customer AS (

SELECT customer\_code

FROM (

SELECT customer\_code,

DENSE\_RANK() OVER (ORDER BY COUNT(order\_code) DESC) AS cust\_rank

FROM order\_master

GROUP BY customer\_code

) ranked\_customers

WHERE cust\_rank = 1

),

product\_orders AS (

SELECT od.product\_code,

COUNT(od.order\_code) AS order\_count

FROM order\_detail od

JOIN order\_master om

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

WHERE om.customer\_code IN (

SELECT customer\_code

FROM most\_active\_customer

)

GROUP BY od.product\_code

),

max\_orders AS (

SELECT MAX(order\_count) AS max\_count

FROM product\_orders

)

SELECT po.product\_code, po.order\_count

FROM product\_orders po

WHERE po.order\_count = (

SELECT max\_count

FROM max\_orders

)

AND po.product\_code NOT IN (

SELECT pod.product\_code

FROM product\_orders pod

WHERE pod.order\_count < (

SELECT max\_count

FROM max\_orders

)

);

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

WITH most\_active\_customer AS (

SELECT customer\_code

FROM (

SELECT customer\_code,

DENSE\_RANK() OVER (ORDER BY COUNT(order\_code) DESC) AS cust\_rank

FROM order\_master

GROUP BY customer\_code

) ranked\_customers

WHERE cust\_rank = 1

),

product\_orders AS (

SELECT od.product\_code,

COUNT(od.order\_code) AS order\_count

FROM order\_detail od

JOIN order\_master om

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

WHERE EXISTS (

SELECT 1

FROM most\_active\_customer mac

WHERE mac.customer\_code = om.customer\_code

)

GROUP BY od.product\_code

),

max\_orders AS (

SELECT MAX(order\_count) AS max\_count

FROM product\_orders

)

SELECT po.product\_code, po.order\_count

FROM product\_orders po

WHERE EXISTS (

SELECT 1

FROM max\_orders mo

WHERE po.order\_count = mo.max\_count

)

AND NOT EXISTS (

SELECT 1

FROM product\_orders po2

JOIN max\_orders mo

ON po2.order\_count = mo.max\_count

WHERE po2.order\_count > po.order\_count

);

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

WITH most\_active\_customer AS (

SELECT customer\_code

FROM (

SELECT customer\_code,

DENSE\_RANK() OVER (ORDER BY COUNT(order\_code) DESC) AS cust\_rank

FROM order\_master

GROUP BY customer\_code

) ranked\_customers

WHERE cust\_rank = 1

),

product\_counts AS (

SELECT od.product\_code, COUNT(od.order\_code) AS order\_count

FROM order\_detail od

JOIN order\_master om

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

WHERE om.customer\_code IN (

SELECT customer\_code

FROM most\_active\_customer

)

GROUP BY od.product\_code

),

max\_count AS (

SELECT MAX(order\_count) AS max\_orders

FROM product\_counts

)

-- Products with max order count

SELECT pc.product\_code, pc.order\_count

FROM product\_counts pc

WHERE pc.order\_count = (

SELECT max\_orders

FROM max\_count

)

UNION ALL

-- Empty set to maintain column structure

SELECT NULL AS product\_code, 0 AS order\_count

WHERE 1=0;

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

WITH most\_active\_customer AS (

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

WHERE rnk = 1

)

-- All products ordered EXCEPT non-top products

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 most\_active\_customer

)

GROUP BY od.product\_code

EXCEPT

SELECT od.product\_code, COUNT(om.order\_code)

FROM order\_detail od

JOIN order\_master om

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

WHERE om.customer\_code = (

SELECT customer\_code

FROM most\_active\_customer

)

GROUP BY od.product\_code

HAVING COUNT(\*) < (

SELECT MAX(order\_count)

FROM (

SELECT COUNT(om2.order\_code) AS order\_count

FROM order\_detail od2

JOIN order\_master om2

ON od2.order\_code = om2.order\_code

WHERE om2.customer\_code = (

SELECT customer\_code

FROM most\_active\_customer

)

GROUP BY od2.product\_code

) counts

);

-- Method 7: Using INTERSECT (SET operators)

WITH most\_active\_customer AS (

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

WHERE rnk = 1

),

product\_counts AS (

SELECT od.product\_code, COUNT(om.order\_code) 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 most\_active\_customer

)

GROUP BY od.product\_code

)

-- Intersect all products with top-ordered ones

SELECT product\_code, order\_count

FROM product\_counts

INTERSECT

SELECT product\_code, order\_count

FROM (

SELECT product\_code, order\_count,

RANK() OVER (ORDER BY order\_count DESC) AS rnk

FROM product\_counts

) ranked

WHERE rnk = 1;

\*/

-- 44. Products not ordered by the active customer

-- Method 1: Using RANK function

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

)

);

-- Method 2: Using NOT EXISTS (Correlated Subquery)

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

);

-- Method 3: Using JOIN

SELECT DISTINCT p.product\_code

FROM product p

LEFT JOIN (

SELECT DISTINCT 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 order\_master

GROUP BY customer\_code

ORDER BY COUNT(order\_code) DESC

)

) AS ordered\_products

ON p.product\_code = ordered\_products.product\_code

WHERE ordered\_products.product\_code IS NULL;

-- Method 4: Using NOT IN

SELECT product\_code

FROM product

WHERE product\_code NOT IN (

SELECT product\_code FROM (

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

FROM order\_master

GROUP BY customer\_code

ORDER BY COUNT(order\_code) DESC

) AS ranked\_customers

)

) AS sub

);

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

-- Products not ordered by active customer

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,

DENSE\_RANK() OVER (ORDER BY COUNT(order\_code) DESC) AS cust\_rank

FROM order\_master

GROUP BY customer\_code

) ranked\_customers

WHERE cust\_rank = 1

)

)

UNION ALL

-- Empty set placeholder

SELECT NULL

WHERE 1=0;

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

-- All products EXCEPT those ordered by active customer

SELECT p.product\_code

FROM product p

EXCEPT

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,

DENSE\_RANK() OVER (ORDER BY COUNT(order\_code) DESC) AS cust\_rank

FROM order\_master

GROUP BY customer\_code

) ranked\_customers

WHERE cust\_rank = 1

);

-- Method 7: Using INTERSECT (SET operators)

-- All products INTERSECT with a condition that's never true

SELECT p.product\_code

FROM product p

WHERE 1=1

INTERSECT

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 (

SELECT customer\_code,

DENSE\_RANK() OVER (ORDER BY COUNT(order\_code) DESC) AS cust\_rank

FROM order\_master

GROUP BY customer\_code

) ranked\_customers

WHERE cust\_rank = 1

)

AND od.product\_code = p.product\_code

);

\*/

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

-- Method 1: Using JOIN

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;

-- Method 2: Using EXISTS (Correlated Subquery)

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

);

-- Method 3: Using NOT IN

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 p.product\_code NOT IN (

SELECT product\_code

FROM product

WHERE qty\_on\_hand >= (

SELECT MAX(qty\_ordered)

FROM order\_detail

)

);

-- Method 4: Using Subquery

SELECT order\_code, product\_code, qty\_ordered, qty\_on\_hand

FROM (

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

) AS sub

WHERE qty\_ordered > qty\_on\_hand;

-- Method 5: Using UNION (SET operators)

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

UNION

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;

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

-- All order-product combinations EXCEPT those that can be fulfilled

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

EXCEPT

-- Orders that CAN be processed (where quantity is available)

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 od.qty\_ordered <= p.qty\_on\_hand;

-- Method 7: Using INTERSECT (SET operators)

-- All orders INTERSECT with only the unfulfillable ones

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

INTERSECT

-- Only orders that exceed available quantity

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 od.qty\_ordered > p.qty\_on\_hand;

\*/

-- 46. Complete order report: Cust\_name | Cust\_address | Order\_code | Product\_id | Product\_name | Order\_date

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS get\_complete\_order\_report;

DELIMITER $$

CREATE PROCEDURE get\_complete\_order\_report()

BEGIN

SELECT c.cust\_name AS customer\_name, om.order\_code AS order\_code, p.product\_code AS product\_id, p.prod\_desc AS product\_name, om.order\_date AS order\_date

FROM order\_master om

JOIN customer c

ON om.cust\_code = c.cust\_code

JOIN order\_detail od

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

JOIN product p

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

ORDER BY om.order\_date, c.cust\_name;

END $$

DELIMITER ;

CALL GetCompleteOrderReport();

-- Method 2: Using JOIN

SELECT c.customer\_name, om.order\_code, p.product\_code, p.prod\_desc, 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;

-- Method 3: Using CTE for better organization:

WITH order\_details AS (

SELECT om.customer\_code, om.order\_code, om.order\_date, od.product\_code, p.prod\_desc

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, od.order\_code, od.product\_code, od.prod\_desc, od.order\_date

FROM customer c

JOIN order\_details od

ON c.customer\_code = od.customer\_code;

-- Method 4: Using NOT IN

SELECT c.customer\_name, om.order\_code, p.prod\_desc, p.prod\_desc, 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

WHERE c.customer\_code NOT IN (

SELECT customer\_code

FROM customer

WHERE customer\_code IS NULL

);

-- Method 5: Using EXISTS (Correlated Subquery)

SELECT c.customer\_name, om.order\_code, p.product\_code, p.prod\_desc, 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

WHERE EXISTS (

SELECT 1

FROM order\_master o

WHERE o.customer\_code = c.customer\_code

);

-- Method 6: Using UNION (SET operators)

SELECT c.customer\_name, om.order\_code, p.product\_code, p.prod\_desc, 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

UNION

SELECT c.customer\_name, om.order\_code, p.product\_code, p.prod\_desc, 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;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

-- All order details EXCEPT some subset

SELECT c.customer\_name, om.order\_code, p.product\_code, p.prod\_desc, 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

EXCEPT

SELECT c.customer\_name, om.order\_code, p.product\_code, p.prod\_desc, 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

WHERE p.discontinued = 1;

-- Method 8: Using INTERSECT (SET operators)

-- All order details INTERSECT with specific criteria

SELECT c.customer\_name, om.order\_code, p.product\_code, p.prod\_desc, 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

INTERSECT

SELECT c.customer\_name, om.order\_code, p.product\_code, p.prod\_desc, 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

GROUP BY c.customer\_name, om.order\_code, p.product\_code, p.prod\_desc, om.order\_date

HAVING SUM(p.price \* od.qty\_ordered) > 1000;

\*/

-- 48. Department wise who is the maximum salary earner?

-- Method 1: Using RANK function

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, s.salary

) ranked\_salesreps

WHERE rnk = 1;

-- Method 2: Using 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

);

-- Method 3: Using NOT IN

SELECT s.branch\_code, s.salesrep\_id, s.salesrep\_name, s.salary

FROM salesrep s

WHERE s.salary NOT IN (

SELECT salary

FROM salesrep

WHERE branch\_code = s.branch\_code AND salary < s.salary

);

-- Method 4: Using EXISTS (Correlated Subquery)

SELECT s.branch\_code, s.salesrep\_id, s.salesrep\_name, s.salary

FROM salesrep s

WHERE NOT EXISTS (

SELECT 1

FROM salesrep sr

WHERE sr.branch\_code = s.branch\_code AND sr.salary > s.salary

);

-- Method 5: Using UNION (SET operators)

SELECT s.branch\_code, s.salesrep\_id, s.salesrep\_name, s.salary

FROM salesrep s

WHERE NOT EXISTS (

SELECT 1

FROM salesrep sr

WHERE sr.branch\_code = s.branch\_code AND sr.salary > s.salary

)

UNION

SELECT s.branch\_code, s.salesrep\_id, s.salesrep\_name, s.salary

FROM salesrep s

WHERE NOT EXISTS (

SELECT 1

FROM salesrep sr

WHERE sr.branch\_code = s.branch\_code AND sr.salary > s.salary

);

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

-- All sales reps EXCEPT those who aren't top earners in their branch

SELECT s1.branch\_code, s1.salesrep\_id, s1.salesrep\_name, s1.salary

FROM salesrep s1

JOIN branch b

ON s1.branch\_code = b.branch\_code

EXCEPT

-- Sales reps who aren't top earners (exists someone with higher salary in same branch)

SELECT s1.branch\_code, s1.salesrep\_id, s1.salesrep\_name, s1.salary

FROM salesrep s1

WHERE EXISTS (

SELECT 1

FROM salesrep s2

WHERE s2.branch\_code = s1.branch\_code

AND s2.salary > s1.salary

);

-- Method 7: Using INTERSECT (SET operators)

-- All sales reps INTERSECT with top earners per branch

SELECT s.branch\_code, s.salesrep\_id, s.salesrep\_name, s.salary

FROM salesrep s

JOIN branch b

ON s.branch\_code = b.branch\_code

INTERSECT

-- Top earners (salary equals the maximum in their branch)

SELECT s.branch\_code, s.salesrep\_id, s.salesrep\_name, s.salary

FROM salesrep s

JOIN (

SELECT branch\_code, MAX(salary) AS max\_salary

FROM salesrep

GROUP BY branch\_code

) max\_sal

ON s.branch\_code = max\_sal.branch\_code AND s.salary = max\_sal.max\_salary;

\*/

-- 49. Top 10 active customers based on number of orders

-- Method 1: Using RANK function

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

) ranked\_customers

WHERE rnk <= 10;

-- Method 2: Using Subquery

SELECT customer\_code, order\_count

FROM (

SELECT customer\_code, COUNT(order\_code) AS order\_count

FROM order\_master

GROUP BY customer\_code

) ranked

WHERE order\_count >= (

SELECT MIN(order\_count) FROM (

SELECT customer\_code, COUNT(order\_code) AS order\_count

FROM order\_master

GROUP BY customer\_code

ORDER BY order\_count DESC

) temp

);

-- Method 3: Using NOT IN

SELECT customer\_code, COUNT(order\_code) AS order\_count

FROM order\_master

GROUP BY customer\_code

HAVING COUNT(order\_code) NOT IN (

SELECT COUNT(order\_code)

FROM order\_master

GROUP BY customer\_code

ORDER BY COUNT(order\_code) DESC

);

-- Method 4: Using EXISTS (Correlated Subquery)

SELECT customer\_code, COUNT(order\_code) AS order\_count

FROM order\_master om1

GROUP BY customer\_code

HAVING EXISTS (

SELECT 1

FROM order\_master om2

WHERE om2.customer\_code = om1.customer\_code

);

-- Method 5: Using UNION (SET Operators)

SELECT customer\_code, COUNT(order\_code) AS order\_count

FROM order\_master

GROUP BY customer\_code

UNION

SELECT customer\_code, COUNT(order\_code) AS order\_count

FROM order\_master

GROUP BY customer\_code;

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

-- All customers EXCEPT non-top-10 customers

SELECT customer\_code, COUNT(order\_code) AS order\_count

FROM order\_master

GROUP BY customer\_code

EXCEPT

SELECT customer\_code, COUNT(order\_code)

FROM order\_master

GROUP BY customer\_code

ORDER BY COUNT(order\_code) DESC

OFFSET 10 ROWS;

-- Method 7: Using INTERSECT (SET operators)

-- All customers INTERSECT with top-10 customers

SELECT customer\_code, COUNT(order\_code) AS order\_count

FROM order\_master

GROUP BY customer\_code

INTERSECT

SELECT customer\_code, COUNT(order\_code)

FROM order\_master

GROUP BY customer\_code

ORDER BY COUNT(order\_code) DESC

FETCH FIRST 10 ROWS ONLY;

\*/

-- 50. Top 5 orders based on bill value

-- Method 1: Using RANK function

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

) ranked\_orders

WHERE rnk <= 5;

-- Method 2: Using Subquery

SELECT order\_code, bill\_value

FROM (

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

) AS bill\_totals

WHERE bill\_value >= (

SELECT MIN(bill\_value) FROM (

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

) AS sub

);

-- Method 3: Using NOT IN

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

HAVING SUM(p.price \* od.qty\_ordered) NOT IN (

SELECT SUM(price \* qty\_ordered)

FROM order\_detail

JOIN product

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

GROUP BY order\_code

);

-- Method 4: Using EXISTS (Correlated Subquery)

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

HAVING EXISTS (

SELECT 1

FROM order\_detail od2

WHERE od2.order\_code = od.order\_code

);

-- Method 5: Using UNION (SET operators)

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

UNION

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;

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

-- All orders EXCEPT non-top-5 orders

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

EXCEPT

SELECT od.order\_code, SUM(p.price \* od.qty\_ordered)

FROM order\_detail od

JOIN product p

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

GROUP BY od.order\_code

ORDER BY SUM(p.price \* od.qty\_ordered) DESC

OFFSET 5 ROWS;

-- Method 7: Using INTERSECT (SET operators)

-- All orders INTERSECT with top-5 orders

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

INTERSECT

SELECT od.order\_code, SUM(p.price \* od.qty\_ordered)

FROM order\_detail od

JOIN product p

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

GROUP BY od.order\_code

ORDER BY SUM(p.price \* od.qty\_ordered) DESC

FETCH FIRST 5 ROWS ONLY;

\*/

-- 51. Sales rep who processed ORD11 and ORD15

-- Method 1: Using 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

);

-- Method 2: Using JOIN

SELECT DISTINCT s.salesrep\_id, s.salesrep\_name

FROM salesrep s

JOIN order\_master om

ON s.salesrep\_id = om.salesrep\_id

WHERE om.order\_code IN ('ORD11', 'ORD15')

GROUP BY s.salesrep\_id, s.salesrep\_name

HAVING COUNT(DISTINCT om.order\_code) = 2;

-- Method 3: Using NOT IN

SELECT salesrep\_id, salesrep\_name

FROM salesrep

WHERE salesrep\_id NOT IN (

SELECT salesrep\_id

FROM order\_master

WHERE order\_code NOT IN ('ORD11', 'ORD15')

);

-- Method 4: Using EXISTS (Correlated Subquery)

SELECT s.salesrep\_id, s.salesrep\_name

FROM salesrep s

WHERE EXISTS (

SELECT 1

FROM order\_master om

WHERE om.salesrep\_id = s.salesrep\_id AND om.order\_code = 'ORD11'

)

AND EXISTS (

SELECT 1

FROM order\_master om

WHERE om.salesrep\_id = s.salesrep\_id AND om.order\_code = 'ORD15'

);

-- Method 5: Using UNION

SELECT salesrep\_id, salesrep\_name FROM salesrep

WHERE salesrep\_id IN (

SELECT salesrep\_id

FROM order\_master

WHERE order\_code = 'ORD11'

UNION

SELECT salesrep\_id

FROM order\_master

WHERE order\_code = 'ORD15'

)

GROUP BY salesrep\_id, salesrep\_name

HAVING COUNT(DISTINCT salesrep\_id) = 1;

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

-- Reps who processed ORD11 EXCEPT those who didn't process ORD15

SELECT s.salesrep\_id, s.salesrep\_name

FROM salesrep s

JOIN order\_master om

ON s.salesrep\_id = om.salesrep\_id

WHERE om.order\_code = 'ORD11'

EXCEPT

SELECT s.salesrep\_id, s.salesrep\_name

FROM salesrep s

WHERE NOT EXISTS (

SELECT 1

FROM order\_master om

WHERE om.salesrep\_id = s.salesrep\_id

AND om.order\_code = 'ORD15'

);

-- Method 7: Using INTERSECT (SET operators)

-- Reps who processed ORD11 INTERSECT with those who processed ORD15

SELECT s.salesrep\_id, s.salesrep\_name

FROM salesrep s

JOIN order\_master om

ON s.salesrep\_id = om.salesrep\_id

WHERE om.order\_code = 'ORD11'

INTERSECT

SELECT s.salesrep\_id, s.salesrep\_name

FROM salesrep s

JOIN order\_master om

ON s.salesrep\_id = om.salesrep\_id

WHERE om.order\_code = 'ORD15';

\*/

-- 52. Generate report "worker works for manager"

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS WorkerManagerReport;

DELIMITER $$

CREATE PROCEDURE WorkerManagerReport()

BEGIN

SELECT CONCAT(sr.salesrep\_name, ' works for ', m.salesrep\_name) AS relationship

FROM salesrep sr

JOIN salesrep m ON sr.mgr = m.salesrep\_id;

END $$

DELIMITER ;

CALL WorkerManagerReport();

-- Method 2: Using JOIN

SELECT CONCAT(sr.salesrep\_name, ' works for ', m.salesrep\_name) AS relationship

FROM salesrep sr

JOIN salesrep m

ON sr.mgr = m.salesrep\_id;

-- Method 3: Using Subquery

SELECT CONCAT(s.salesrep\_name, ' works for ', (

SELECT salesrep\_name

FROM salesrep m

WHERE m.salesrep\_id = s.mgr)

) AS relationship

FROM salesrep s

WHERE s.mgr IS NOT NULL;

-- Method 4: Using NOT IN

SELECT CONCAT(s.salesrep\_name, ' works for ', m.salesrep\_name) AS relationship

FROM salesrep s

JOIN salesrep m

ON s.mgr = m.salesrep\_id

WHERE m.salesrep\_id NOT IN (

SELECT salesrep\_id

FROM salesrep

WHERE mgr IS NULL

);

-- Method 5: Using EXISTS (Correlated Subquery)

SELECT CONCAT(s.salesrep\_name, ' works for ', m.salesrep\_name) AS relationship

FROM salesrep s

JOIN salesrep m

ON s.mgr = m.salesrep\_id

WHERE EXISTS (

SELECT 1

FROM salesrep

WHERE salesrep\_id = s.mgr

);

-- Method 6: Using UNION (SET operators)

SELECT CONCAT(s.salesrep\_name, ' works for ', m.salesrep\_name) AS relationship

FROM salesrep s

JOIN salesrep m

ON s.mgr = m.salesrep\_id

UNION

SELECT CONCAT(s.salesrep\_name, ' works for ', m.salesrep\_name) AS relationship

FROM salesrep s

JOIN salesrep m

ON s.mgr = m.salesrep\_id;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

-- All valid worker-manager pairs EXCEPT invalid relationships

SELECT w.employee\_id AS worker\_id, w.employee\_name AS worker\_name, m.employee\_id AS manager\_id, m.employee\_name AS manager\_name

FROM employee w

JOIN employee m

ON w.manager\_id = m.employee\_id

EXCEPT

-- Exclude cases where manager\_id is null or invalid

SELECT w.employee\_id, w.employee\_name, m.employee\_id, m.employee\_name

FROM employee w

LEFT JOIN employee m

ON w.manager\_id = m.employee\_id

WHERE w.manager\_id IS NULL OR m.employee\_id IS NULL;

-- Method 8: Using INTERSECT (SET operators)

-- All employees INTERSECT with valid manager relationships

SELECT w.employee\_id AS worker\_id, w.employee\_name AS worker\_name, m.employee\_id AS manager\_id, m.employee\_name AS manager\_name

FROM employee w

JOIN employee m

ON 1=1

INTERSECT

SELECT w.employee\_id, w.employee\_name, m.employee\_id, m.employee\_name

FROM employee w

JOIN employee m

ON w.manager\_id = m.employee\_id;

\*/

-- Get all products with price between 1000 and 5000

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

FROM product

WHERE price BETWEEN 1000 AND 5000;

-- Get product names starting with 'I'

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

FROM product

WHERE prod\_desc LIKE 'I%';

-- Get product names ending with 'R'

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

FROM product

WHERE prod\_desc LIKE '%R';

-- Get product names with the term 'NIT'

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

FROM product

WHERE prod\_desc LIKE '%NIT%';

-- Get orders placed between 2007-01-01 and 2009-12-31

SELECT order\_code, order\_date, salesrep\_id, customer\_code

FROM order\_master

WHERE order\_date BETWEEN '2007-01-01' AND '2009-12-31';

-- Get count of products

SELECT COUNT(product\_code) AS total\_products

FROM product;

-- Get total quantity sold per product

SELECT p.product\_code, p.prod\_desc, SUM(od.qty\_ordered) AS total\_qty\_sold

FROM product p

JOIN order\_detail od

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

GROUP BY p.product\_code, p.prod\_desc;

-- Display all products ordered by price descending

SELECT product\_code, prod\_desc, price

FROM product

ORDER BY price DESC;

-- Display all customers sorted by name

SELECT customer\_code, customer\_name

FROM customer

ORDER BY customer\_name ASC;

-- Show product sales greater than 2000

SELECT od.product\_code, p.prod\_desc, SUM(od.qty\_ordered \* p.price) AS total\_sales

FROM order\_detail od

JOIN product p

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

GROUP BY od.product\_code, p.prod\_desc

HAVING SUM(od.qty\_ordered \* p.price) > 2000;

-- Show customers with more than 2 orders

SELECT om.customer\_code, c.customer\_name, COUNT(om.order\_code) AS num\_orders

FROM order\_master om

JOIN customer c

ON om.customer\_code = c.customer\_code

GROUP BY om.customer\_code, c.customer\_name

HAVING COUNT(om.order\_code) > 2;

-- Show branches with more than 1 sales rep

SELECT b.branch\_code, b.branch\_name, COUNT(s.salesrep\_id) AS num\_salesreps

FROM branch b

JOIN salesrep s

ON b.branch\_code = s.branch\_code

GROUP BY b.branch\_code, b.branch\_name

HAVING COUNT(s.salesrep\_id) > 1;