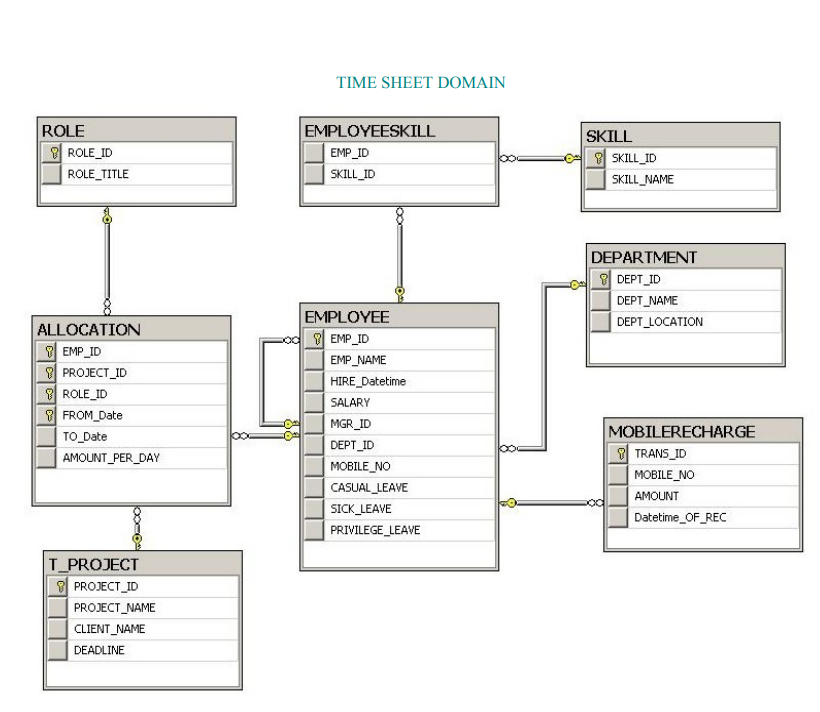
Time Sheet Management



USE TimeSheetManagement;

CREATE TABLE SKILL (

SKILL\_ID VARCHAR(10) PRIMARY KEY,

SKILL\_NAME VARCHAR(25)

);

CREATE TABLE DEPARTMENT (

DEPT\_ID VARCHAR(10) PRIMARY KEY,

DEPT\_NAME VARCHAR(25),

DEPT\_LOCATION VARCHAR(25)

);

CREATE TABLE EMPLOYEE (

EMP\_ID VARCHAR(10) PRIMARY KEY,

EMP\_NAME VARCHAR(25),

HIRE\_DATE DATE,

SALARY DECIMAL(10,2),

MGR\_ID VARCHAR(10),

DEPT\_ID VARCHAR(10),

MOBILE\_NO BIGINT UNIQUE,

CASUAL\_LEAVE INT,

SICK\_LEAVE INT,

PRIVILEGE\_LEAVE INT,

FOREIGN KEY (DEPT\_ID) REFERENCES DEPARTMENT(DEPT\_ID),

FOREIGN KEY (MGR\_ID) REFERENCES EMPLOYEE(EMP\_ID)

);

CREATE TABLE MOBILERECHARGE (

TRANS\_ID VARCHAR(10) PRIMARY KEY,

MOBILE\_NO BIGINT,

AMOUNT DECIMAL(10,2),

DATE\_OF\_REC DATE,

FOREIGN KEY (MOBILE\_NO) REFERENCES EMPLOYEE(MOBILE\_NO)

);

CREATE TABLE EMPLOYEESKILL (

EMP\_ID VARCHAR(10),

SKILL\_ID VARCHAR(10),

PRIMARY KEY (EMP\_ID, SKILL\_ID),

FOREIGN KEY (EMP\_ID) REFERENCES EMPLOYEE(EMP\_ID),

FOREIGN KEY (SKILL\_ID) REFERENCES SKILL(SKILL\_ID)

);

CREATE TABLE ROLE (

ROLE\_ID VARCHAR(10) PRIMARY KEY,

ROLE\_TITLE VARCHAR(25)

);

CREATE TABLE T\_PROJECT (

PROJECT\_ID VARCHAR(10) PRIMARY KEY,

PROJECT\_NAME VARCHAR(25),

CLIENT\_NAME VARCHAR(25),

DEADLINE DATE

);

CREATE TABLE ALLOCATION (

EMP\_ID VARCHAR(10),

PROJECT\_ID VARCHAR(10),

ROLE\_ID VARCHAR(10),

FROM\_DATE DATE,

TO\_DATE DATE,

AMOUNT\_PER\_DAY DECIMAL(10,2),

PRIMARY KEY (EMP\_ID, PROJECT\_ID, ROLE\_ID, FROM\_DATE),

FOREIGN KEY (EMP\_ID) REFERENCES EMPLOYEE(EMP\_ID),

FOREIGN KEY (PROJECT\_ID) REFERENCES T\_PROJECT(PROJECT\_ID),

FOREIGN KEY (ROLE\_ID) REFERENCES ROLE(ROLE\_ID)

);

-- Insert Data

INSERT INTO SKILL VALUES

('S01','C'), ('S02','C++'), ('S03','JAVA'), ('S04','SQL'),

('S05','PL/SQL'), ('S06','MAINFRAMES'), ('S07','.NET'), ('S08','PHP'),

('S09','TESTING'), ('S10','DB2');

INSERT INTO DEPARTMENT VALUES

('D01','ADMINISTRATION','CHENNAI'), ('D02','TRAINING','CHENNAI'),

('D03','PROJECTS','CHENNAI'), ('D04','ACCOUNTS','CHENNAI'),

('D05','DESIGNING','CHENNAI'), ('D06','DEVELOPMENT','CHENNAI'),

('D07','RESEARCH','CHENNAI');

INSERT INTO EMPLOYEE VALUES

('E01','mahesh','2006-01-01',25000,NULL,'D01',8144538983,3,2,10),

('E02','rajasekhar','2007-06-01',27000,'E01','D02',9944539983,2,4,10),

('E03','raja','2009-06-15',27000,'E01','D02',9966539983,2,3,10),

('E04','ambhani','2009-08-15',27000,'E02','D04',9254539983,2,3,10),

('E05','mukesh','2010-10-16',21000,'E02','D05',9290539983,2,3,10),

('E06','anil','2011-09-16',2000,'E03','D06',8854530983,4,5,10),

('E07','abhi','2012-09-06',20000,'E04','D04',9247530983,4,5,10),

('E08','guna','2013-01-06',20000,'E05','D06',9368530983,4,5,10),

('E09','sowmya','2013-04-01',20000,'E05','D07',9414530983,4,3,10),

('E10','anjali','2013-02-14',20000,'E06','D03',9314530983,4,3,10);

INSERT INTO MOBILERECHARGE VALUES

('T01',8144538983,300,'2007-01-01'), ('T02',9944539983,30,'2008-06-13'),

('T03',9966539983,200,'2010-01-15'), ('T04',9254539983,50,'2011-08-14'),

('T05',9290539983,100,'2012-09-21'), ('T06',8854530983,150,'2013-01-01'),

('T07',9247530983,249,'2013-02-06'), ('T08',9368530983,250,'2013-05-01'),

('T09',9414530983,300,'2013-04-25');

INSERT INTO EMPLOYEESKILL VALUES

('E01','S01'), ('E02','S02'), ('E01','S03'), ('E01','S02');

INSERT INTO ROLE VALUES

('R01','ADMIN'), ('R02','TRAINER'), ('R03','PROJECT LEAD'), ('R04','PROGRAMMER'),

('R05','ONSITE CO-ORDINATOR'), ('R06','HR'), ('R07','ACCOUNTANT'), ('R08','TESTER'),

('R09','DESIGNER'), ('R10','ANALYST');

INSERT INTO T\_PROJECT VALUES

('P01','SBI ONLINE BANKING','SBI','2011-01-12'),

('P02','MEIJYAR','ITC','2009-07-09'),

('P03','STATE FORM','AIR INDIA','2010-12-11'),

('P04','AXIS','AXIS BANK','2004-12-15'),

('P05','IBPS','IBPS','2007-12-12'),

('P06','ONLINE RESERVATION','INDIAN RAILWAYS','2007-12-01'),

('P07','HOSPITAL MAINTENANCE','VASAN','2010-08-07'),

('P08','HOSTAL MAINTENANCE','JNTU','2010-09-01');

INSERT INTO ALLOCATION VALUES

('E01','P01','R01','2010-06-20','2010-07-30',500),

('E01','P01','R02','2010-07-31','2010-09-01',1000),

('E01','P01','R03','2010-09-08','2011-09-02',2000),

('E02','P01','R03','2008-09-02','2009-01-30',2000),

('E03','P01','R03','2011-11-07','2012-12-12',2000),

('E04','P01','R01','2010-07-31','2010-12-22',500),

('E05','P02','R04','2012-01-10','2011-07-20',1500),

('E05','P02','R03','2011-07-21','2012-01-20',2000),

('E04','P02','R02','2010-12-23','2012-06-24',1000),

('E05','P03','R05','2012-01-21','2012-06-24',3000);

-- 1. Which project has the maximum number of employees?

-- Method 1: Using RANK function

SELECT project\_id, project\_name, employee\_count

FROM (

SELECT p.project\_id, p.project\_name, COUNT(a.emp\_id) AS employee\_count,

RANK() OVER (ORDER BY COUNT(a.emp\_id) DESC) AS rnk

FROM t\_project AS p

JOIN allocation a

ON p.project\_id = a.project\_id

GROUP BY p.project\_id, p.project\_name

) ranked

WHERE rnk = 1;

-- Method 2: Using HAVING

SELECT p.project\_id, p.project\_name, COUNT(a.emp\_id) AS employee\_count

FROM t\_project AS p

JOIN allocation AS a

ON p.project\_id = a.project\_id

GROUP BY p.project\_id, p.project\_name

HAVING COUNT(a.emp\_id) = (

SELECT MAX(emp\_count)

FROM (

SELECT COUNT(emp\_id) AS emp\_count

FROM allocation

GROUP BY project\_id

) counts

);

-- Method 3: Using NOT IN

SELECT p.project\_id, p.project\_name, COUNT(a.emp\_id) AS employee\_count

FROM t\_project AS p

JOIN allocation AS a

ON p.project\_id = a.project\_id

GROUP BY p.project\_id, p.project\_name

HAVING COUNT(a.emp\_id) NOT IN (

SELECT COUNT(a2.emp\_id)

FROM allocation AS a2

GROUP BY a2.project\_id

HAVING COUNT(a2.emp\_id) < (

SELECT MAX(emp\_count)

FROM (

SELECT COUNT(emp\_id) AS emp\_count

FROM allocation

GROUP BY project\_id

) max\_counts

)

);

-- Method 4: Using JOIN

SELECT p.project\_id, p.project\_name, c.employee\_count

FROM t\_project AS p

JOIN (

SELECT project\_id, COUNT(emp\_id) AS employee\_count

FROM allocation

GROUP BY project\_id

) AS c

ON c.project\_id = p.project\_id

JOIN (

SELECT MAX(emp\_cnt) AS max\_emp\_count

FROM (

SELECT COUNT(emp\_id) AS emp\_cnt

FROM allocation

GROUP BY project\_id

) AS tmp

) AS m

ON c.employee\_count = m.max\_emp\_count;

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

SELECT p.project\_id, p.project\_name, ec.employee\_count

FROM (

SELECT project\_id, COUNT(emp\_id) AS employee\_count

FROM allocation

GROUP BY project\_id

) AS ec

JOIN t\_project AS p

ON p.project\_id = ec.project\_id

WHERE NOT EXISTS (

SELECT 1

FROM (

SELECT project\_id, COUNT(emp\_id) AS employee\_count

FROM allocation

GROUP BY project\_id

) AS other

WHERE other.employee\_count > ec.employee\_count

);

-- Method 6: Using UNION (SET operators)

SELECT p.project\_id, p.project\_name, ec.employee\_count

FROM t\_project AS p

JOIN (

SELECT project\_id, COUNT(emp\_id) AS employee\_count

FROM allocation

GROUP BY project\_id

) AS ec

ON ec.project\_id = p.project\_id

WHERE ec.employee\_count = (

SELECT MAX(cnt)

FROM (

SELECT COUNT(emp\_id) AS cnt

FROM allocation

GROUP BY project\_id

) AS mx

)

UNION

SELECT p2.project\_id, p2.project\_name, ec2.employee\_count

FROM t\_project AS p2

JOIN (

SELECT project\_id, COUNT(emp\_id) AS employee\_count

FROM allocation

GROUP BY project\_id

) AS ec2

ON ec2.project\_id = p2.project\_id

WHERE ec2.employee\_count >= ALL (

SELECT COUNT(emp\_id)

FROM allocation

GROUP BY project\_id

);

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT p.project\_id, p.project\_name, ec.employee\_count

FROM t\_project AS p

JOIN (

SELECT project\_id, COUNT(emp\_id) AS employee\_count

FROM allocation

GROUP BY project\_id

) AS ec

ON ec.project\_id = p.project\_id

EXCEPT

SELECT p3.project\_id, p3.project\_name, ec3.employee\_count

FROM t\_project AS p3

JOIN (

SELECT project\_id, COUNT(emp\_id) AS employee\_count

FROM allocation

GROUP BY project\_id

) AS ec3

ON ec3.project\_id = p3.project\_id

JOIN (

SELECT COUNT(emp\_id) AS bigger\_cnt

FROM allocation

GROUP BY project\_id

) AS any\_bigger

ON any\_bigger.bigger\_cnt > ec3.employee\_count;

-- Method 8: Using INTERSECT (SET operators)

SELECT p.project\_id, p.project\_name, ec.employee\_count

FROM t\_project AS p

JOIN (

SELECT project\_id, COUNT(emp\_id) AS employee\_count

FROM allocation

GROUP BY project\_id

) AS ec

ON ec.project\_id = p.project\_id

WHERE ec.employee\_count = (

SELECT MAX(cnt)

FROM (

SELECT COUNT(emp\_id) AS cnt

FROM allocation

GROUP BY project\_id

) AS mx

)

INTERSECT

SELECT p.project\_id, p.project\_name, ec.employee\_count

FROM t\_project AS p

JOIN (

SELECT project\_id, COUNT(emp\_id) AS employee\_count

FROM allocation

GROUP BY project\_id

) AS ec

ON ec.project\_id = p.project\_id

WHERE ec.employee\_count >= ALL (

SELECT COUNT(emp\_id)

FROM allocation

GROUP BY project\_id

);

\*/

-- 2. Which employee has not yet been allocated to any project?

-- Method 1: Using Subquery

SELECT emp\_id, emp\_name

FROM employee

WHERE emp\_id NOT IN (

SELECT DISTINCT emp\_id

FROM allocation

);

-- Method 2: Using NOT IN

SELECT e.emp\_id, e.emp\_name

FROM employee AS e

WHERE e.emp\_id NOT IN (

SELECT emp\_id

FROM allocation

);

-- Method 3: Using LEFT JOIN

SELECT e.emp\_id, e.emp\_name

FROM employee AS e

LEFT JOIN allocation AS a

ON a.emp\_id = e.emp\_id

WHERE a.project\_id IS NULL;

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

SELECT e.emp\_id, e.emp\_name

FROM employee AS e

WHERE NOT EXISTS (

SELECT 1

FROM allocation AS a

WHERE a.emp\_id = e.emp\_id

);

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

SELECT emp\_id, emp\_name

FROM (

SELECT emp\_id, emp\_name, 1 AS flag

FROM employee

UNION ALL

SELECT e.emp\_id, e.emp\_name, -1 AS flag

FROM allocation AS a

JOIN employee AS e

ON e.emp\_id = a.emp\_id

) AS u

GROUP BY emp\_id, emp\_name

HAVING SUM(flag) = 1;

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT emp\_id, emp\_name

FROM employee

EXCEPT

SELECT e.emp\_id, e.emp\_name

FROM employee AS e

JOIN allocation AS a

ON a.emp\_id = e.emp\_id;

-- Method 7: Using INTERSECT (SET operators)

-- Not supported in MySQL

SELECT emp\_id, emp\_name

FROM employee

INTERSECT

SELECT emp\_id, emp\_name

FROM employee AS e

WHERE NOT EXISTS (

SELECT 1

FROM allocation AS a

WHERE a.emp\_id = e.emp\_id

);

\*/

-- 3. Which role has employee 'E03' played most frequently?

-- Method 1: Using RANK Function

SELECT role\_title, role\_id

FROM (

SELECT r.role\_title, a.role\_id, COUNT(a.role\_id) AS role\_frequency,

RANK() OVER (ORDER BY COUNT(a.role\_id) DESC) AS rnk

FROM allocation a

JOIN role r

ON a.role\_id = r.role\_id

WHERE a.emp\_id = 'E03'

GROUP BY r.role\_title, a.role\_id

) ranked

WHERE rnk = 1;

-- Method 2: Using NOT IN

SELECT r.role\_title, a.role\_id

FROM allocation a

JOIN role r

ON r.role\_id = a.role\_id

WHERE a.emp\_id = 'E03'

GROUP BY r.role\_title, a.role\_id

HAVING COUNT(\*) NOT IN (

SELECT COUNT(\*)

FROM allocation

WHERE emp\_id = 'E03'

GROUP BY role\_id

HAVING COUNT(\*) < (

SELECT MAX(cnt)

FROM (

SELECT COUNT(\*) AS cnt

FROM allocation

WHERE emp\_id = 'E03'

GROUP BY role\_id

) mx

)

);

-- Method 3: Using JOIN

SELECT r.role\_title, rc.role\_id

FROM (

SELECT role\_id, COUNT(\*) AS role\_freq

FROM allocation

WHERE emp\_id = 'E03'

GROUP BY role\_id

) rc

JOIN (

SELECT MAX(cnt) AS max\_freq

FROM (

SELECT COUNT(\*) AS cnt

FROM allocation

WHERE emp\_id = 'E03'

GROUP BY role\_id

) mx

) m

ON rc.role\_freq = m.max\_freq

JOIN role r

ON r.role\_id = rc.role\_id;

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

SELECT r.role\_title, rc.role\_id

FROM (

SELECT role\_id, COUNT(\*) AS role\_freq

FROM allocation

WHERE emp\_id = 'E03'

GROUP BY role\_id

) rc

JOIN role r

ON r.role\_id = rc.role\_id

WHERE NOT EXISTS (

SELECT 1

FROM (

SELECT role\_id, COUNT(\*) AS role\_freq

FROM allocation

WHERE emp\_id = 'E03'

GROUP BY role\_id

) other

WHERE other.role\_freq > rc.role\_freq

);

-- Method 5: Using UNION (SET operators)

SELECT r.role\_title, rc.role\_id

FROM (

SELECT role\_id, COUNT(\*) AS role\_freq

FROM allocation

WHERE emp\_id = 'E03'

GROUP BY role\_id

HAVING COUNT(\*) = (

SELECT MAX(cnt)

FROM (

SELECT COUNT(\*) AS cnt

FROM allocation

WHERE emp\_id = 'E03'

GROUP BY role\_id

) mx

)

) rc

JOIN role r

ON r.role\_id = rc.role\_id

UNION

SELECT r2.role\_title, rc2.role\_id

FROM (

SELECT role\_id, COUNT(\*) AS role\_freq

FROM allocation

WHERE emp\_id = 'E03'

GROUP BY role\_id

HAVING COUNT(\*) >= ALL (

SELECT COUNT(\*)

FROM allocation

WHERE emp\_id = 'E03'

GROUP BY role\_id

)

) rc2

JOIN role r2

ON r2.role\_id = rc2.role\_id;

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT r.role\_title, rc.role\_id

FROM (

SELECT role\_id, COUNT(\*) AS role\_freq

FROM allocation

WHERE emp\_id = 'E03'

GROUP BY role\_id

) AS rc

JOIN role AS r

ON r.role\_id = rc.role\_id

EXCEPT

SELECT r2.role\_title, rc2.role\_id

FROM (

SELECT role\_id, COUNT(\*) AS role\_freq

FROM allocation

WHERE emp\_id = 'E03'

GROUP BY role\_id

) AS rc2

JOIN role AS r2

ON r2.role\_id = rc2.role\_id

JOIN (

SELECT COUNT(\*) AS bigger\_cnt

FROM allocation

WHERE emp\_id = 'E03'

GROUP BY role\_id

) AS bigger

ON bigger.bigger\_cnt > rc2.role\_freq;

-- Method 7: Using INTERSECT (SET operators)

SELECT r.role\_title, rc.role\_id

FROM (

SELECT role\_id, COUNT(\*) AS role\_freq

FROM allocation

WHERE emp\_id = 'E03'

GROUP BY role\_id

HAVING COUNT(\*) = (

SELECT MAX(cnt)

FROM (

SELECT COUNT(\*) AS cnt

FROM allocation

WHERE emp\_id = 'E03'

GROUP BY role\_id

) AS mx

)

) AS rc

JOIN role AS r

ON r.role\_id = rc.role\_id

INTERSECT

SELECT r.role\_title, rc.role\_id

FROM (

SELECT role\_id, COUNT(\*) AS role\_freq

FROM allocation

WHERE emp\_id = 'E03'

GROUP BY role\_id

HAVING COUNT(\*) >= ALL (

SELECT COUNT(\*)

FROM allocation

WHERE emp\_id = 'E03'

GROUP BY role\_id

)

) AS rc

JOIN role AS r

ON r.role\_id = rc.role\_id;

\*/

-- 4. Which is the costliest project?

-- Method 1: Using RANK Function

SELECT project\_name, total\_cost

FROM (

SELECT p.project\_name, SUM(a.amount\_per\_day \* DATEDIFF(a.to\_date, a.from\_date)) AS total\_cost,

RANK() OVER (ORDER BY SUM(a.amount\_per\_day \* DATEDIFF(a.to\_date, a.from\_date)) DESC) AS rnk

FROM allocation a

JOIN t\_project p

ON a.project\_id = p.project\_id

GROUP BY p.project\_name

) ranked

WHERE rnk = 1;

-- Method 2: Using NOT IN

SELECT p.project\_id, p.project\_name, SUM(a.amount\_per\_day \* DATEDIFF(a.to\_date, a.from\_date)) AS total\_cost

FROM t\_project p

JOIN allocation a

ON a.project\_id = p.project\_id

GROUP BY p.project\_id, p.project\_name

HAVING SUM(a.amount\_per\_day \* DATEDIFF(a.to\_date, a.from\_date)) NOT IN (

SELECT SUM(a2.amount\_per\_day \* DATEDIFF(a2.to\_date, a2.from\_date))

FROM allocation a2

GROUP BY a2.project\_id

HAVING SUM(a2.amount\_per\_day \* DATEDIFF(a2.to\_date, a2.from\_date)) < (

SELECT MAX(cost\_val)

FROM (

SELECT SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date)) AS cost\_val

FROM allocation

GROUP BY project\_id

) mx

)

);

-- Method 3: Using JOIN

SELECT p.project\_id, p.project\_name, c.total\_cost

FROM (

SELECT project\_id, SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date)) AS total\_cost

FROM allocation

GROUP BY project\_id

) c

JOIN t\_project p

ON p.project\_id = c.project\_id

JOIN (

SELECT MAX(max\_cost) AS max\_cost

FROM (

SELECT SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date)) AS max\_cost

FROM allocation

GROUP BY project\_id

) mx

) m

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

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

SELECT p.project\_id, p.project\_name, c.total\_cost

FROM (

SELECT project\_id, SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date)) AS total\_cost

FROM allocation

GROUP BY project\_id

) c

JOIN t\_project p

ON p.project\_id = c.project\_id

WHERE NOT EXISTS (

SELECT 1

FROM (

SELECT project\_id, SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date)) AS total\_cost

FROM allocation

GROUP BY project\_id

) other

WHERE other.total\_cost > c.total\_cost

);

-- Method 5: Using UNION (SET operators)

SELECT p.project\_id, p.project\_name, c.total\_cost

FROM (

SELECT project\_id, SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date)) AS total\_cost

FROM allocation

GROUP BY project\_id

HAVING SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date)) = (

SELECT MAX(cost\_val)

FROM (

SELECT SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date)) AS cost\_val

FROM allocation

GROUP BY project\_id

) mx

)

) c

JOIN t\_project p

ON p.project\_id = c.project\_id

UNION

SELECT p2.project\_id, p2.project\_name, c2.total\_cost

FROM (

SELECT project\_id, SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date)) AS total\_cost

FROM allocation

GROUP BY project\_id

HAVING SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date)) >= ALL (

SELECT SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date))

FROM allocation

GROUP BY project\_id

)

) c2

JOIN t\_project p2

ON p2.project\_id = c2.project\_id;

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT p.project\_id, p.project\_name, c.total\_cost

FROM (

SELECT project\_id, SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date)) AS total\_cost

FROM allocation

GROUP BY project\_id

) c

JOIN t\_project p

ON p.project\_id = c.project\_id

EXCEPT

SELECT p3.project\_id, p3.project\_name, c3.total\_cost

FROM (

SELECT project\_id, SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date)) AS total\_cost

FROM allocation

GROUP BY project\_id

) c3

JOIN t\_project p3

ON p3.project\_id = c3.project\_id

JOIN (

SELECT project\_id, SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date)) AS total\_cost

FROM allocation

GROUP BY project\_id

) bigger

ON bigger.total\_cost > c3.total\_cost;

-- Method 7: Using INTERSECT (SET operators)

SELECT p.project\_id, p.project\_name, c.total\_cost

FROM (

SELECT project\_id, SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date)) AS total\_cost

FROM allocation

GROUP BY project\_id

HAVING SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date)) = (

SELECT MAX(cost\_val)

FROM (

SELECT SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date)) AS cost\_val

FROM allocation

GROUP BY project\_id

) mx

)

) c

JOIN t\_project p

ON p.project\_id = c.project\_id

INTERSECT

SELECT p.project\_id, p.project\_name, c.total\_cost

FROM (

SELECT project\_id, SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date)) AS total\_cost

FROM allocation

GROUP BY project\_id

HAVING SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date)) >= ALL (

SELECT SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date))

FROM allocation

GROUP BY project\_id

)

) c

JOIN t\_project p

ON p.project\_id = c.project\_id;

\*/

-- 5. How many employees worked on the costliest project?

-- Method 1: Using RANK Function

SELECT project\_id, project\_name, employee\_count

FROM (

SELECT p.project\_id, p.project\_name, COUNT(DISTINCT a.emp\_id) AS employee\_count,

RANK() OVER (ORDER BY SUM(a.amount\_per\_day \* DATEDIFF(a.to\_date, a.from\_date)) DESC) AS rnk

FROM t\_project p

JOIN allocation a

ON p.project\_id = a.project\_id

GROUP BY p.project\_id, p.project\_name

) ranked

WHERE rnk = 1;

-- Method 2: Using NOT IN

SELECT p.project\_id, p.project\_name, COUNT(DISTINCT a.emp\_id) AS employee\_count

FROM t\_project p

JOIN allocation a

ON a.project\_id = p.project\_id

GROUP BY p.project\_id, p.project\_name

HAVING SUM(a.amount\_per\_day \* DATEDIFF(a.to\_date, a.from\_date)) NOT IN (

SELECT SUM(a2.amount\_per\_day \* DATEDIFF(a2.to\_date, a2.from\_date))

FROM allocation a2

GROUP BY a2.project\_id

HAVING SUM(a2.amount\_per\_day \* DATEDIFF(a2.to\_date, a2.from\_date)) < (

SELECT MAX(cost\_val)

FROM (

SELECT SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date)) AS cost\_val

FROM allocation

GROUP BY project\_id

) mx

)

);

-- Method 3: Using JOIN

WITH cost\_per\_project AS (

SELECT project\_id, SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date)) AS total\_cost

FROM allocation

GROUP BY project\_id

),

max\_cost AS (

SELECT MAX(total\_cost) AS mx

FROM cost\_per\_project

)

SELECT p.project\_id, p.project\_name, COUNT(DISTINCT a.emp\_id) AS employee\_count

FROM t\_project p

JOIN allocation a

ON a.project\_id = p.project\_id

JOIN cost\_per\_project cp

ON cp.project\_id = p.project\_id

JOIN max\_cost mc

ON cp.total\_cost = mc.mx

GROUP BY p.project\_id, p.project\_name;

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

WITH cost\_per\_project AS (

SELECT project\_id, SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date)) AS total\_cost

FROM allocation

GROUP BY project\_id

)

SELECT p.project\_id, p.project\_name, COUNT(DISTINCT a.emp\_id) AS employee\_count

FROM cost\_per\_project cp

JOIN t\_project p

ON p.project\_id = cp.project\_id

JOIN allocation a

ON a.project\_id = p.project\_id

WHERE NOT EXISTS (

SELECT 1

FROM cost\_per\_project cp2

WHERE cp2.total\_cost > cp.total\_cost

)

GROUP BY p.project\_id, p.project\_name;

-- Method 5: Using UNION (SET operators)

SELECT p.project\_id, p.project\_name, COUNT(DISTINCT a.emp\_id) AS employee\_count

FROM t\_project p

JOIN allocation a

ON a.project\_id = p.project\_id

GROUP BY p.project\_id, p.project\_name

HAVING SUM(a.amount\_per\_day \* DATEDIFF(a.to\_date, a.from\_date)) = (

SELECT MAX(cost\_val)

FROM (

SELECT SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date)) AS cost\_val

FROM allocation

GROUP BY project\_id

) mx

)

UNION

SELECT p2.project\_id, p2.project\_name, COUNT(DISTINCT a2.emp\_id) AS employee\_count

FROM t\_project p2

JOIN allocation a2

ON a2.project\_id = p2.project\_id

GROUP BY p2.project\_id, p2.project\_name

HAVING SUM(a2.amount\_per\_day \* DATEDIFF(a2.to\_date, a2.from\_date)) >= ALL (

SELECT SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date))

FROM allocation

GROUP BY project\_id

);

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

WITH cost\_per\_project AS (

SELECT project\_id,

SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date)) AS total\_cost

FROM allocation

GROUP BY project\_id

)

SELECT p.project\_id, p.project\_name, c.total\_cost

FROM t\_project p

JOIN cost\_per\_project c

ON c.project\_id = p.project\_id

EXCEPT

SELECT p3.project\_id, p3.project\_name, c3.total\_cost

FROM t\_project p3

JOIN cost\_per\_project c3

ON c3.project\_id = p3.project\_id

JOIN cost\_per\_project bigger

ON bigger.total\_cost > c3.total\_cost;

-- Method 7: Using INTERSECT (SET operators)

WITH cost\_per\_project AS (

SELECT project\_id,

SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date)) AS total\_cost

FROM allocation

GROUP BY project\_id

),

max\_cost AS (

SELECT MAX(total\_cost) AS mx FROM cost\_per\_project

)

SELECT p.project\_id, p.project\_name, c.total\_cost

FROM t\_project p

JOIN cost\_per\_project c

ON c.project\_id = p.project\_id

JOIN max\_cost m

ON c.total\_cost = m.mx

INTERSECT

SELECT p.project\_id, p.project\_name, c.total\_cost

FROM t\_project p

JOIN cost\_per\_project c

ON c.project\_id = p.project\_id

WHERE NOT EXISTS (

SELECT 1

FROM cost\_per\_project other

WHERE other.total\_cost > c.total\_cost

);

\*/

-- 6. Which is the cheapest project in the year 2012?

-- Method 1: Using RANK Function

SELECT project\_id, project\_name

FROM (

SELECT p.project\_id, p.project\_name, SUM(a.amount\_per\_day \* DATEDIFF(a.to\_date, a.from\_date)) AS total\_cost,

RANK() OVER (ORDER BY SUM(a.amount\_per\_day \* DATEDIFF(a.to\_date, a.from\_date)) ASC) AS rnk

FROM t\_project p

JOIN allocation a

ON p.project\_id = a.project\_id

WHERE YEAR(a.to\_date) = 2012

GROUP BY p.project\_id, p.project\_name

) ranked

WHERE rnk = 1;

-- Method 2: Using NOT IN

SELECT p.project\_id, p.project\_name, SUM(a.amount\_per\_day \* DATEDIFF(a.to\_date, a.from\_date)) AS total\_cost

FROM t\_project p

JOIN allocation a

ON a.project\_id = p.project\_id

WHERE YEAR(a.to\_date) = 2012

GROUP BY p.project\_id, p.project\_name

HAVING SUM(a.amount\_per\_day \* DATEDIFF(a.to\_date, a.from\_date)) NOT IN (

SELECT SUM(a2.amount\_per\_day \* DATEDIFF(a2.to\_date, a2.from\_date))

FROM allocation a2

WHERE YEAR(a2.to\_date) = 2012

GROUP BY a2.project\_id

HAVING SUM(a2.amount\_per\_day \* DATEDIFF(a2.to\_date, a2.from\_date)) > (

SELECT MIN(cost\_val)

FROM (

SELECT SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date)) AS cost\_val

FROM allocation

WHERE YEAR(to\_date) = 2012

GROUP BY project\_id

) mx

)

);

-- Method 3: Using JOIN

WITH cost12 AS (

SELECT project\_id, SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date)) AS total\_cost

FROM allocation

WHERE YEAR(to\_date) = 2012

GROUP BY project\_id

),

min12 AS (

SELECT MIN(total\_cost) AS mn

FROM cost12

)

SELECT p.project\_id, p.project\_name, c.total\_cost

FROM t\_project p

JOIN cost12 c

ON c.project\_id = p.project\_id

JOIN min12 m

ON c.total\_cost = m.mn;

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

WITH cost12 AS (

SELECT project\_id, SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date)) AS total\_cost

FROM allocation

WHERE YEAR(to\_date) = 2012

GROUP BY project\_id

)

SELECT p.project\_id, p.project\_name, c.total\_cost

FROM cost12 c

JOIN t\_project p

ON p.project\_id = c.project\_id

WHERE NOT EXISTS (

SELECT 1

FROM cost12 other

WHERE other.total\_cost < c.total\_cost

);

-- Method 5: Using UNION (SET operators)

SELECT p.project\_id, p.project\_name, SUM(a.amount\_per\_day \* DATEDIFF(a.to\_date, a.from\_date)) AS total\_cost

FROM t\_project p

JOIN allocation a

ON a.project\_id = p.project\_id

WHERE YEAR(a.to\_date) = 2012

GROUP BY p.project\_id, p.project\_name

HAVING SUM(a.amount\_per\_day \* DATEDIFF(a.to\_date, a.from\_date)) = (

SELECT MIN(cost\_val)

FROM (

SELECT SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date)) AS cost\_val

FROM allocation

WHERE YEAR(to\_date) = 2012

GROUP BY project\_id

) mx

)

UNION

SELECT p2.project\_id, p2.project\_name, SUM(a2.amount\_per\_day \* DATEDIFF(a2.to\_date, a2.from\_date)) AS total\_cost

FROM t\_project p2

JOIN allocation a2

ON a2.project\_id = p2.project\_id

WHERE YEAR(a2.to\_date) = 2012

GROUP BY p2.project\_id, p2.project\_name

HAVING SUM(a2.amount\_per\_day \* DATEDIFF(a2.to\_date, a2.from\_date)) <= ALL (

SELECT SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date))

FROM allocation

WHERE YEAR(to\_date) = 2012

GROUP BY project\_id

);

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

WITH cost12 AS (

SELECT project\_id, SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date)) AS total\_cost

FROM allocation

WHERE YEAR(to\_date) = 2012

GROUP BY project\_id

)

SELECT p.project\_id, p.project\_name, c.total\_cost

FROM t\_project p

JOIN cost12 c

ON c.project\_id = p.project\_id

EXCEPT

SELECT p3.project\_id, p3.project\_name, c3.total\_cost

FROM t\_project p3

JOIN cost12 c3

ON c3.project\_id = p3.project\_id

JOIN cost12 smaller

ON smaller.total\_cost < c3.total\_cost;

-- Method 7: Using INTERSECT (SET operators)

WITH cost12 AS (

SELECT project\_id, SUM(amount\_per\_day \* DATEDIFF(to\_date, from\_date)) AS total\_cost

FROM allocation

WHERE YEAR(to\_date) = 2012

GROUP BY project\_id

),

mn AS (

SELECT MIN(total\_cost) AS mn FROM cost12

)

SELECT p.project\_id, p.project\_name, c.total\_cost

FROM t\_project p

JOIN cost12 c

ON c.project\_id = p.project\_id

JOIN mn m

ON c.total\_cost = m.mn

INTERSECT

SELECT p.project\_id, p.project\_name, c.total\_cost

FROM t\_project p

JOIN cost12 c

ON c.project\_id = p.project\_id

WHERE NOT EXISTS (

SELECT 1

FROM cost12 other

WHERE other.total\_cost < c.total\_cost

);

\*/

-- 7. What is the salary of the employee who played the maximum roles in Project 'P07'?

-- Method 1: Using RANK Function

SELECT emp\_name, salary

FROM (

SELECT e.emp\_id, e.emp\_name, e.salary, COUNT(a.role\_id) AS role\_count,

RANK() OVER (ORDER BY COUNT(a.role\_id) DESC) AS rnk

FROM employee e

JOIN allocation a

ON e.emp\_id = a.emp\_id

WHERE a.project\_id = 'P07'

GROUP BY e.emp\_id, e.emp\_name, e.salary

) ranked

WHERE rnk = 1;

-- Method 2: Using NOT IN

SELECT e.emp\_name, e.salary

FROM employee e

JOIN allocation a

ON a.emp\_id = e.emp\_id

WHERE a.project\_id = 'P07'

GROUP BY e.emp\_id, e.emp\_name, e.salary

HAVING COUNT(a.role\_id) NOT IN (

SELECT COUNT(a2.role\_id)

FROM allocation a2

WHERE a2.project\_id = 'P07'

GROUP BY a2.emp\_id

HAVING COUNT(a2.role\_id) > (

SELECT MAX(role\_cnt)

FROM (

SELECT COUNT(role\_id) AS role\_cnt

FROM allocation

WHERE project\_id = 'P07'

GROUP BY emp\_id

) mx

)

);

-- Method 3: Using JOIN

WITH role\_count AS (

SELECT emp\_id, COUNT(role\_id) AS rc

FROM allocation

WHERE project\_id = 'P07'

GROUP BY emp\_id

),

max\_rc AS (

SELECT MAX(rc) AS mx FROM role\_count

)

SELECT e.emp\_name, e.salary

FROM role\_count r

JOIN max\_rc m

ON r.rc = m.mx

JOIN employee e

ON e.emp\_id = r.emp\_id;

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

WITH role\_count AS (

SELECT emp\_id, COUNT(role\_id) AS rc

FROM allocation

WHERE project\_id = 'P07'

GROUP BY emp\_id

)

SELECT e.emp\_name, e.salary

FROM role\_count r

JOIN employee e

ON e.emp\_id = r.emp\_id

WHERE NOT EXISTS (

SELECT 1

FROM role\_count other

WHERE other.rc > r.rc

);

-- Method 5: Using UNION (SET operators)

SELECT e.emp\_name, e.salary

FROM employee e

JOIN allocation a

ON a.emp\_id = e.emp\_id

WHERE a.project\_id = 'P07'

GROUP BY e.emp\_id, e.emp\_name, e.salary

HAVING COUNT(a.role\_id) = (

SELECT MAX(role\_cnt)

FROM (

SELECT COUNT(role\_id) AS role\_cnt

FROM allocation

WHERE project\_id = 'P07'

GROUP BY emp\_id

) mx

)

UNION

SELECT e2.emp\_name, e2.salary

FROM employee e2

JOIN allocation a2

ON a2.emp\_id = e2.emp\_id

WHERE a2.project\_id = 'P07'

GROUP BY e2.emp\_id, e2.emp\_name, e2.salary

HAVING COUNT(a2.role\_id) >= ALL (

SELECT COUNT(role\_id)

FROM allocation

WHERE project\_id = 'P07'

GROUP BY emp\_id

);

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

WITH role\_count AS (

SELECT emp\_id, COUNT(role\_id) AS rc

FROM allocation

WHERE project\_id = 'P07'

GROUP BY emp\_id

)

SELECT e.emp\_name, e.salary

FROM role\_count r

JOIN employee e

ON e.emp\_id = r.emp\_id

EXCEPT

SELECT e3.emp\_name, e3.salary

FROM role\_count r3

JOIN employee e3

ON e3.emp\_id = r3.emp\_id

JOIN role\_count bigger

ON bigger.rc > r3.rc;

-- Method 7: Using INTERSECT (SET operators)

WITH role\_count AS (

SELECT emp\_id, COUNT(role\_id) AS rc

FROM allocation

WHERE project\_id = 'P07'

GROUP BY emp\_id

),

mx AS (

SELECT MAX(rc) AS mx FROM role\_count

)

SELECT e.emp\_name, e.salary

FROM role\_count r

JOIN employee e

ON e.emp\_id = r.emp\_id

JOIN mx m

ON r.rc = m.mx

INTERSECT

SELECT e.emp\_name, e.salary

FROM role\_count r

JOIN employee e

ON e.emp\_id = r.emp\_id

WHERE NOT EXISTS (

SELECT 1

FROM role\_count other

WHERE other.rc > r.rc

);

\*/

-- 8. How many projects are handled by senior most employee?

-- Method 1: Using RANK Function

SELECT emp\_name, project\_count

FROM (

SELECT e.emp\_id, e.emp\_name, COUNT(a.project\_id) AS project\_count,

RANK() OVER (ORDER BY e.hire\_date) AS rnk

FROM employee e

JOIN allocation a

ON e.emp\_id = a.emp\_id

GROUP BY e.emp\_id, e.emp\_name

) ranked

WHERE rnk = 1;

-- Method 2: Using NOT IN

WITH senior AS (

SELECT MIN(hire\_date) AS first\_day

FROM employee

)

SELECT e.emp\_name, COUNT(a.project\_id) AS project\_count

FROM employee e

JOIN senior s

ON e.hire\_date = s.first\_day

JOIN allocation a

ON a.emp\_id = e.emp\_id

GROUP BY e.emp\_name

HAVING e.emp\_id NOT IN (

SELECT emp\_id

FROM employee

WHERE hire\_date > (

SELECT first\_day

FROM senior

)

);

-- Method 3: Using JOIN

WITH senior AS (

SELECT emp\_id

FROM employee

WHERE hire\_date = (

SELECT MIN(hire\_date)

FROM employee

)

)

SELECT e.emp\_name, COUNT(a.project\_id) AS project\_count

FROM senior s

JOIN employee e

ON e.emp\_id = s.emp\_id

JOIN allocation a

ON a.emp\_id = e.emp\_id

GROUP BY e.emp\_name;

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

SELECT e.emp\_name, COUNT(a.project\_id) AS project\_count

FROM employee e

JOIN allocation a

ON a.emp\_id = e.emp\_id

WHERE NOT EXISTS (

SELECT 1

FROM employee newer

WHERE newer.hire\_date < e.hire\_date

)

GROUP BY e.emp\_name;

-- Method 5: Using UNION (SET operators)

WITH min\_date AS (

SELECT MIN(hire\_date) AS md

FROM employee

)

SELECT e.emp\_name, COUNT(a.project\_id) AS project\_count

FROM employee e

JOIN allocation a

ON a.emp\_id = e.emp\_id

JOIN min\_date m

ON e.hire\_date = m.md

GROUP BY e.emp\_name

UNION

SELECT e2.emp\_name, COUNT(a2.project\_id) AS project\_count

FROM employee e2

JOIN allocation a2

ON a2.emp\_id = e2.emp\_id

GROUP BY e2.emp\_name

HAVING e2.hire\_date <= ALL (

SELECT hire\_date

FROM employee

);

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT e.emp\_name, COUNT(DISTINCT a.project\_id) AS project\_count

FROM employee e

JOIN allocation a

ON e.emp\_id = a.emp\_id

WHERE e.emp\_id = (

SELECT emp\_id FROM employee

EXCEPT

SELECT emp\_id

FROM employee

WHERE hire\_date > ALL (

SELECT hire\_date

FROM employee

)

)

GROUP BY e.emp\_name;

-- Method 7: Using INTERSECT (SET operators)

SELECT e.emp\_name, COUNT(DISTINCT a.project\_id) AS project\_count

FROM employee e

JOIN allocation a

ON e.emp\_id = a.emp\_id

WHERE e.emp\_id IN (

SELECT emp\_id FROM employee

INTERSECT

SELECT emp\_id

FROM employee

WHERE hire\_date = (

SELECT MIN(hire\_date)

FROM employee

)

)

GROUP BY e.emp\_name;

\*/

-- 9. What is the total amount spent for unassigned employees?

-- Method 1: Using NOT IN

SELECT SUM(e.salary) AS total\_spent

FROM employee e

WHERE e.emp\_id NOT IN (

SELECT DISTINCT emp\_id

FROM allocation

);

-- Method 2: Using LEFT JOIN

SELECT SUM(e.salary) AS total\_spent

FROM employee e

LEFT JOIN allocation a

ON a.emp\_id = e.emp\_id

WHERE a.emp\_id IS NULL;

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

SELECT SUM(e.salary) AS total\_spent

FROM employee e

WHERE NOT EXISTS (

SELECT 1

FROM allocation a

WHERE a.emp\_id = e.emp\_id

);

-- Method 4: Using UNION (SET operators)

SELECT SUM(salary) AS total\_spent

FROM (

SELECT emp\_id, salary, 1 AS flag

FROM employee

UNION ALL

SELECT e.emp\_id, e.salary, -1 AS flag

FROM allocation a

JOIN employee e

ON e.emp\_id = a.emp\_id

) u

GROUP BY flag

HAVING flag = 1;

/\* Does not work in MySQL

-- Method 5: Using EXCEPT (SET operators)

SELECT emp\_id, salary

FROM employee

EXCEPT

SELECT e.emp\_id, e.salary

FROM employee e

JOIN allocation a

ON a.emp\_id = e.emp\_id;

-- Method 6: Using INTERSECT (SET operators)

SELECT emp\_id, salary

FROM employee

INTERSECT

SELECT emp\_id, salary

FROM employee e

WHERE NOT EXISTS (

SELECT 1

FROM allocation a

WHERE a.emp\_id = e.emp\_id

);

\*/

-- Q10. How many projects are completed till date?

-- Method 1: Using WHERE

SELECT COUNT(DISTINCT project\_id) AS completed\_projects

FROM allocation

WHERE to\_date <= CURDATE();

-- Method 2: Using NOT IN

SELECT COUNT(\*) AS completed\_projects

FROM t\_project

WHERE project\_id NOT IN (

SELECT DISTINCT project\_id

FROM allocation

WHERE to\_date > CURDATE()

);

-- Method 3: Using JOIN

SELECT COUNT(DISTINCT p.project\_id) AS completed\_projects

FROM t\_project p

JOIN allocation a

ON p.project\_id = a.project\_id

GROUP BY p.project\_id

HAVING MAX(a.to\_date) <= CURDATE();

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

SELECT COUNT(\*) AS completed\_projects

FROM t\_project p

WHERE NOT EXISTS (

SELECT 1

FROM allocation a

WHERE a.project\_id = p.project\_id

AND a.to\_date > CURDATE()

);

-- Method 5: Using UNION (SET operators)

SELECT COUNT(\*) AS completed\_projects

FROM (

SELECT project\_id

FROM allocation

WHERE to\_date <= CURDATE()

UNION

SELECT project\_id

FROM t\_project

WHERE project\_id NOT IN (

SELECT project\_id

FROM allocation

WHERE to\_date > CURDATE()

)

) completed;

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT COUNT(\*) AS completed\_projects

FROM (

SELECT project\_id

FROM allocation

EXCEPT

SELECT project\_id

FROM allocation

WHERE to\_date > CURDATE()

) completed;

-- Method 7: Using INTERSECT (SET operators)

SELECT COUNT(\*) AS completed\_projects

FROM (

SELECT project\_id

FROM allocation

WHERE to\_date <= CURDATE()

INTERSECT

SELECT project\_id

FROM allocation

) completed;

\*/

-- 11. How many employees have worked for less than 10 projects?

-- Method 1: Using HAVING

SELECT COUNT(emp\_id)

FROM (

SELECT emp\_id, COUNT(DISTINCT project\_id) AS project\_count

FROM allocation

GROUP BY emp\_id

HAVING project\_count < 10

) emp\_proj;

-- Method 2: Using NOT IN

SELECT COUNT(emp\_id)

FROM employee

WHERE emp\_id NOT IN (

SELECT emp\_id

FROM (

SELECT emp\_id

FROM allocation

GROUP BY emp\_id

HAVING COUNT(DISTINCT project\_id) >= 10

) more\_than\_10

);

-- Method 3: Using JOIN

SELECT COUNT(DISTINCT a.emp\_id)

FROM employee e

JOIN allocation a

ON e.emp\_id = a.emp\_id

GROUP BY a.emp\_id

HAVING COUNT(DISTINCT a.project\_id) < 10;

-- Method 4: Using EXISTS (Correlated Subquery)

SELECT COUNT(\*)

FROM employee e

WHERE EXISTS (

SELECT 1

FROM allocation a

WHERE e.emp\_id = a.emp\_id

GROUP BY a.emp\_id

HAVING COUNT(DISTINCT a.project\_id) < 10

);

-- Method 5: Using UNION (SET operators)

SELECT COUNT(\*)

FROM (

SELECT emp\_id

FROM allocation

GROUP BY emp\_id

HAVING COUNT(DISTINCT project\_id) < 10

UNION

SELECT emp\_id

FROM allocation

GROUP BY emp\_id

HAVING COUNT(DISTINCT project\_id) < 10

) union\_emp;

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT COUNT(\*)

FROM (

SELECT emp\_id

FROM allocation

GROUP BY emp\_id

EXCEPT

SELECT emp\_id

FROM allocation

GROUP BY emp\_id

HAVING COUNT(DISTINCT project\_id) >= 10

) less\_than\_10;

-- Method 7: Using INTERSECT (SET operators)

SELECT COUNT(\*)

FROM (

SELECT emp\_id

FROM allocation

GROUP BY emp\_id

HAVING COUNT(DISTINCT project\_id) < 10

INTERSECT

SELECT emp\_id

FROM allocation

) intersect\_result;

\*/

-- 12. How many employees are working with role 'R02' in project 'P04'?

-- Method 1: Using WHERE

SELECT COUNT(DISTINCT emp\_id)

FROM allocation

WHERE role\_id = 'R02' AND project\_id = 'P04';

-- Method 2: Using NOT IN

SELECT COUNT(DISTINCT emp\_id)

FROM allocation

WHERE emp\_id NOT IN (

SELECT emp\_id

FROM allocation

WHERE role\_id <> 'R02' OR project\_id <> 'P04'

);

-- Method 3: Using JOIN

SELECT COUNT(DISTINCT a.emp\_id)

FROM allocation a

JOIN role r

ON a.role\_id = r.role\_id

JOIN t\_project p

ON a.project\_id = p.project\_id

WHERE r.role\_id = 'R02' AND p.project\_id = 'P04';

-- Method 4: Using EXISTS (Correlated Subquery)

SELECT COUNT(\*)

FROM employee e

WHERE EXISTS (

SELECT 1

FROM allocation a

WHERE a.emp\_id = e.emp\_id

AND a.role\_id = 'R02'

AND a.project\_id = 'P04'

);

-- Method 5: Using UNION (SET operators)

SELECT COUNT(\*)

FROM (

SELECT emp\_id

FROM allocation

WHERE role\_id = 'R02' AND project\_id = 'P04'

UNION

SELECT emp\_id

FROM allocation

WHERE role\_id = 'R02' AND project\_id = 'P04'

) union\_result;

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT COUNT(\*)

FROM (

SELECT emp\_id

FROM allocation

WHERE role\_id = 'R02'

EXCEPT

SELECT emp\_id

FROM allocation

WHERE project\_id <> 'P04'

) result;

-- Method 7: Using INTERSECT (SET operators)

SELECT COUNT(\*)

FROM (

SELECT emp\_id

FROM allocation

WHERE role\_id = 'R02'

INTERSECT

SELECT emp\_id

FROM allocation

WHERE project\_id = 'P04'

) result;

\*/

-- 13. Which client has given the maximum number of projects?

-- Method 1: Using RANK Function

SELECT client\_name, project\_count

FROM (

SELECT client\_name, COUNT(project\_id) AS project\_count,

RANK() OVER (ORDER BY COUNT(project\_id) DESC) AS rnk

FROM t\_project

GROUP BY client\_name

) ranked

WHERE rnk = 1;

-- Method 2: Using NOT IN

SELECT client\_name, COUNT(project\_id) AS project\_count

FROM t\_project

GROUP BY client\_name

HAVING COUNT(project\_id) NOT IN (

SELECT COUNT(project\_id)

FROM t\_project

GROUP BY client\_name

HAVING COUNT(project\_id) < (

SELECT MAX(projects\_given)

FROM (

SELECT COUNT(project\_id) AS projects\_given

FROM t\_project

GROUP BY client\_name

) max\_proj

)

);

-- Method 3: Using JOIN

SELECT t1.client\_name, COUNT(t1.project\_id) AS project\_count

FROM t\_project t1

JOIN (

SELECT client\_name, COUNT(project\_id) AS max\_count

FROM t\_project

GROUP BY client\_name

) t2

ON t1.client\_name = t2.client\_name

GROUP BY t1.client\_name

HAVING COUNT(t1.project\_id) = (

SELECT MAX(projects\_given)

FROM (

SELECT COUNT(project\_id) AS projects\_given

FROM t\_project

GROUP BY client\_name

) max\_proj

);

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

SELECT client\_name, COUNT(project\_id) AS project\_count

FROM t\_project tp

GROUP BY client\_name

HAVING NOT EXISTS (

SELECT 1

FROM (

SELECT client\_name, COUNT(project\_id) AS proj\_count

FROM t\_project

GROUP BY client\_name

) sub

WHERE sub.proj\_count > COUNT(tp.project\_id)

);

-- Method 5: Using UNION (SET operators)

SELECT client\_name, project\_count

FROM (

SELECT client\_name, COUNT(project\_id) AS project\_count

FROM t\_project

GROUP BY client\_name

) t1

WHERE project\_count = (

SELECT MAX(project\_count)

FROM (

SELECT COUNT(project\_id) AS project\_count

FROM t\_project

GROUP BY client\_name

) t2

);

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT client\_name, COUNT(project\_id) AS project\_count

FROM t\_project

GROUP BY client\_name

EXCEPT

SELECT client\_name, COUNT(project\_id)

FROM t\_project

GROUP BY client\_name

HAVING COUNT(project\_id) < (

SELECT MAX(projects\_given)

FROM (

SELECT COUNT(project\_id) AS projects\_given

FROM t\_project

GROUP BY client\_name

) max\_proj

);

-- Method 7: Using INTERSECT (SET operators)

SELECT client\_name, COUNT(project\_id) AS project\_count

FROM t\_project

GROUP BY client\_name

INTERSECT

SELECT client\_name, COUNT(project\_id)

FROM t\_project

GROUP BY client\_name

HAVING COUNT(project\_id) = (

SELECT MAX(projects\_given)

FROM (

SELECT COUNT(project\_id) AS projects\_given

FROM t\_project

GROUP BY client\_name

) max\_proj

);

\*/

-- 14. Which employee has not been allocated to any project in the year 2010?

-- Method 1: Using Subquery

SELECT emp\_id, emp\_name

FROM employee

WHERE emp\_id NOT IN (

SELECT DISTINCT emp\_id

FROM allocation

WHERE YEAR(from\_date) = 2010 OR YEAR(to\_date) = 2010

);

-- Method 2: Using NOT IN

SELECT emp\_id, emp\_name

FROM employee

WHERE emp\_id NOT IN (

SELECT emp\_id

FROM allocation

WHERE YEAR(from\_date) = 2010 OR YEAR(to\_date) = 2010

);

-- Method 3: Using JOIN

SELECT e.emp\_id, e.emp\_name

FROM employee e

LEFT JOIN allocation a

ON e.emp\_id = a.emp\_id AND (YEAR(a.from\_date) = 2010 OR YEAR(a.to\_date) = 2010)

WHERE a.emp\_id IS NULL;

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

SELECT e.emp\_id, e.emp\_name

FROM employee e

WHERE NOT EXISTS (

SELECT 1

FROM allocation a

WHERE a.emp\_id = e.emp\_id

AND (YEAR(a.from\_date) = 2010 OR YEAR(a.to\_date) = 2010)

);

-- Method 5: Using UNION (SET operators)

SELECT emp\_id, emp\_name

FROM employee

WHERE emp\_id IN (

SELECT emp\_id FROM employee

UNION

SELECT emp\_id FROM employee

WHERE emp\_id NOT IN (

SELECT emp\_id

FROM allocation

WHERE YEAR(from\_date) = 2010 OR YEAR(to\_date) = 2010

)

)

AND emp\_id NOT IN (

SELECT emp\_id

FROM allocation

WHERE YEAR(from\_date) = 2010 OR YEAR(to\_date) = 2010

);

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT emp\_id, emp\_name

FROM employee

EXCEPT

SELECT DISTINCT e.emp\_id, e.emp\_name

FROM employee e

JOIN allocation a

ON e.emp\_id = a.emp\_id

WHERE YEAR(a.from\_date) = 2010 OR YEAR(a.to\_date) = 2010;

-- Method 7: Using INTERSECT (SET operators)

SELECT emp\_id, emp\_name

FROM employee

INTERSECT

SELECT emp\_id, emp\_name

FROM employee

WHERE emp\_id NOT IN (

SELECT emp\_id

FROM allocation

WHERE YEAR(from\_date) = 2010 OR YEAR(to\_date) = 2010

);

\*/

-- 15. Find the total number of days worked by employee 'E04' in project 'P02'?

-- Method 1: Using WHERE clause and SUM of DATEDIFF

SELECT SUM(DATEDIFF(to\_date, from\_date)) AS total\_days

FROM allocation

WHERE emp\_id = 'E04' AND project\_id = 'P02';

-- Method 2: Using NOT IN

SELECT SUM(DATEDIFF(to\_date, from\_date)) AS total\_days

FROM allocation

WHERE emp\_id = 'E04' AND project\_id = 'P02'

AND emp\_id NOT IN (

SELECT emp\_id

FROM allocation

WHERE emp\_id <> 'E04'

);

-- Method 3: Using JOIN

SELECT SUM(DATEDIFF(a.to\_date, a.from\_date)) AS total\_days

FROM allocation a

JOIN employee e

ON a.emp\_id = e.emp\_id

WHERE a.emp\_id = 'E04' AND a.project\_id = 'P02';

-- Method 4: Using EXISTS (Correlated Subquery)

SELECT SUM(DATEDIFF(a.to\_date, a.from\_date)) AS total\_days

FROM allocation a

WHERE a.emp\_id = 'E04' AND a.project\_id = 'P02'

AND EXISTS (

SELECT 1

FROM employee e

WHERE e.emp\_id = a.emp\_id

);

/\* Does not work in MySQL

-- Method 5: Using UNION (SET operators)

SELECT SUM(total\_days) AS total\_days FROM (

SELECT DATEDIFF(to\_date, from\_date) AS total\_days

FROM allocation

WHERE emp\_id = 'E04' AND project\_id = 'P02'

UNION ALL

SELECT 0

) unioned;

-- Method 6: Using EXCEPT (SET operators)

SELECT DATEDIFF(to\_date, from\_date) AS total\_days

FROM allocation

WHERE emp\_id = 'E04' AND project\_id = 'P02'

EXCEPT

SELECT 0;

-- Method 7: Using INTERSECT (SET operators)

SELECT DATEDIFF(to\_date, from\_date) AS total\_days

FROM allocation

WHERE emp\_id = 'E04' AND project\_id = 'P02'

INTERSECT

SELECT DATEDIFF(to\_date, from\_date) AS total\_days

FROM allocation

WHERE emp\_id = 'E04' AND project\_id = 'P02';

\*/

-- 16. Which project has been completed exactly on its deadline date?

-- Method 1: Using JOIN

SELECT p.project\_id, p.project\_name

FROM t\_project p

JOIN allocation a

ON p.project\_id = a.project\_id

WHERE p.deadline = a.to\_date;

-- Method 2: Using NOT IN

SELECT project\_id, project\_name

FROM t\_project

WHERE project\_id NOT IN (

SELECT project\_id

FROM allocation

WHERE to\_date <> deadline

);

-- Method 3: Using Subquery

SELECT project\_id, project\_name

FROM t\_project

WHERE deadline IN (

SELECT to\_date

FROM allocation

WHERE allocation.project\_id = t\_project.project\_id

);

-- Method 4: Using EXISTS (Correlated Subquery)

SELECT p.project\_id, p.project\_name

FROM t\_project p

WHERE EXISTS (

SELECT 1

FROM allocation a

WHERE a.project\_id = p.project\_id AND a.to\_date = p.deadline

);

/\* Does not work in MySQL

-- Method 5: Using UNION (SET operators)

SELECT project\_id, project\_name FROM (

SELECT p.project\_id, p.project\_name

FROM t\_project p

JOIN allocation a

ON p.project\_id = a.project\_id

WHERE p.deadline = a.to\_date

UNION

SELECT NULL, NULL

) combined

WHERE project\_id IS NOT NULL;

-- Method 6: Using EXCEPT (SET operators)

SELECT project\_id, project\_name

FROM t\_project

EXCEPT

SELECT project\_id, project\_name

FROM t\_project

WHERE project\_id NOT IN (

SELECT project\_id

FROM allocation

WHERE to\_date = deadline

);

-- Method 7: Using INTERSECT (SET operators)

SELECT project\_id, project\_name

FROM t\_project

INTERSECT

SELECT p.project\_id, p.project\_name

FROM t\_project p

JOIN allocation a

ON p.project\_id = a.project\_id

WHERE p.deadline = a.to\_date;

\*/

-- 17. How many employees worked on a project that exceeded the deadline?

-- Method 1: Using JOIN

SELECT COUNT(DISTINCT a.emp\_id) AS employee\_count

FROM allocation a

JOIN t\_project p

ON a.project\_id = p.project\_id

WHERE a.to\_date > p.deadline;

-- Method 2: Using NOT IN

SELECT COUNT(DISTINCT emp\_id) AS employee\_count

FROM allocation

WHERE emp\_id NOT IN (

SELECT a.emp\_id

FROM allocation a

JOIN t\_project p

ON a.project\_id = p.project\_id

WHERE a.to\_date <= p.deadline

);

-- Method 3: Using Subquery

SELECT COUNT(DISTINCT emp\_id) AS employee\_count

FROM allocation

WHERE emp\_id IN (

SELECT a.emp\_id

FROM allocation a

JOIN t\_project p

ON a.project\_id = p.project\_id

WHERE a.to\_date > p.deadline

);

-- Method 4: Using EXISTS (Correlated Subquery)

SELECT COUNT(DISTINCT a.emp\_id) AS employee\_count

FROM allocation a

WHERE EXISTS (

SELECT 1 FROM t\_project p

WHERE p.project\_id = a.project\_id AND a.to\_date > p.deadline

);

/\* Does not work in MySQL

-- Method 5: Using UNION (SET operators)

SELECT COUNT(DISTINCT emp\_id) AS employee\_count FROM (

SELECT a.emp\_id

FROM allocation a

JOIN t\_project p

ON a.project\_id = p.project\_id

WHERE a.to\_date > p.deadline

UNION

SELECT NULL

) unioned WHERE emp\_id IS NOT NULL;

-- Method 6: Using EXCEPT (SET operators)

SELECT emp\_id

FROM allocation

EXCEPT

SELECT a.emp\_id

FROM allocation a

JOIN t\_project p

ON a.project\_id = p.project\_id

WHERE a.to\_date <= p.deadline;

-- Method 7: Using INTERSECT (SET operators)

SELECT emp\_id

FROM allocation

INTERSECT

SELECT a.emp\_id

FROM allocation a

JOIN t\_project p

ON a.project\_id = p.project\_id

WHERE a.to\_date > p.deadline;

\*/

-- 18. Which project was completed the earliest?

-- Method 1: Using RANK Function

SELECT project\_id, earliest\_completion

FROM (

SELECT project\_id, MIN(to\_date) AS earliest\_completion,

RANK() OVER (ORDER BY MIN(to\_date)) AS rnk

FROM allocation

GROUP BY project\_id

) ranked

WHERE rnk = 1;

-- Method 2: Using NOT IN

SELECT project\_id, MIN(to\_date) AS earliest\_completion

FROM allocation

WHERE to\_date NOT IN (

SELECT MIN(to\_date)

FROM allocation

GROUP BY project\_id

HAVING MIN(to\_date) < (

SELECT MIN(to\_date)

FROM allocation

)

)

GROUP BY project\_id;

-- Method 3: Using Subquery

SELECT project\_id, to\_date AS earliest\_completion

FROM allocation

WHERE to\_date = (

SELECT MIN(to\_date)

FROM allocation

);

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

SELECT a.project\_id, a.to\_date AS earliest\_completion

FROM allocation a

WHERE NOT EXISTS (

SELECT 1

FROM allocation a2

WHERE a2.to\_date < a.to\_date

);

-- Method 5: Using UNION (SET operators)

SELECT project\_id, earliest\_completion FROM (

SELECT project\_id, MIN(to\_date) AS earliest\_completion

FROM allocation

GROUP BY project\_id

UNION

SELECT NULL, NULL

) AS unioned

WHERE earliest\_completion = (

SELECT MIN(min\_to\_date) FROM (

SELECT MIN(to\_date) AS min\_to\_date

FROM allocation

GROUP BY project\_id

) AS sub

);

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT project\_id, to\_date

FROM allocation

EXCEPT

SELECT project\_id, to\_date

FROM allocation

WHERE to\_date > (

SELECT MIN(to\_date)

FROM allocation

);

-- Method 7: Using INTERSECT (SET operators)

SELECT project\_id, to\_date

FROM allocation

INTERSECT

SELECT project\_id, to\_date

FROM allocation

WHERE to\_date = (

SELECT MIN(to\_date)

FROM allocation

);

\*/

-- 19. Which project has taken the maximum duration?

-- Method 1: Using RANK Function

SELECT project\_id, SUM(DATEDIFF(to\_date, from\_date)) AS total\_days

FROM allocation

GROUP BY project\_id

ORDER BY total\_days DESC

LIMIT 1;

-- Method 2: Using NOT IN

SELECT project\_id, SUM(DATEDIFF(to\_date, from\_date)) AS total\_days

FROM allocation

WHERE project\_id NOT IN (

SELECT project\_id FROM (

SELECT project\_id, SUM(DATEDIFF(to\_date, from\_date)) AS total\_days

FROM allocation

GROUP BY project\_id

HAVING total\_days > (

SELECT MAX(total\_days) FROM (

SELECT project\_id, SUM(DATEDIFF(to\_date, from\_date)) AS total\_days

FROM allocation

GROUP BY project\_id

) sub

)

) sub2

)

GROUP BY project\_id;

-- Method 3: Using Subquery

SELECT project\_id, SUM(DATEDIFF(to\_date, from\_date)) AS total\_days

FROM allocation

GROUP BY project\_id

HAVING total\_days = (

SELECT MAX(total\_duration) FROM (

SELECT project\_id, SUM(DATEDIFF(to\_date, from\_date)) AS total\_duration

FROM allocation

GROUP BY project\_id

) durations

);

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

SELECT a.project\_id, SUM(DATEDIFF(a.to\_date, a.from\_date)) AS total\_days

FROM allocation a

GROUP BY a.project\_id

HAVING NOT EXISTS (

SELECT 1

FROM allocation a2

GROUP BY a2.project\_id

HAVING SUM(DATEDIFF(a2.to\_date, a2.from\_date)) > total\_days

);

-- Method 5: Using UNION (SET operators)

SELECT project\_id, total\_days FROM (

SELECT project\_id, SUM(DATEDIFF(to\_date, from\_date)) AS total\_days

FROM allocation

GROUP BY project\_id

UNION

SELECT NULL, 0

) AS unioned

WHERE total\_days = (

SELECT MAX(total\_days) FROM (

SELECT SUM(DATEDIFF(to\_date, from\_date)) AS total\_days

FROM allocation

GROUP BY project\_id

) AS sub

);

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT project\_id, SUM(DATEDIFF(to\_date, from\_date)) AS total\_days

FROM allocation

GROUP BY project\_id

EXCEPT

SELECT project\_id, SUM(DATEDIFF(to\_date, from\_date)) AS total\_days

FROM allocation

GROUP BY project\_id

HAVING total\_days < (

SELECT MAX(total\_days) FROM (

SELECT project\_id, SUM(DATEDIFF(to\_date, from\_date)) AS total\_days

FROM allocation

GROUP BY project\_id

) sub

);

-- Method 7: Using INTERSECT (SET operators)

SELECT project\_id, SUM(DATEDIFF(to\_date, from\_date)) AS total\_days

FROM allocation

GROUP BY project\_id

INTERSECT

SELECT project\_id, MAX(total\_days) AS total\_days

FROM (

SELECT project\_id, SUM(DATEDIFF(to\_date, from\_date)) AS total\_days

FROM allocation

GROUP BY project\_id

) sub

GROUP BY project\_id;

\*/

-- 20. Prepare a report: Emp Id | Total Number of Days in Bench

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS get\_days\_in\_bench;

DELIMITER $$

CREATE PROCEDURE get\_days\_in\_bench()

BEGIN

SELECT emp\_id, SUM(DATEDIFF(CURDATE(), to\_date)) AS total\_number\_of\_days\_in\_bench

FROM employee e

LEFT JOIN allocation a

ON e.emp\_id = a.emp\_id

WHERE a.emp\_id IS NULL

GROUP BY emp\_id;

END $$

DELIMITER ;

CALL get\_days\_in\_bench();

-- Method 2: Using Subquery

SELECT emp\_id, SUM(DATEDIFF(CURDATE(), to\_date)) AS total\_number\_of\_days\_in\_bench

FROM employee

WHERE emp\_id NOT IN (

SELECT DISTINCT emp\_id

FROM allocation

)

GROUP BY emp\_id;

-- Method 3: Using NOT IN

SELECT emp\_id, SUM(DATEDIFF(CURDATE(), to\_date)) AS total\_number\_of\_days\_in\_bench

FROM employee

WHERE emp\_id NOT IN (

SELECT emp\_id FROM allocation

)

GROUP BY emp\_id;

-- Method 4: Using JOIN

SELECT e.emp\_id, SUM(DATEDIFF(CURDATE(), a.to\_date)) AS total\_number\_of\_days\_in\_bench

FROM employee e

LEFT JOIN allocation a

ON e.emp\_id = a.emp\_id

WHERE a.emp\_id IS NULL

GROUP BY e.emp\_id;

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

SELECT emp\_id, SUM(DATEDIFF(CURDATE(), to\_date)) AS total\_number\_of\_days\_in\_bench

FROM employee e

WHERE NOT EXISTS (

SELECT 1

FROM allocation a

WHERE a.emp\_id = e.emp\_id

)

GROUP BY emp\_id;

-- Method 6: Using UNION (SET operators)

SELECT emp\_id, total\_days FROM (

SELECT emp\_id, SUM(DATEDIFF(CURDATE(), to\_date)) AS total\_days

FROM employee

WHERE emp\_id NOT IN (SELECT emp\_id FROM allocation)

GROUP BY emp\_id

UNION

SELECT NULL, 0

) unioned WHERE emp\_id IS NOT NULL;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT emp\_id, SUM(DATEDIFF(CURDATE(), to\_date)) AS total\_days

FROM employee

EXCEPT

SELECT emp\_id, SUM(DATEDIFF(CURDATE(), to\_date)) AS total\_days

FROM allocation;

-- Method 8: Using INTERSECT (SET operators)

SELECT emp\_id, SUM(DATEDIFF(CURDATE(), to\_date)) AS total\_days

FROM employee

INTERSECT

SELECT emp\_id, SUM(DATEDIFF(CURDATE(), to\_date)) AS total\_days

FROM allocation;

\*/

-- 21. Prepare a report: Project Name | Number of Employees

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS get\_project\_employee\_counts;

DELIMITER $$

CREATE PROCEDURE get\_project\_employee\_counts()

BEGIN

SELECT p.project\_name, COUNT(DISTINCT a.emp\_id) AS number\_of\_employees

FROM t\_project p

LEFT JOIN allocation a

ON p.project\_id = a.project\_id

GROUP BY p.project\_name;

END $$

DELIMITER ;

CALL get\_project\_employee\_counts();

-- Method 2: Using JOIN

SELECT p.project\_name, COUNT(DISTINCT a.emp\_id) AS number\_of\_employees

FROM allocation a

JOIN t\_project p

ON a.project\_id = p.project\_id

GROUP BY p.project\_name;

-- Method 3: Using NOT IN

SELECT p.project\_name, COUNT(DISTINCT a.emp\_id) AS number\_of\_employees

FROM t\_project p

LEFT JOIN allocation a

ON p.project\_id = a.project\_id

WHERE p.project\_id NOT IN (

SELECT project\_id

FROM allocation

)

GROUP BY p.project\_name;

-- Method 4: Using Subquery

SELECT p.project\_name, (

SELECT COUNT(DISTINCT a.emp\_id)

FROM allocation a

WHERE a.project\_id = p.project\_id

) AS number\_of\_employees

FROM t\_project p;

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

SELECT p.project\_name, COUNT(DISTINCT a.emp\_id) AS number\_of\_employees

FROM t\_project p

LEFT JOIN allocation a

ON p.project\_id = a.project\_id

WHERE EXISTS (

SELECT 1

FROM allocation a2

WHERE a2.project\_id = p.project\_id

)

GROUP BY p.project\_name;

-- Method 6: Using UNION (SET operators)

SELECT project\_name, employee\_count FROM (

SELECT p.project\_name, COUNT(DISTINCT a.emp\_id) AS employee\_count

FROM t\_project p

JOIN allocation a

ON p.project\_id = a.project\_id

GROUP BY p.project\_name

UNION

SELECT NULL, 0

) unioned

WHERE project\_name IS NOT NULL;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT p.project\_name, COUNT(DISTINCT a.emp\_id) AS emp\_count

FROM t\_project p

LEFT JOIN allocation a

ON p.project\_id = a.project\_id

GROUP BY p.project\_name

EXCEPT

SELECT p.project\_name, COUNT(DISTINCT a.emp\_id) AS emp\_count

FROM allocation a

JOIN t\_project p

ON a.project\_id = p.project\_id

GROUP BY p.project\_name;

-- Method 8: Using INTERSECT (SET operators)

SELECT p.project\_name, COUNT(DISTINCT a.emp\_id) AS emp\_count

FROM t\_project p

LEFT JOIN allocation a

ON p.project\_id = a.project\_id

GROUP BY p.project\_name

INTERSECT

SELECT p.project\_name, COUNT(DISTINCT a.emp\_id) AS emp\_count

FROM allocation a

JOIN t\_project p

ON a.project\_id = p.project\_id

GROUP BY p.project\_name;

\*/

-- 22. Prepare a report: Role Name | Number of Employees

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS get\_role\_employee\_counts;

DELIMITER $$

CREATE PROCEDURE get\_role\_employee\_counts()

BEGIN

SELECT r.role\_title, COUNT(DISTINCT a.emp\_id) AS emp\_count

FROM role r

LEFT JOIN allocation a

ON r.role\_id = a.role\_id

GROUP BY r.role\_title;

END $$

DELIMITER ;

CALL get\_role\_employee\_counts();

-- Method 2: Using JOIN

SELECT r.role\_title, COUNT(DISTINCT a.emp\_id) AS emp\_count

FROM allocation a

JOIN role r

ON a.role\_id = r.role\_id

GROUP BY r.role\_title;

-- Method 3: Using NOT IN

SELECT r.role\_title, COUNT(DISTINCT a.emp\_id) AS emp\_count

FROM role r

LEFT JOIN allocation a

ON r.role\_id = a.role\_id

WHERE r.role\_id NOT IN (

SELECT role\_id FROM allocation

)

GROUP BY r.role\_title;

-- Method 4: Using Subquery

SELECT r.role\_title, (

SELECT COUNT(DISTINCT a.emp\_id)

FROM allocation a

WHERE a.role\_id = r.role\_id

) AS emp\_count

FROM role r;

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

SELECT r.role\_title, COUNT(DISTINCT a.emp\_id) AS emp\_count

FROM role r

LEFT JOIN allocation a

ON r.role\_id = a.role\_id

WHERE EXISTS (

SELECT 1 FROM allocation a2 WHERE a2.role\_id = r.role\_id

)

GROUP BY r.role\_title;

-- Method 6: Using UNION (SET operators)

SELECT role\_title, emp\_count FROM (

SELECT r.role\_title, COUNT(DISTINCT a.emp\_id) AS emp\_count

FROM role r

JOIN allocation a

ON r.role\_id = a.role\_id

GROUP BY r.role\_title

UNION

SELECT NULL, 0

) unioned

WHERE role\_title IS NOT NULL;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT r.role\_title, COUNT(DISTINCT a.emp\_id) AS emp\_count

FROM role r

LEFT JOIN allocation a

ON r.role\_id = a.role\_id

GROUP BY r.role\_title

EXCEPT

SELECT r.role\_title, COUNT(DISTINCT a.emp\_id) AS emp\_count

FROM allocation a

JOIN role r

ON a.role\_id = r.role\_id

GROUP BY r.role\_title;

-- Method 8: Using INTERSECT (SET operators)

SELECT r.role\_title, COUNT(DISTINCT a.emp\_id) AS emp\_count

FROM role r

LEFT JOIN allocation a

ON r.role\_id = a.role\_id

GROUP BY r.role\_title

INTERSECT

SELECT r.role\_title, COUNT(DISTINCT a.emp\_id) AS emp\_count

FROM allocation a

JOIN role r

ON a.role\_id = r.role\_id

GROUP BY r.role\_title;

\*/

-- 23. Prepare a report: Emp Name | Number of Projects

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS get\_emp\_project\_counts;

DELIMITER $$

CREATE PROCEDURE get\_emp\_project\_counts()

BEGIN

SELECT e.emp\_name, COUNT(DISTINCT a.project\_id) AS number\_of\_projects

FROM employee e

LEFT JOIN allocation a

ON e.emp\_id = a.emp\_id

GROUP BY e.emp\_name;

END $$

DELIMITER ;

CALL get\_emp\_project\_counts();

-- Method 2: Using JOIN

SELECT e.emp\_name, COUNT(DISTINCT a.project\_id) AS number\_of\_projects

FROM employee e

JOIN allocation a

ON e.emp\_id = a.emp\_id

GROUP BY e.emp\_name;

-- Method 3: Using NOT IN

SELECT emp\_name, (

SELECT COUNT(DISTINCT project\_id)

FROM allocation

WHERE emp\_id = e.emp\_id

) AS number\_of\_projects

FROM employee e

WHERE emp\_id IN (

SELECT emp\_id FROM allocation

)

ORDER BY emp\_name;

-- Method 4: Using Subquery

SELECT emp\_name, project\_count FROM (

SELECT e.emp\_name, (

SELECT COUNT(DISTINCT project\_id)

FROM allocation a

WHERE a.emp\_id = e.emp\_id

) AS project\_count

FROM employee e

) sub;

-- Method 5: Using EXISTS (Correlated Subquery)

SELECT e.emp\_name, COUNT(DISTINCT a.project\_id) AS number\_of\_projects

FROM employee e

LEFT JOIN allocation a

ON e.emp\_id = a.emp\_id

WHERE EXISTS (

SELECT 1

FROM allocation a2

WHERE a2.emp\_id = e.emp\_id

)

GROUP BY e.emp\_name;

-- Method 6: Using UNION (SET operators)

SELECT emp\_name, number\_of\_projects FROM (

SELECT e.emp\_name, COUNT(DISTINCT a.project\_id) AS number\_of\_projects

FROM employee e

JOIN allocation a

ON e.emp\_id = a.emp\_id

GROUP BY e.emp\_name

UNION

SELECT NULL, 0

) unioned

WHERE emp\_name IS NOT NULL;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT e.emp\_name, COUNT(DISTINCT a.project\_id) AS number\_of\_projects

FROM employee e

JOIN allocation a

ON e.emp\_id = a.emp\_id

EXCEPT

SELECT NULL, 0;

-- Method 8: Using INTERSECT (SET operators)

SELECT e.emp\_name, COUNT(DISTINCT a.project\_id) AS number\_of\_projects

FROM employee e

JOIN allocation a

ON e.emp\_id = a.emp\_id

INTERSECT

SELECT e.emp\_name, COUNT(DISTINCT a.project\_id) AS number\_of\_projects

FROM employee e

JOIN allocation a

ON e.emp\_id = a.emp\_id;

\*/

-- 24. Prepare a report: Emp Name | Number of Roles

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS get\_emp\_role\_counts;

DELIMITER $$

CREATE PROCEDURE get\_emp\_role\_counts()

BEGIN

SELECT e.emp\_name, COUNT(DISTINCT a.role\_id) AS number\_of\_roles

FROM employee e

LEFT JOIN allocation a

ON e.emp\_id = a.emp\_id

GROUP BY e.emp\_name;

END $$

DELIMITER ;

CALL get\_emp\_role\_counts();

-- Method 2: Using JOIN

SELECT e.emp\_name, COUNT(DISTINCT a.role\_id) AS number\_of\_roles

FROM employee e

JOIN allocation a

ON e.emp\_id = a.emp\_id

GROUP BY e.emp\_name;

-- Method 3: Using NOT IN

SELECT emp\_name, (

SELECT COUNT(DISTINCT role\_id)

FROM allocation

WHERE emp\_id = e.emp\_id

) AS number\_of\_roles

FROM employee e

WHERE emp\_id IN (

SELECT emp\_id FROM allocation

)

ORDER BY emp\_name;

-- Method 4: Using Subquery

SELECT emp\_name, role\_count FROM (

SELECT e.emp\_name,(

SELECT COUNT(DISTINCT role\_id)

FROM allocation a

WHERE a.emp\_id = e.emp\_id

) AS role\_count

FROM employee e

) sub;

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

SELECT e.emp\_name, COUNT(DISTINCT a.role\_id) AS number\_of\_roles

FROM employee e

LEFT JOIN allocation a

ON e.emp\_id = a.emp\_id

WHERE EXISTS (

SELECT 1

FROM allocation a2

WHERE a2.emp\_id = e.emp\_id

)

GROUP BY e.emp\_name;

-- Method 6: Using UNION (SET operators)

SELECT emp\_name, number\_of\_roles

FROM (

SELECT e.emp\_name, COUNT(DISTINCT a.role\_id) AS number\_of\_roles

FROM employee e

JOIN allocation a

ON e.emp\_id = a.emp\_id

GROUP BY e.emp\_name

UNION

SELECT NULL, 0

) unioned

WHERE emp\_name IS NOT NULL;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT e.emp\_name, COUNT(DISTINCT a.role\_id) AS number\_of\_roles

FROM employee e

JOIN allocation a

ON e.emp\_id = a.emp\_id

EXCEPT

SELECT NULL, 0;

-- Method 8: Using INTERSECT (SET operators)

SELECT e.emp\_name, COUNT(DISTINCT a.role\_id) AS number\_of\_roles

FROM employee e

JOIN allocation a

ON e.emp\_id = a.emp\_id

INTERSECT

SELECT e.emp\_name, COUNT(DISTINCT a.role\_id) AS number\_of\_roles

FROM employee e

JOIN allocation a

ON e.emp\_id = a.emp\_id;

\*/

-- 25. Prepare a report: Role Name | Number of Employees

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS get\_role\_employee\_counts;

DELIMITER $$

CREATE PROCEDURE get\_role\_employee\_counts()

BEGIN

SELECT r.role\_title, COUNT(DISTINCT a.emp\_id) AS number\_of\_employees

FROM allocation a

JOIN role r

ON a.role\_id = r.role\_id

GROUP BY r.role\_title;

END $$

DELIMITER ;

CALL get\_role\_employee\_counts();

-- Method 2: Using JOIN

SELECT r.role\_title, COUNT(DISTINCT a.emp\_id) AS number\_of\_employees

FROM allocation a

JOIN role r

ON a.role\_id = r.role\_id

GROUP BY r.role\_title;

-- Method 3: Using NOT IN

SELECT r.role\_title, (

SELECT COUNT(DISTINCT emp\_id)

FROM allocation a

WHERE a.role\_id = r.role\_id

) AS number\_of\_employees

FROM role r

WHERE r.role\_id IN (

SELECT DISTINCT role\_id

FROM allocation

)

ORDER BY r.role\_title;

-- Method 4: Using Subquery

SELECT role\_title, employee\_count FROM (

SELECT r.role\_title, (

SELECT COUNT(DISTINCT a.emp\_id)

FROM allocation a

WHERE a.role\_id = r.role\_id

) AS employee\_count

FROM role r

) sub;

-- Method 5: Using EXISTS (Correlated Subquery)

SELECT r.role\_title, COUNT(DISTINCT a.emp\_id) AS number\_of\_employees

FROM role r

JOIN allocation a

ON r.role\_id = a.role\_id

WHERE EXISTS (

SELECT 1

FROM allocation a2

WHERE a2.role\_id = r.role\_id

)

GROUP BY r.role\_title;

-- Method 6: Using UNION (SET operators)

SELECT role\_title, number\_of\_employees FROM (

SELECT r.role\_title, COUNT(DISTINCT a.emp\_id) AS number\_of\_employees

FROM allocation a

JOIN role r

ON a.role\_id = r.role\_id

GROUP BY r.role\_title

UNION

SELECT NULL, 0

) unioned

WHERE role\_title IS NOT NULL;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT r.role\_title, COUNT(DISTINCT a.emp\_id) AS number\_of\_employees

FROM allocation a

JOIN role r

ON a.role\_id = r.role\_id

EXCEPT

SELECT NULL, 0;

-- Method 8: Using INTERSECT (SET operators)

SELECT r.role\_title, COUNT(DISTINCT a.emp\_id) AS number\_of\_employees

FROM allocation a

JOIN role r

ON a.role\_id = r.role\_id

INTERSECT

SELECT r.role\_title, COUNT(DISTINCT a.emp\_id) AS number\_of\_employees

FROM allocation a

JOIN role r

ON a.role\_id = r.role\_id;

\*/

-- 26. Prepare a report in this format: Role Name | Number of Projects

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS get\_role\_project\_counts;

DELIMITER $$

CREATE PROCEDURE get\_role\_project\_counts()

BEGIN

SELECT r.role\_title, COUNT(DISTINCT a.project\_id) AS number\_of\_projects

FROM allocation a

JOIN role r

ON a.role\_id = r.role\_id

GROUP BY r.role\_title;

END $$

DELIMITER ;

CALL get\_role\_project\_counts();

-- Method 2: Using JOIN

SELECT r.role\_title, COUNT(DISTINCT a.project\_id) AS number\_of\_projects

FROM allocation a

JOIN role r

ON a.role\_id = r.role\_id

GROUP BY r.role\_title;

-- Method 3: Using NOT IN

SELECT r.role\_title, (

SELECT COUNT(DISTINCT project\_id)

FROM allocation a

WHERE a.role\_id = r.role\_id

) AS number\_of\_projects

FROM role r

WHERE r.role\_id IN (

SELECT DISTINCT role\_id FROM allocation

)

ORDER BY r.role\_title;

-- Method 4: Using Subquery

SELECT role\_title, project\_count FROM (

SELECT r.role\_title, (

SELECT COUNT(DISTINCT a.project\_id)

FROM allocation a

WHERE a.role\_id = r.role\_id

) AS project\_count

FROM role r

) sub;

-- Method 5: Using EXISTS (Correlated Subquery)

SELECT r.role\_title, COUNT(DISTINCT a.project\_id) AS number\_of\_projects

FROM role r

JOIN allocation a

ON r.role\_id = a.role\_id

WHERE EXISTS (

SELECT 1

FROM allocation a2

WHERE a2.role\_id = r.role\_id

)

GROUP BY r.role\_title;

-- Method 6: Using UNION (SET operators)

SELECT role\_title, number\_of\_projects FROM (

SELECT r.role\_title, COUNT(DISTINCT a.project\_id) AS number\_of\_projects

FROM allocation a

JOIN role r

ON a.role\_id = r.role\_id

GROUP BY r.role\_title

UNION

SELECT NULL, 0

) unioned

WHERE role\_title IS NOT NULL;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT r.role\_title, COUNT(DISTINCT a.project\_id) AS number\_of\_projects

FROM allocation a

JOIN role r

ON a.role\_id = r.role\_id

EXCEPT

SELECT NULL, 0;

-- Method 8: Using INTERSECT (SET operators)

SELECT r.role\_title, COUNT(DISTINCT a.project\_id) AS number\_of\_projects

FROM allocation a

JOIN role r

ON a.role\_id = r.role\_id

INTERSECT

SELECT r.role\_title, COUNT(DISTINCT a.project\_id) AS number\_of\_projects

FROM allocation a

JOIN role r

ON a.role\_id = r.role\_id;

\*/

-- 27. Prepare a report in this format: Emp Name | Role Name | Number of Projects

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS get\_emp\_role\_project\_counts;

DELIMITER $$

CREATE PROCEDURE get\_emp\_role\_project\_counts()

BEGIN

SELECT e.emp\_name, r.role\_title, COUNT(DISTINCT a.project\_id) AS number\_of\_projects

FROM allocation a

JOIN employee e

ON a.emp\_id = e.emp\_id

JOIN role r

ON a.role\_id = r.role\_id

GROUP BY e.emp\_name, r.role\_title;

END $$

DELIMITER ;

CALL get\_emp\_role\_project\_counts();

-- Method 2: Using JOIN

SELECT e.emp\_name, r.role\_title, COUNT(DISTINCT a.project\_id) AS number\_of\_projects

FROM allocation a

JOIN employee e

ON a.emp\_id = e.emp\_id

JOIN role r

ON a.role\_id = r.role\_id

GROUP BY e.emp\_name, r.role\_title;

-- Method 3: Using NOT IN

SELECT e.emp\_name, r.role\_title, (

SELECT COUNT(DISTINCT project\_id)

FROM allocation a

WHERE a.emp\_id = e.emp\_id AND a.role\_id = r.role\_id

) AS number\_of\_projects

FROM employee e, role r

WHERE (e.emp\_id, r.role\_id) IN (

SELECT emp\_id, role\_id

FROM allocation

)

ORDER BY e.emp\_name, r.role\_title;

-- Method 4: Using Subquery

SELECT emp\_name, role\_title, project\_count FROM (

SELECT e.emp\_name, r.role\_title,(

SELECT COUNT(DISTINCT a.project\_id)

FROM allocation a

WHERE a.emp\_id = e.emp\_id AND a.role\_id = r.role\_id

) AS project\_count

FROM employee e

CROSS JOIN role r

) sub

WHERE project\_count > 0;

-- Method 5: Using EXISTS (Correlated Subquery)

SELECT e.emp\_name, r.role\_title, COUNT(DISTINCT a.project\_id) AS number\_of\_projects

FROM employee e

JOIN role r

JOIN allocation a

ON e.emp\_id = a.emp\_id AND r.role\_id = a.role\_id

WHERE EXISTS (

SELECT 1

FROM allocation a2

WHERE a2.emp\_id = e.emp\_id AND a2.role\_id = r.role\_id

)

GROUP BY e.emp\_name, r.role\_title;

-- Method 6: Using UNION (SET operators)

SELECT emp\_name, role\_title, number\_of\_projects FROM (

SELECT e.emp\_name, r.role\_title, COUNT(DISTINCT a.project\_id) AS number\_of\_projects

FROM allocation a

JOIN employee e

ON a.emp\_id = e.emp\_id

JOIN role r

ON a.role\_id = r.role\_id

GROUP BY e.emp\_name, r.role\_title

UNION

SELECT NULL, NULL, 0

) unioned

WHERE emp\_name IS NOT NULL;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT e.emp\_name, r.role\_title, COUNT(DISTINCT a.project\_id) AS number\_of\_projects

FROM allocation a

JOIN employee e

ON a.emp\_id = e.emp\_id

JOIN role r

ON a.role\_id = r.role\_id

EXCEPT

SELECT NULL, NULL, 0;

-- Method 8: Using INTERSECT (SET operators)

SELECT e.emp\_name, r.role\_title, COUNT(DISTINCT a.project\_id) AS number\_of\_projects

FROM allocation a

JOIN employee e

ON a.emp\_id = e.emp\_id

JOIN role r

ON a.role\_id = r.role\_id

INTERSECT

SELECT e.emp\_name, r.role\_title, COUNT(DISTINCT a.project\_id) AS number\_of\_projects

FROM allocation a

JOIN employee e

ON a.emp\_id = e.emp\_id

JOIN role r

ON a.role\_id = r.role\_id;

\*/

-- 28. Prepare a report in this format: Project Name | Role Name | Number of Employees

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS get\_project\_role\_employee\_counts;

DELIMITER $$

CREATE PROCEDURE get\_project\_role\_employee\_counts()

BEGIN

SELECT p.project\_name, r.role\_title, COUNT(DISTINCT a.emp\_id) AS number\_of\_employees

FROM allocation a

JOIN t\_project p

ON a.project\_id = p.project\_id

JOIN role r

ON a.role\_id = r.role\_id

GROUP BY p.project\_name, r.role\_title;

END $$

DELIMITER ;

CALL get\_project\_role\_employee\_counts();

-- Method 2: Using JOIN

SELECT p.project\_name, r.role\_title, COUNT(DISTINCT a.emp\_id) AS number\_of\_employees

FROM allocation a

JOIN t\_project p

ON a.project\_id = p.project\_id

JOIN role r

ON a.role\_id = r.role\_id

GROUP BY p.project\_name, r.role\_title;

-- Method 3: Using NOT IN

SELECT p.project\_name, r.role\_title, (

SELECT COUNT(DISTINCT emp\_id)

FROM allocation a

WHERE a.project\_id = p.project\_id AND a.role\_id = r.role\_id

) AS number\_of\_employees

FROM t\_project p, role r

WHERE (p.project\_id, r.role\_id) IN (

SELECT project\_id, role\_id

FROM allocation

)

ORDER BY p.project\_name, r.role\_title;

-- Method 4: Using Subquery

SELECT project\_name, role\_title, employee\_count FROM (

SELECT p.project\_name, r.role\_title,(

SELECT COUNT(DISTINCT a.emp\_id)

FROM allocation a

WHERE a.project\_id = p.project\_id AND a.role\_id = r.role\_id

) AS employee\_count

FROM t\_project p

CROSS JOIN role r

) sub

WHERE employee\_count > 0;

-- Method 5: Using EXISTS (Correlated Subquery)

SELECT p.project\_name, r.role\_title, COUNT(DISTINCT a.emp\_id) AS number\_of\_employees

FROM t\_project p

JOIN role r

JOIN allocation a

ON p.project\_id = a.project\_id AND r.role\_id = a.role\_id

WHERE EXISTS (

SELECT 1

FROM allocation a2

WHERE a2.project\_id = p.project\_id AND a2.role\_id = r.role\_id

)

GROUP BY p.project\_name, r.role\_title;

-- Method 6: Using UNION (SET operators)

SELECT project\_name, role\_title, number\_of\_employees FROM (

SELECT p.project\_name, r.role\_title, COUNT(DISTINCT a.emp\_id) AS number\_of\_employees

FROM allocation a

JOIN t\_project p

ON a.project\_id = p.project\_id

JOIN role r

ON a.role\_id = r.role\_id

GROUP BY p.project\_name, r.role\_title

UNION

SELECT NULL, NULL, 0

) unioned

WHERE project\_name IS NOT NULL;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT p.project\_name, r.role\_title, COUNT(DISTINCT a.emp\_id) AS number\_of\_employees

FROM allocation a

JOIN t\_project p

ON a.project\_id = p.project\_id

JOIN role r

ON a.role\_id = r.role\_id

EXCEPT

SELECT NULL, NULL, 0;

-- Method 8: Using INTERSECT (SET operators)

SELECT p.project\_name, r.role\_title, COUNT(DISTINCT a.emp\_id) AS number\_of\_employees

FROM allocation a

JOIN t\_project p

ON a.project\_id = p.project\_id

JOIN role r

ON a.role\_id = r.role\_id

INTERSECT

SELECT p.project\_name, r.role\_title, COUNT(DISTINCT a.emp\_id) AS number\_of\_employees

FROM allocation a

JOIN t\_project p

ON a.project\_id = p.project\_id

JOIN role r

ON a.role\_id = r.role\_id;

\*/

-- 29. Prepare a report in this format: Role Name | Emp Name | Number of Projects

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS get\_role\_emp\_project\_counts;

DELIMITER $$

CREATE PROCEDURE get\_role\_emp\_project\_counts()

BEGIN

SELECT r.role\_title, e.emp\_name, COUNT(DISTINCT a.project\_id) AS number\_of\_projects

FROM allocation a

JOIN role r

ON a.role\_id = r.role\_id

JOIN employee e

ON a.emp\_id = e.emp\_id

GROUP BY r.role\_title, e.emp\_name;

END $$

DELIMITER ;

CALL get\_role\_emp\_project\_counts();

-- Method 2: Using JOIN

SELECT r.role\_title, e.emp\_name, COUNT(DISTINCT a.project\_id) AS number\_of\_projects

FROM allocation a

JOIN role r

ON a.role\_id = r.role\_id

JOIN employee e

ON a.emp\_id = e.emp\_id

GROUP BY r.role\_title, e.emp\_name;

-- Method 3: Using NOT IN

SELECT r.role\_title, e.emp\_name, (

SELECT COUNT(DISTINCT project\_id)

FROM allocation a

WHERE a.role\_id = r.role\_id AND a.emp\_id = e.emp\_id

) AS number\_of\_projects

FROM role r, employee e

WHERE (r.role\_id, e.emp\_id) IN (

SELECT role\_id, emp\_id

FROM allocation

)

ORDER BY r.role\_title, e.emp\_name;

-- Method 4: Using Subquery

SELECT role\_title, emp\_name, project\_count FROM (

SELECT r.role\_title, e.emp\_name,(

SELECT COUNT(DISTINCT a.project\_id)

FROM allocation a

WHERE a.role\_id = r.role\_id AND a.emp\_id = e.emp\_id

) AS project\_count

FROM role r

CROSS JOIN employee e

) sub

WHERE project\_count > 0;

-- Method 5: Using EXISTS (Correlated Subquery)

SELECT r.role\_title, e.emp\_name, COUNT(DISTINCT a.project\_id) AS number\_of\_projects

FROM role r

JOIN employee e

JOIN allocation a

ON r.role\_id = a.role\_id AND e.emp\_id = a.emp\_id

WHERE EXISTS (

SELECT 1

FROM allocation a2

WHERE a2.role\_id = r.role\_id AND a2.emp\_id = e.emp\_id

)

GROUP BY r.role\_title, e.emp\_name;

-- Method 6: Using UNION (SET operators)

SELECT role\_title, emp\_name, number\_of\_projects FROM (

SELECT r.role\_title, e.emp\_name, COUNT(DISTINCT a.project\_id) AS number\_of\_projects

FROM allocation a

JOIN role r

ON a.role\_id = r.role\_id

JOIN employee e

ON a.emp\_id = e.emp\_id

GROUP BY r.role\_title, e.emp\_name

UNION

SELECT NULL, NULL, 0

) unioned

WHERE role\_title IS NOT NULL;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT r.role\_title, e.emp\_name, COUNT(DISTINCT a.project\_id) AS number\_of\_projects

FROM allocation a

JOIN role r

ON a.role\_id = r.role\_id

JOIN employee e

ON a.emp\_id = e.emp\_id

EXCEPT

SELECT NULL, NULL, 0;

-- Method 8: Using INTERSECT (SET operators)

SELECT r.role\_title, e.emp\_name, COUNT(DISTINCT a.project\_id) AS number\_of\_projects

FROM allocation a

JOIN role r

ON a.role\_id = r.role\_id

JOIN employee e

ON a.emp\_id = e.emp\_id

INTERSECT

SELECT r.role\_title, e.emp\_name, COUNT(DISTINCT a.project\_id) AS number\_of\_projects

FROM allocation a

JOIN role r

ON a.role\_id = r.role\_id

JOIN employee e

ON a.emp\_id = e.emp\_id;

\*/

-- 30. Prepare a report in this format: Dept Id | Number of Employees

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS get\_dept\_employee\_counts;

DELIMITER $$

CREATE PROCEDURE get\_dept\_employee\_counts()

BEGIN

SELECT d.dept\_id, COUNT(e.emp\_id) AS number\_of\_employees

FROM employee e

JOIN department d

ON e.dept\_id = d.dept\_id

GROUP BY d.dept\_id;

END $$

DELIMITER ;

CALL get\_dept\_employee\_counts();

-- Method 2: Using JOIN

SELECT d.dept\_id, COUNT(e.emp\_id) AS number\_of\_employees

FROM employee e

JOIN department d

ON e.dept\_id = d.dept\_id

GROUP BY d.dept\_id;

-- Method 3: Using NOT IN

SELECT d.dept\_id, (

SELECT COUNT(emp\_id)

FROM employee e

WHERE e.dept\_id = d.dept\_id

) AS number\_of\_employees

FROM department d

WHERE d.dept\_id IN (

SELECT DISTINCT dept\_id FROM employee

)

ORDER BY d.dept\_id;

-- Method 4: Using Subquery

SELECT dept\_id, employee\_count FROM (

SELECT d.dept\_id, (

SELECT COUNT(emp\_id)

FROM employee e

WHERE e.dept\_id = d.dept\_id

) AS employee\_count

FROM department d

) sub;

-- Method 5: Using EXISTS (Correlated Subquery)

SELECT d.dept\_id, COUNT(e.emp\_id) AS number\_of\_employees

FROM department d

JOIN employee e

ON d.dept\_id = e.dept\_id

WHERE EXISTS (

SELECT 1

FROM employee e2

WHERE e2.dept\_id = d.dept\_id

)

GROUP BY d.dept\_id;

-- Method 6: Using UNION (SET operators)

SELECT dept\_id, number\_of\_employees FROM (

SELECT d.dept\_id, COUNT(e.emp\_id) AS number\_of\_employees

FROM employee e

JOIN department d

ON e.dept\_id = d.dept\_id

GROUP BY d.dept\_id

UNION

SELECT NULL, 0

) unioned

WHERE dept\_id IS NOT NULL;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT d.dept\_id, COUNT(e.emp\_id) AS number\_of\_employees

FROM employee e

JOIN department d

ON e.dept\_id = d.dept\_id

EXCEPT

SELECT NULL, 0;

-- Method 8: Using INTERSECT (SET operators)

SELECT d.dept\_id, COUNT(e.emp\_id) AS number\_of\_employees

FROM employee e

JOIN department d

ON e.dept\_id = d.dept\_id

INTERSECT

SELECT d.dept\_id, COUNT(e.emp\_id) AS number\_of\_employees

FROM employee e

JOIN department d

ON e.dept\_id = d.dept\_id;

\*/

-- 31. Prepare a report in this format: Mgr\_id | Number of Employees

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS mgr\_employee\_count;

DELIMITER $$

CREATE PROCEDURE mgr\_employee\_count()

BEGIN

SELECT mgr\_id, COUNT(emp\_id) AS number\_of\_employees

FROM employee

WHERE mgr\_id IS NOT NULL

GROUP BY mgr\_id;

END $$

DELIMITER ;

CALL mgr\_employee\_count();

-- Method 2: Using WHERE

SELECT e.mgr\_id, COUNT(e.emp\_id) AS number\_of\_employees

FROM employee e

WHERE e.mgr\_id IS NOT NULL

GROUP BY e.mgr\_id;

-- Method 3: Using NOT IN

SELECT mgr\_id, COUNT(emp\_id) AS number\_of\_employees

FROM employee

WHERE mgr\_id NOT IN (

SELECT emp\_id

FROM employee

WHERE emp\_id IS NULL

)

AND mgr\_id IS NOT NULL

GROUP BY mgr\_id;

-- Method 4: Using Subquery

SELECT DISTINCT mgr\_id, (

SELECT COUNT(\*)

FROM employee e2

WHERE e2.mgr\_id = e1.mgr\_id

) AS number\_of\_employees

FROM employee e1

WHERE mgr\_id IS NOT NULL;

-- Method 5: Using EXISTS (Correlated Subquery)

SELECT DISTINCT e1.mgr\_id, (

SELECT COUNT(\*)

FROM employee e2

WHERE e2.mgr\_id = e1.mgr\_id

) AS number\_of\_employees

FROM employee e1

WHERE EXISTS (

SELECT 1

FROM employee e2

WHERE e2.mgr\_id = e1.mgr\_id

);

-- Method 6: Using UNION (SET operators)

SELECT mgr\_id, COUNT(emp\_id) AS number\_of\_employees

FROM employee

WHERE mgr\_id IS NOT NULL AND mgr\_id < 1000

GROUP BY mgr\_id

UNION

SELECT mgr\_id, COUNT(emp\_id)

FROM employee

WHERE mgr\_id IS NOT NULL AND mgr\_id >= 1000

GROUP BY mgr\_id;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT mgr\_id

FROM employee

WHERE mgr\_id IS NOT NULL

EXCEPT

SELECT emp\_id

FROM employee

WHERE emp\_id IS NULL;

-- Method 8: Using INTERSECT (SET operators)

SELECT mgr\_id

FROM employee

WHERE mgr\_id IS NOT NULL

INTERSECT

SELECT emp\_id

FROM employee;

\*/

-- 32. Prepare a report in this format: Emp Name | Role Name | Project Name

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS get\_emp\_role\_project;

DELIMITER $$

CREATE PROCEDURE get\_emp\_role\_project()

BEGIN

SELECT e.emp\_name, r.role\_title, p.project\_name

FROM allocation a

JOIN employee e

ON a.emp\_id = e.emp\_id

JOIN role r

ON a.role\_id = r.role\_id

JOIN t\_project p

ON a.project\_id = p.project\_id

ORDER BY e.emp\_name, r.role\_title, p.project\_name;

END $$

DELIMITER ;

CALL get\_emp\_role\_project();

-- Method 2: Using JOIN

SELECT e.emp\_name, r.role\_title, p.project\_name

FROM allocation a

JOIN employee e

ON a.emp\_id = e.emp\_id

JOIN role r

ON a.role\_id = r.role\_id

JOIN t\_project p

ON a.project\_id = p.project\_id

ORDER BY e.emp\_name, r.role\_title, p.project\_name;

-- Method 3: Using NOT IN

SELECT e.emp\_name, r.role\_title, p.project\_name

FROM employee e

JOIN allocation a

ON e.emp\_id = a.emp\_id

JOIN role r

ON a.role\_id = r.role\_id

JOIN t\_project p

ON a.project\_id = p.project\_id

WHERE a.emp\_id NOT IN (

SELECT emp\_id

FROM employee

WHERE emp\_id IS NULL

)

ORDER BY e.emp\_name, r.role\_title, p.project\_name;

-- Method 4: Using Subquery

SELECT emp\_name, role\_title, project\_name

FROM (

SELECT e.emp\_name, r.role\_title, p.project\_name

FROM allocation a

JOIN employee e

ON a.emp\_id = e.emp\_id

JOIN role r

ON a.role\_id = r.role\_id

JOIN t\_project p

ON a.project\_id = p.project\_id

) sub

ORDER BY emp\_name, role\_title, project\_name;

-- Method 5: Using EXISTS (Correlated Subquery)

SELECT e.emp\_name, r.role\_title, p.project\_name

FROM allocation a

JOIN employee e

ON a.emp\_id = e.emp\_id

JOIN role r

ON a.role\_id = r.role\_id

JOIN t\_project p

ON a.project\_id = p.project\_id

WHERE EXISTS (

SELECT 1

FROM allocation a2

WHERE a2.emp\_id = a.emp\_id

AND a2.role\_id = a.role\_id

AND a2.project\_id = a.project\_id

)

ORDER BY e.emp\_name, r.role\_title, p.project\_name;

-- Method 6: Using UNION (SET operators)

SELECT e.emp\_name, r.role\_title, p.project\_name

FROM allocation a

JOIN employee e

ON a.emp\_id = e.emp\_id

JOIN role r

ON a.role\_id = r.role\_id

JOIN t\_project p

ON a.project\_id = p.project\_id

WHERE a.project\_id < 1000

UNION

SELECT e.emp\_name, r.role\_title, p.project\_name

FROM allocation a

JOIN employee e

ON a.emp\_id = e.emp\_id

JOIN role r

ON a.role\_id = r.role\_id

JOIN t\_project p

ON a.project\_id = p.project\_id

WHERE a.project\_id >= 1000

ORDER BY 1,2,3;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT e.emp\_name, r.role\_title, p.project\_name

FROM allocation a

JOIN employee e

ON a.emp\_id = e.emp\_id

JOIN role r

ON a.role\_id = r.role\_id

JOIN t\_project p

ON a.project\_id = p.project\_id

EXCEPT

SELECT NULL, NULL, NULL;

-- Method 8: Using INTERSECT (SET operators)

SELECT e.emp\_name, r.role\_title, p.project\_name

FROM allocation a

JOIN employee e

ON a.emp\_id = e.emp\_id

JOIN role r

ON a.role\_id = r.role\_id

JOIN t\_project p

ON a.project\_id = p.project\_id

INTERSECT

SELECT e.emp\_name, r.role\_title, p.project\_name

FROM allocation a

JOIN employee e

ON a.emp\_id = e.emp\_id

JOIN role r

ON a.role\_id = r.role\_id

JOIN t\_project p

ON a.project\_id = p.project\_id;

\*/

-- 33. Prepare a report in this format: Project id | Emp id | Total Amount collected Sort the report with respect to the Total Amount collected in Descending Order.

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS get\_project\_emp\_amount;

DELIMITER $$

CREATE PROCEDURE get\_project\_emp\_amount()

BEGIN

SELECT a.project\_id,

a.emp\_id,

SUM(a.amount\_per\_day \* (DATEDIFF(a.to\_date, a.from\_date) + 1)) AS total\_amount\_collected

FROM allocation a

GROUP BY a.project\_id, a.emp\_id

ORDER BY total\_amount\_collected DESC;

END $$

DELIMITER ;

CALL get\_project\_emp\_amount();

-- Method 2: Using ORDER BY

SELECT a.project\_id, a.emp\_id, SUM(a.amount\_per\_day \* (DATEDIFF(a.to\_date, a.from\_date) + 1)) AS total\_amount\_collected

FROM allocation a

GROUP BY a.project\_id, a.emp\_id

ORDER BY total\_amount\_collected DESC;

-- Method 3: Using NOT IN

SELECT project\_id, emp\_id, total\_amount\_collected

FROM (

SELECT a.project\_id, a.emp\_id, SUM(a.amount\_per\_day \* (DATEDIFF(a.to\_date, a.from\_date) + 1)) AS total\_amount\_collected

FROM allocation a

GROUP BY a.project\_id, a.emp\_id

) t

WHERE emp\_id NOT IN (

SELECT NULL

)

ORDER BY total\_amount\_collected DESC;

-- Method 4: Using JOIN

SELECT a.project\_id, a.emp\_id, SUM(a.amount\_per\_day \* (DATEDIFF(a.to\_date, a.from\_date) + 1)) AS total\_amount\_collected

FROM allocation a

JOIN allocation b

ON a.project\_id = b.project\_id

AND a.emp\_id = b.emp\_id

GROUP BY a.project\_id, a.emp\_id

ORDER BY total\_amount\_collected DESC;

-- Method 5: Using EXISTS (Correlated Subquery)

SELECT a.project\_id, a.emp\_id, (

SELECT SUM(amount\_per\_day \* (DATEDIFF(to\_date, from\_date) + 1))

FROM allocation x

WHERE x.project\_id = a.project\_id AND x.emp\_id = a.emp\_id

) AS total\_amount\_collected

FROM allocation a

WHERE EXISTS (

SELECT 1

FROM allocation y

WHERE y.project\_id = a.project\_id

AND y.emp\_id = a.emp\_id

)

GROUP BY a.project\_id, a.emp\_id, total\_amount\_collected

ORDER BY total\_amount\_collected DESC;

-- Method 6: Using UNION (SET operators)

SELECT project\_id, emp\_id, total\_amount\_collected

FROM (

SELECT a.project\_id, a.emp\_id, SUM(a.amount\_per\_day \* (DATEDIFF(a.to\_date, a.from\_date) + 1)) AS total\_amount\_collected

FROM allocation a

GROUP BY a.project\_id, a.emp\_id

UNION

SELECT NULL, NULL, 0

) u

WHERE project\_id IS NOT NULL

ORDER BY total\_amount\_collected DESC;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT project\_id, emp\_id, SUM(amount\_per\_day \* (DATEDIFF(to\_date, from\_date) + 1)) AS total\_amount

FROM allocation

GROUP BY project\_id, emp\_id

EXCEPT

SELECT NULL, NULL, 0;

-- Method 8: Using INTERSECT (SET operators)

SELECT project\_id, emp\_id, SUM(amount\_per\_day \* (DATEDIFF(to\_date, from\_date) + 1)) AS total\_amount

FROM allocation

GROUP BY project\_id, emp\_id

INTERSECT

SELECT project\_id, emp\_id, SUM(amount\_per\_day \* (DATEDIFF(to\_date, from\_date) + 1)) AS total\_amount

FROM allocation

GROUP BY project\_id, emp\_id;

\*/

-- 34. Prepare a report in this format: Emp id | Role id | Total Amount collected

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS get\_emp\_role\_amount;

DELIMITER $$

CREATE PROCEDURE get\_emp\_role\_amount()

BEGIN

SELECT a.emp\_id, a.role\_id, SUM(a.amount\_per\_day \* (DATEDIFF(a.to\_date, a.from\_date) + 1)) AS total\_amount\_collected

FROM allocation a

GROUP BY a.emp\_id, a.role\_id

ORDER BY total\_amount\_collected;

END $$

DELIMITER ;

CALL get\_emp\_role\_amount();

-- Method 2: Using ORDER BY

SELECT a.emp\_id, a.role\_id, SUM(a.amount\_per\_day \* (DATEDIFF(a.to\_date, a.from\_date) + 1)) AS total\_amount\_collected

FROM allocation a

GROUP BY a.emp\_id, a.role\_id

ORDER BY total\_amount\_collected;

-- Method 3: Using NOT IN

SELECT emp\_id, role\_id, total\_amount\_collected

FROM (

SELECT emp\_id, role\_id, SUM(amount\_per\_day \* (DATEDIFF(to\_date, from\_date) + 1)) AS total\_amount\_collected

FROM allocation

GROUP BY emp\_id, role\_id

) t

WHERE role\_id NOT IN (

SELECT NULL

)

ORDER BY total\_amount\_collected;

-- Method 4: Using JOIN

SELECT a.emp\_id, a.role\_id, SUM(a.amount\_per\_day \* (DATEDIFF(a.to\_date, a.from\_date) + 1)) AS total\_amount\_collected

FROM allocation a

JOIN allocation b

ON a.emp\_id = b.emp\_id AND a.role\_id = b.role\_id

GROUP BY a.emp\_id, a.role\_id

ORDER BY total\_amount\_collected;

-- Method 5: Using EXISTS (Correlated Subquery)

SELECT a.emp\_id, a.role\_id, (

SELECT SUM(amount\_per\_day \* (DATEDIFF(to\_date, from\_date) + 1))

FROM allocation x

WHERE x.emp\_id = a.emp\_id

AND x.role\_id = a.role\_id) AS total\_amount\_collected

FROM allocation a

WHERE EXISTS (

SELECT 1

FROM allocation y

WHERE y.emp\_id = a.emp\_id

AND y.role\_id = a.role\_id

)

GROUP BY a.emp\_id, a.role\_id, total\_amount\_collected

ORDER BY total\_amount\_collected;

-- Method 6: Using UNION (SET operators)

SELECT emp\_id, role\_id, total\_amount\_collected

FROM (

SELECT emp\_id, role\_id, SUM(amount\_per\_day \* (DATEDIFF(to\_date, from\_date) + 1)) AS total\_amount\_collected

FROM allocation

GROUP BY emp\_id, role\_id

UNION

SELECT NULL, NULL, 0

) u

WHERE emp\_id IS NOT NULL

ORDER BY total\_amount\_collected;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT emp\_id, role\_id, SUM(amount\_per\_day \* (DATEDIFF(to\_date, from\_date) + 1)) AS total\_amount

FROM allocation

GROUP BY emp\_id, role\_id

EXCEPT

SELECT NULL, NULL, 0;

-- Method 8: Using INTERSECT (SET operators)

SELECT emp\_id, role\_id, SUM(amount\_per\_day \* (DATEDIFF(to\_date, from\_date) + 1)) AS total\_amount

FROM allocation

GROUP BY emp\_id, role\_id

INTERSECT

SELECT emp\_id, role\_id, SUM(amount\_per\_day \* (DATEDIFF(to\_date, from\_date) + 1)) AS total\_amount

FROM allocation

GROUP BY emp\_id, role\_id;

\*/

-- 35. Prepare a report in this format: Emp id | Role id | Project id | Total Amount collected Sort the report with respect to the Total Amount collected in Descending Order.

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS get\_emp\_role\_amount;

DELIMITER $$

CREATE PROCEDURE get\_emp\_role\_amount()

BEGIN

SELECT a.emp\_id, a.role\_id, SUM(a.amount\_per\_day \* (DATEDIFF(a.to\_date, a.from\_date) + 1)) AS total\_amount\_collected

FROM allocation a

GROUP BY a.emp\_id, a.role\_id

ORDER BY total\_amount\_collected;

END $$

DELIMITER ;

CALL get\_emp\_role\_amount();

-- Method 2: Using ORDER BY

SELECT a.emp\_id, a.role\_id, a.project\_id, SUM(a.amount\_per\_day \* (DATEDIFF(a.to\_date, a.from\_date) + 1)) AS total\_amount\_collected

FROM allocation a

GROUP BY a.emp\_id, a.role\_id, a.project\_id

ORDER BY total\_amount\_collected DESC;

-- Method 3: Using NOT IN

SELECT emp\_id, role\_id, total\_amount\_collected

FROM (

SELECT emp\_id, role\_id, SUM(amount\_per\_day \* (DATEDIFF(to\_date, from\_date) + 1)) AS total\_amount\_collected

FROM allocation

GROUP BY emp\_id, role\_id

) t

WHERE role\_id NOT IN (

SELECT NULL

)

ORDER BY total\_amount\_collected;

-- Method 4: Using JOIN

SELECT a.emp\_id, a.role\_id, SUM(a.amount\_per\_day \* (DATEDIFF(a.to\_date, a.from\_date) + 1)) AS total\_amount\_collected

FROM allocation a

JOIN allocation b

ON a.emp\_id = b.emp\_id AND a.role\_id = b.role\_id

GROUP BY a.emp\_id, a.role\_id

ORDER BY total\_amount\_collected;

-- Method 5: Using EXISTS (Correlated Subquery)

SELECT a.emp\_id, a.role\_id, (SELECT SUM(amount\_per\_day \* (DATEDIFF(to\_date, from\_date) + 1))

FROM allocation x

WHERE x.emp\_id = a.emp\_id

AND x.role\_id = a.role\_id) AS total\_amount\_collected

FROM allocation a

WHERE EXISTS (

SELECT 1

FROM allocation y

WHERE y.emp\_id = a.emp\_id

AND y.role\_id = a.role\_id

)

GROUP BY a.emp\_id, a.role\_id, total\_amount\_collected

ORDER BY total\_amount\_collected;

-- Method 6: Using UNION (SET operators)

SELECT emp\_id, role\_id, total\_amount\_collected

FROM (

SELECT emp\_id, role\_id, SUM(amount\_per\_day \* (DATEDIFF(to\_date, from\_date) + 1)) AS total\_amount\_collected

FROM allocation

GROUP BY emp\_id, role\_id

UNION

SELECT NULL, NULL, 0

) u

WHERE emp\_id IS NOT NULL

ORDER BY total\_amount\_collected;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT emp\_id, role\_id, SUM(amount\_per\_day \* (DATEDIFF(to\_date, from\_date) + 1)) AS total\_amount

FROM allocation

GROUP BY emp\_id, role\_id

EXCEPT

SELECT NULL, NULL, 0;

-- Method 8: Using INTERSECT (SET operators)

SELECT emp\_id, role\_id, SUM(amount\_per\_day \* (DATEDIFF(to\_date, from\_date) + 1)) AS total\_amount

FROM allocation

GROUP BY emp\_id, role\_id

INTERSECT

SELECT emp\_id, role\_id, SUM(amount\_per\_day \* (DATEDIFF(to\_date, from\_date) + 1)) AS total\_amount

FROM allocation

GROUP BY emp\_id, role\_id;

\*/

-- 36. Prepare a report in this format

-- Emp id | Mgr Id | Comments

-- 123 | 432 | Has Manager

-- 456 | - | No Manager

-- 125 | 456 | Has Manager

-- If Manager id is NULL for an employee then the comment must be “No Manager “

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS get\_manager\_report;

DELIMITER $$

CREATE PROCEDURE get\_manager\_report()

BEGIN

SELECT emp\_id, COALESCE(mgr\_id, '-') AS mgr\_id,

CASE

WHEN mgr\_id IS NULL THEN 'No Manager'

ELSE 'Has Manager'

END AS comments

FROM employee;

END $$

DELIMITER ;

CALL get\_manager\_report();

-- Method 2: Using CASE

SELECT emp\_id, COALESCE(mgr\_id, '-') AS mgr\_id,

CASE

WHEN mgr\_id IS NULL THEN 'No Manager'

ELSE 'Has Manager'

END AS comments

FROM employee;

-- 37. Prepare a report in this format

-- Ram Works for Ashok

-- Adam works for Akbar

-- Where Ram is employee name and Ashok is his corresponding manager

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS get\_emp\_manager\_report;

DELIMITER $$

CREATE PROCEDURE get\_emp\_manager\_report()

BEGIN

SELECT CONCAT(e1.emp\_name, ' works for ', COALESCE(e2.emp\_name, 'No Manager')) AS report\_line

FROM employee e1

LEFT JOIN employee e2 ON e1.mgr\_id = e2.emp\_id

ORDER BY e1.emp\_name;

END $$

DELIMITER ;

CALL get\_emp\_manager\_report();

-- Method 2: Using LEFT JOIN

SELECT e1.emp\_name, COALESCE(e2.emp\_name, 'No Manager') AS manager

FROM employee e1

LEFT JOIN employee e2

ON e1.mgr\_id = e2.emp\_id;

-- Method 3: Using NOT IN

SELECT CONCAT(emp\_name, ' works for ', COALESCE(mgr\_name, 'No Manager')) AS report\_line

FROM (

SELECT e1.emp\_name,(

SELECT e2.emp\_name

FROM employee e2

WHERE e2.emp\_id = e1.mgr\_id

) AS mgr\_name

FROM employee e1

WHERE e1.emp\_id NOT IN (SELECT NULL)

) t

ORDER BY emp\_name;

-- Method 4: Using Subquery

SELECT CONCAT(emp\_name, ' works for ', COALESCE(mgr\_name, 'No Manager')) AS report\_line

FROM (

SELECT e1.emp\_name,(

SELECT e2.emp\_name

FROM employee e2

WHERE e2.emp\_id = e1.mgr\_id

) AS mgr\_name

FROM employee e1

) sub

ORDER BY emp\_name;

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

SELECT CONCAT(e1.emp\_name, ' works for ', COALESCE(e2.emp\_name, 'No Manager')) AS report\_line

FROM employee e1

LEFT JOIN employee e2

ON e1.mgr\_id = e2.emp\_id

WHERE EXISTS (SELECT 1)

ORDER BY e1.emp\_name;

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

SELECT CONCAT(emp\_name, ' works for ', COALESCE(mgr\_name, 'No Manager')) AS report\_line

FROM (

SELECT e1.emp\_name, e2.emp\_name AS mgr\_name

FROM employee e1

JOIN employee e2

ON e1.mgr\_id = e2.emp\_id

UNION ALL

SELECT e1.emp\_name, NULL AS mgr\_name

FROM employee e1

WHERE e1.mgr\_id IS NULL

) u

ORDER BY emp\_name;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT e1.emp\_name, COALESCE(e2.emp\_name, 'No Manager') AS mgr\_name

FROM employee e1

LEFT JOIN employee e2 ON e1.mgr\_id = e2.emp\_id

EXCEPT

SELECT NULL, NULL;

-- Method 8: Using INTERSECT (SET operators)

SELECT e1.emp\_name, COALESCE(e2.emp\_name, 'No Manager') AS mgr\_name

FROM employee e1

LEFT JOIN employee e2 ON e1.mgr\_id = e2.emp\_id

INTERSECT

SELECT e1.emp\_name, COALESCE(e2.emp\_name, 'No Manager') AS mgr\_name

FROM employee e1

LEFT JOIN employee e2 ON e1.mgr\_id = e2.emp\_id;

\*/

-- 38. Which employees earns more than his/her manager?

-- Method 1: Using JOIN

SELECT e1.emp\_id, e1.emp\_name AS manager\_name, e1.hire\_date AS manager\_hiredate, e2.emp\_id AS subordinate\_id, e2.emp\_name AS subordinate\_name, e2.hire\_date AS subordinate\_hiredate

FROM employee e1

JOIN employee e2

ON e1.emp\_id = e2.mgr\_id

WHERE e1.hire\_date < e2.hire\_date;

-- Method 2: Using NOT IN

SELECT emp\_id, emp\_name, salary, mgr\_id, (

SELECT salary

FROM employee m

WHERE m.emp\_id = e.mgr\_id

) AS mgr\_salary

FROM employee e

WHERE mgr\_id IS NOT NULL

AND salary > (

SELECT salary

FROM employee m

WHERE m.emp\_id = e.mgr\_id

)

AND mgr\_id NOT IN (

SELECT emp\_id

FROM employee

WHERE salary >= ALL (

SELECT salary

FROM employee

)

);

-- Method 3: Using Subquery

SELECT CONCAT(emp\_id, ' has a manager earning ', COALESCE(mgr\_salary, 'No Manager Salary')) AS report\_line

FROM (

SELECT emp\_id, emp\_name, salary, mgr\_id, (SELECT salary FROM employee m WHERE m.emp\_id = e.mgr\_id) AS mgr\_salary

FROM employee e

WHERE salary > (

SELECT salary

FROM employee m

WHERE m.emp\_id = e.mgr\_id

)

) sub

ORDER BY emp\_id;

-- Method 4: Using EXISTS (Correlated Subquery)

SELECT CONCAT(e.emp\_id, ' has a manager earning ', COALESCE(mgr\_salary, 'No Manager Salary')) AS report\_line

FROM employee e

LEFT JOIN (

SELECT emp\_id, salary AS mgr\_salary

FROM employee

) m

ON e.mgr\_id = m.emp\_id

WHERE EXISTS (

SELECT 1

FROM employee m

WHERE m.emp\_id = e.mgr\_id

AND e.salary > m.salary

)

ORDER BY e.emp\_id;

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

SELECT CONCAT(emp\_id, ' has a manager earning ', COALESCE(mgr\_salary, 'No Manager Salary')) AS report\_line

FROM (

SELECT emp\_id, emp\_name, salary, mgr\_id, (SELECT salary FROM employee m WHERE m.emp\_id = e.mgr\_id) AS mgr\_salary

FROM employee e

WHERE salary > (

SELECT salary

FROM employee m

WHERE m.emp\_id = e.mgr\_id

)

UNION ALL

SELECT emp\_id, emp\_name, salary, mgr\_id, NULL AS mgr\_salary

FROM employee

WHERE mgr\_id IS NULL

) u

ORDER BY emp\_id;

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT emp\_id, emp\_name

FROM employee

EXCEPT

SELECT emp\_id, emp\_name

FROM employee

WHERE salary <= (

SELECT salary

FROM employee m

WHERE m.emp\_id = employee.mgr\_id

);

-- Method 7: Using INTERSECT (SET operators)

SELECT emp\_id, emp\_name

FROM employee

INTERSECT

SELECT emp\_id, emp\_name

FROM employee

WHERE salary > (

SELECT salary

FROM employee m

WHERE m.emp\_id = employee.mgr\_id

);

\*/

-- 39. Which manager joined after his/her sub ordinates?

-- Method 1: Using WITH

WITH dept\_avg\_salary AS (

SELECT dept\_id, AVG(salary) AS avg\_salary

FROM employee

GROUP BY dept\_id

)

SELECT e.emp\_id, e.emp\_name, e.salary, e.dept\_id

FROM employee e

JOIN dept\_avg\_salary das

ON e.dept\_id = das.dept\_id

WHERE e.salary > das.avg\_salary;

-- Method 2: Using NOT IN

SELECT DISTINCT m.emp\_id, m.emp\_name, m.hire\_date, NULL AS mgr\_id

FROM employee m

WHERE m.emp\_id IN (

SELECT mgr\_id

FROM employee

WHERE mgr\_id IS NOT NULL

)

AND m.hire\_date > ALL (

SELECT s.hire\_date

FROM employee s

WHERE s.mgr\_id = m.emp\_id

)

AND m.emp\_id NOT IN (

SELECT emp\_id

FROM employee

WHERE mgr\_id IS NULL

);

-- Method 3: Using Subquery

SELECT emp\_id, emp\_name, hire\_date, NULL AS mgr\_id

FROM employee

WHERE emp\_id IN (

SELECT DISTINCT e1.emp\_id

FROM employee e1

JOIN employee e2

ON e1.emp\_id = e2.mgr\_id

WHERE e1.hire\_date > e2.hire\_date

);

-- Method 4: Using EXISTS (Correlated Subquery)

SELECT DISTINCT e1.emp\_id, e1.emp\_name, e1.hire\_date, NULL AS mgr\_id

FROM employee e1

WHERE EXISTS (

SELECT 1

FROM employee e2

WHERE e2.mgr\_id = e1.emp\_id

AND e1.hire\_date > e2.hire\_date

);

-- Method 5: Using UNION (SET operators)

SELECT emp\_id, emp\_name, hire\_date, NULL AS mgr\_id

FROM (

SELECT e1.emp\_id, e1.emp\_name, e1.hire\_date

FROM employee e1

JOIN employee e2

ON e1.emp\_id = e2.mgr\_id

WHERE e1.hire\_date > e2.hire\_date

UNION

SELECT emp\_id, emp\_name, hire\_date

FROM employee

WHERE mgr\_id IS NULL

) AS union\_result;

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT e1.emp\_id, e1.emp\_name, e1.hire\_date

FROM employee e1

JOIN employee e2

ON e1.emp\_id = e2.mgr\_id

WHERE e1.hire\_date > e2.hire\_date

EXCEPT

SELECT emp\_id, emp\_name, hire\_date

FROM employee

WHERE mgr\_id IS NULL;

-- Method 7: Using INTERSECT (SET operators)

SELECT e1.emp\_id, e1.emp\_name, e1.hire\_date

FROM employee e1

INTERSECT

SELECT e2.mgr\_id, e2.emp\_name, e2.hire\_date

FROM employee e2

WHERE e2.mgr\_id IS NOT NULL;

\*/

-- 40. Using Correlated Subquery and WITH clause: Find all employees who earns more than the average salary of his/her department?

-- Method 1: Using Subquery

SELECT EMP\_ID, EMP\_NAME, SALARY, DEPT\_ID

FROM EMPLOYEE E

WHERE SALARY > (

SELECT AVG(SALARY)

FROM EMPLOYEE

WHERE DEPT\_ID = E.DEPT\_ID

);

-- Method 2: Using EXISTS (Correlated Subquery)

SELECT e1.emp\_id, e1.emp\_name, e1.salary, e1.dept\_id

FROM employee e1

WHERE EXISTS (

SELECT 1

FROM employee e2

WHERE e1.dept\_id = e2.dept\_id

GROUP BY e2.dept\_id

HAVING e1.salary > AVG(e2.salary)

);

-- Method 3: Using WITH

WITH dept\_avg\_salary AS (

SELECT dept\_id, AVG(salary) AS avg\_salary

FROM employee

GROUP BY dept\_id

)

SELECT e.emp\_id, e.emp\_name, e.salary, e.dept\_id

FROM employee e

JOIN dept\_avg\_salary d

ON e.dept\_id = d.dept\_id

WHERE e.salary > d.avg\_salary;

-- 41. Using Correlated Subquery: Display the details of employees who have changed their roles at least twice?

-- Method 1: Using Subquery

SELECT emp\_id, emp\_name, dept\_id, salary

FROM employee e

WHERE (

SELECT COUNT(DISTINCT role\_id)

FROM allocation a

WHERE a.emp\_id = e.emp\_id

) >= 2;

-- Method 2: Using EXISTS (Correlated Subquery)

SELECT e.emp\_id, e.emp\_name, e.dept\_id, e.salary

FROM employee e

WHERE EXISTS (

SELECT 1

FROM allocation a

WHERE a.emp\_id = e.emp\_id

GROUP BY a.emp\_id

HAVING COUNT(DISTINCT a.role\_id) >= 2

);

-- 42. Display the departments that does not have employees(ALL POSSIBILITIES)?

-- Method 1: Using LEFT JOIN

SELECT d.dept\_id, d.dept\_name

FROM department d

LEFT JOIN employee e

ON d.dept\_id = e.dept\_id

WHERE e.dept\_id IS NULL;

-- Method 2: Using NOT IN

SELECT dept\_id, dept\_name

FROM department

WHERE dept\_id NOT IN (

SELECT dept\_id

FROM employee

WHERE dept\_id IS NOT NULL

);

-- Method 3: Using Subquery

SELECT dept\_id, dept\_name

FROM department

WHERE dept\_id IN (

SELECT d.dept\_id

FROM department d

LEFT JOIN employee e

ON d.dept\_id = e.dept\_id

WHERE e.dept\_id IS NULL

);

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

SELECT d.dept\_id, d.dept\_name

FROM department d

WHERE NOT EXISTS (

SELECT 1

FROM employee e

WHERE e.dept\_id = d.dept\_id

);

-- Method 5: Using UNION (SET operators)

SELECT dept\_id, dept\_name

FROM department

WHERE dept\_id NOT IN (

SELECT dept\_id

FROM employee

WHERE dept\_id IS NOT NULL

)

UNION

SELECT d.dept\_id, d.dept\_name

FROM department d

LEFT JOIN employee e

ON d.dept\_id = e.dept\_id

WHERE e.dept\_id IS NULL;

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT dept\_id, dept\_name

FROM department

EXCEPT

SELECT d.dept\_id, d.dept\_name

FROM department d

JOIN employee e

ON d.dept\_id = e.dept\_id;

-- Method 7: Using INTERSECT (SET operators)

SELECT d.dept\_id, d.dept\_name

FROM department d

LEFT JOIN employee e

ON d.dept\_id = e.dept\_id

WHERE e.dept\_id IS NULL

INTERSECT

SELECT dept\_id, dept\_name

FROM department;

\*/

-- 43. Find the department which has employees(ALL POSSIBILITIES)?

-- Method 1: Using JOIN

SELECT DISTINCT d.dept\_id, d.dept\_name

FROM department d

JOIN employee e

ON d.dept\_id = e.dept\_id;

-- Method 2: Using NOT IN

SELECT dept\_id, dept\_name

FROM department

WHERE dept\_id IN (

SELECT dept\_id

FROM employee

WHERE dept\_id IS NOT NULL

);

-- Method 3: Using Subquery

SELECT dept\_id, dept\_name

FROM department

WHERE dept\_id IN (

SELECT DISTINCT e.dept\_id

FROM employee e

);

-- Method 4: Using EXISTS (Correlated Subquery)

SELECT d.dept\_id, d.dept\_name

FROM department d

WHERE EXISTS (

SELECT 1

FROM employee e

WHERE e.dept\_id = d.dept\_id

);

-- Method 5: Using UNION (SET operators)

SELECT dept\_id, dept\_name

FROM department

WHERE dept\_id IN (

SELECT dept\_id

FROM employee

WHERE dept\_id IS NOT NULL

)

UNION

SELECT d.dept\_id, d.dept\_name

FROM department d

JOIN employee e

ON d.dept\_id = e.dept\_id;

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT dept\_id, dept\_name

FROM department

EXCEPT

SELECT d.dept\_id, d.dept\_name

FROM department d

LEFT JOIN employee e

ON d.dept\_id = e.dept\_id

WHERE e.dept\_id IS NULL;

-- Method 7: Using INTERSECT (SET operators)

SELECT dept\_id, dept\_name

FROM department

INTERSECT

SELECT d.dept\_id, d.dept\_name

FROM department d

JOIN employee e

ON d.dept\_id = e.dept\_id;

\*/

-- 44. Using ROLL UP and CUBE Generate a report: Project ID | Role ID | No of Employees

-- Method 1: Using ROLLUP

SELECT project\_id, role\_id, COUNT(emp\_id) AS no\_of\_employees

FROM allocation

GROUP BY project\_id, role\_id WITH ROLLUP;

-- Method 2: Using CUBE

SELECT project\_id, role\_id, COUNT(emp\_id) AS no\_of\_employees

FROM allocation

GROUP BY project\_id, role\_id

UNION ALL

SELECT project\_id, NULL AS role\_id, COUNT(emp\_id) AS no\_of\_employees

FROM allocation

GROUP BY project\_id, role\_id

UNION ALL

SELECT NULL AS project\_id, role\_id, COUNT(emp\_id) AS no\_of\_employees

FROM allocation

GROUP BY project\_id, role\_id

UNION ALL

SELECT NULL AS project\_id, NULL AS role\_id, COUNT(emp\_id) AS no\_of\_employees

FROM allocation;

/\* Funtion using postgres

-- Method 1: Using ROLLUP

SELECT project\_id, role\_id, COUNT(emp\_id) AS no\_of\_employees

FROM allocation

GROUP BY ROLLUP(project\_id, role\_id);

-- Method 2: Using CUBE

SELECT project\_id, role\_id, COUNT(emp\_id) AS no\_of\_employees

FROM allocation

GROUP BY CUBE(project\_id, role\_id);

\*/

-- 45. Using ROLL UP and CUBE Generate a report: Employee ID | Project ID | Total salary

-- Method 1: Using ROLLUP

SELECT emp\_id, project\_id, SUM(amount\_per\_day) AS total\_salary

FROM allocation

GROUP BY emp\_id, project\_id WITH ROLLUP;

-- Method 2: Using CUBE

SELECT emp\_id, project\_id, SUM(amount\_per\_day) AS total\_salary

FROM allocation

GROUP BY emp\_id, project\_id

UNION ALL

SELECT emp\_id, NULL AS project\_id, SUM(amount\_per\_day) AS total\_salary

FROM allocation

GROUP BY emp\_id

UNION ALL

SELECT NULL AS emp\_id, project\_id, SUM(amount\_per\_day) AS total\_salary

FROM allocation

GROUP BY project\_id

UNION ALL

SELECT NULL AS emp\_id, NULL AS project\_id, SUM(amount\_per\_day) AS total\_salary

FROM allocation;

/\* Functional in postgres

-- Method 1: Using ROLLUP

SELECT emp\_id, project\_id, SUM(amount\_per\_day) AS total\_salary

FROM allocation

GROUP BY ROLLUP(emp\_id, project\_id);

-- Method 2: Using CUBE

SELECT emp\_id, project\_id, SUM(amount\_per\_day) AS total\_salary

FROM allocation;

GROUP BY CUBE (emp\_id, project\_id);

\*/

-- 46. Using CASE/DECODE Emp ID | Number of Skills | Description

-- If Employee have, more than 5 no of skills: Description is “Major Resource”,

-- more than 3 no of skills : Description is “Useful Resource”, more than 1 no of skills : Description is “Resource”

-- Method 1: Using CASE

SELECT emp\_id, COUNT(skill\_id) AS number\_of\_skills,

CASE

WHEN COUNT(skill\_id) > 5 THEN 'Major Resource'

WHEN COUNT(skill\_id) > 3 THEN 'Useful Resource'

WHEN COUNT(skill\_id) > 1 THEN 'Resource'

ELSE 'Limited Resource'

END AS description

FROM employeeskill

GROUP BY emp\_id;

-- 47. Generate a report: Emp ID | No of leaves | Description

-- (i) If no of leaves is greater than 6, Description is “Loss of Pay”

-- (ii) If no of leaves is less than or equal to 6, Description is “No loss of pay”

-- (iii) If no of leaves is equal to zero, Description is “Bonus”

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS report\_employee\_leaves;

DELIMITER $$

CREATE PROCEDURE report\_employee\_leaves()

BEGIN

SELECT emp\_id, total\_leaves,

CASE

WHEN total\_leaves = 0 THEN 'Bonus'

WHEN total\_leaves > 6 THEN 'Loss of Pay'

ELSE 'No loss of pay'

END AS description

FROM (

SELECT emp\_id, (casual\_leave + sick\_leave + privilege\_leave) AS total\_leaves

FROM employee

) AS leave\_summary;

END $$

DELIMITER ;

CALL report\_employee\_leaves();

-- Method 2: Using CASE

SELECT emp\_id, (casual\_leave + sick\_leave + privilege\_leave) AS total\_leaves,

CASE

WHEN (casual\_leave + sick\_leave + privilege\_leave) = 0 THEN 'Bonus'

WHEN (casual\_leave + sick\_leave + privilege\_leave) > 6 THEN 'Loss of Pay'

ELSE 'No loss of pay'

END AS description

FROM employee;

-- 48. List TOP 5 salaried Employees?

-- Method 1: Using RANK Function

SELECT emp\_id, emp\_name, salary

FROM (

SELECT emp\_id, emp\_name, salary,

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

FROM employee

) t

WHERE rnk <= 5;

-- Method 2: Using NOT IN

SELECT e1.emp\_id, e1.emp\_name, e1.salary

FROM employee e1

WHERE e1.emp\_id NOT IN (

SELECT e2.emp\_id

FROM employee e2

WHERE (

SELECT COUNT(DISTINCT e3.salary)

FROM employee e3

WHERE e3.salary > e2.salary

) >= 5

)

ORDER BY e1.salary DESC;

-- Method 3: Using LEFT JOIN

SELECT e1.emp\_id, e1.emp\_name, e1.salary

FROM employee e1

LEFT JOIN employee e2

ON e1.salary < e2.salary

GROUP BY e1.emp\_id, e1.emp\_name, e1.salary

HAVING COUNT(DISTINCT e2.emp\_id) < 5

ORDER BY e1.salary DESC;

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

SELECT e1.emp\_id, e1.emp\_name, e1.salary

FROM employee e1

WHERE NOT EXISTS (

SELECT 1

FROM employee e2

WHERE (

SELECT COUNT(DISTINCT e3.salary)

FROM employee e3

WHERE e3.salary > e2.salary

) < 5

AND e2.salary > e1.salary

)

ORDER BY e1.salary DESC;

/\* Does not work in MySQL

-- Method 5: Using EXCEPT (SET operators)

SELECT emp\_id, emp\_name, salary

FROM employee

EXCEPT

SELECT e1.emp\_id, e1.emp\_name, e1.salary

FROM employee e1

WHERE (

SELECT COUNT(DISTINCT e2.salary)

FROM employee e2

WHERE e2.salary > e1.salary

) >= 5;

-- Method 6: Using INTERSECT (SET operators)

SELECT emp\_id, emp\_name, salary

FROM employee

INTERSECT

SELECT emp\_id, emp\_name, salary

FROM (

SELECT emp\_id, emp\_name, salary,

DENSE\_RANK() OVER (ORDER BY salary DESC) AS salary\_rank

FROM employee

) ranked

WHERE salary\_rank <= 5;

\*/

-- 49. List TOP 3 Departments (with respect to maximum no of employees)?

-- Method 1: Using RANK Function

SELECT dept\_id, number\_of\_employees

FROM (

SELECT dept\_id, COUNT(emp\_id) AS number\_of\_employees,

RANK() OVER (ORDER BY COUNT(emp\_id) DESC) AS rnk

FROM employee

GROUP BY dept\_id

) t

WHERE rnk <= 3;

-- Method 2: Using DENSE\_RANK Function

SELECT dept\_id, number\_of\_employees

FROM (

SELECT dept\_id, COUNT(emp\_id) AS number\_of\_employees,

DENSE\_RANK() OVER (ORDER BY COUNT(emp\_id) DESC) AS dense\_rnk

FROM employee

GROUP BY dept\_id

) t

WHERE dense\_rnk <= 3;

-- Method 3: Using COUNT

SELECT e1.dept\_id, COUNT(e1.emp\_id) AS number\_of\_employees

FROM employee e1

GROUP BY e1.dept\_id

HAVING (

SELECT COUNT(DISTINCT COUNT(e2.emp\_id))

FROM employee e2

GROUP BY e2.dept\_id

HAVING COUNT(e2.emp\_id) >= COUNT(e1.emp\_id)

) <= 3;

-- Method 4: Using UNION (SET operators)

SELECT dept\_id, COUNT(emp\_id) AS number\_of\_employees

FROM employee

GROUP BY dept\_id

HAVING COUNT(emp\_id) = (

SELECT MAX(emp\_count)

FROM (

SELECT COUNT(emp\_id) AS emp\_count

FROM employee

GROUP BY dept\_id

) sub

)

UNION

SELECT dept\_id, COUNT(emp\_id) AS number\_of\_employees

FROM employee

GROUP BY dept\_id

HAVING COUNT(emp\_id) = (

SELECT MAX(emp\_count)

FROM (

SELECT COUNT(emp\_id) AS emp\_count

FROM employee

GROUP BY dept\_id

HAVING COUNT(emp\_id) < (

SELECT MAX(emp\_count)

FROM (

SELECT COUNT(emp\_id) AS emp\_count

FROM employee

GROUP BY dept\_id

) sub2

)

) sub1

)

UNION

SELECT dept\_id, COUNT(emp\_id) AS number\_of\_employees

FROM employee

GROUP BY dept\_id

HAVING COUNT(emp\_id) = (

SELECT MAX(emp\_count)

FROM (

SELECT COUNT(emp\_id) AS emp\_count

FROM employee

GROUP BY dept\_id

HAVING COUNT(emp\_id) < (

SELECT MAX(emp\_count)

FROM (

SELECT COUNT(emp\_id) AS emp\_count

FROM employee

GROUP BY dept\_id

HAVING COUNT(emp\_id) < (

SELECT MAX(emp\_count)

FROM (

SELECT COUNT(emp\_id) AS emp\_count

FROM employee

GROUP BY dept\_id

) sub3

)

) sub2

)

) sub1

);

/\* Does not work in MySQL

-- Method 5: Using EXCEPT (SET operators)

WITH DepartmentEmployeeCounts AS (

SELECT dept\_id, COUNT(emp\_id) AS number\_of\_employees

FROM employee

GROUP BY dept\_id

),

RankedDepartments AS (

SELECT dept\_id, number\_of\_employees,

RANK() OVER (ORDER BY number\_of\_employees DESC) AS rnk

FROM DepartmentEmployeeCounts

)

SELECT dept\_id, number\_of\_employees

FROM RankedDepartments

EXCEPT

SELECT dept\_id, number\_of\_employees

FROM RankedDepartments

WHERE rnk > 3;

-- Method 6: Using INTERSECT (SET operators)

WITH DepartmentEmployeeCounts AS (

SELECT dept\_id, COUNT(emp\_id) AS number\_of\_employees

FROM employee

GROUP BY dept\_id

),

RankedDepartments\_Rank AS (

SELECT dept\_id, number\_of\_employees,

RANK() OVER (ORDER BY number\_of\_employees DESC) AS rnk

FROM DepartmentEmployeeCounts

),

RankedDepartments\_DenseRank AS (

SELECT dept\_id, number\_of\_employees,

DENSE\_RANK() OVER (ORDER BY number\_of\_employees DESC) AS dense\_rnk

FROM DepartmentEmployeeCounts

)

SELECT dept\_id, number\_of\_employees

FROM RankedDepartments\_Rank

WHERE rnk <= 3

INTERSECT

SELECT dept\_id, number\_of\_employees

FROM RankedDepartments\_DenseRank

WHERE dense\_rnk <= 3;

\*/

-- 50. For each Department display the 2nd maximum salary earner?

-- Method 1: Using RANK Function

SELECT emp\_id, emp\_name, dept\_id, salary

FROM (

SELECT emp\_id, emp\_name, dept\_id, salary,

RANK() OVER (PARTITION BY dept\_id ORDER BY salary DESC) AS rnk

FROM employee

) t

WHERE rnk = 2;

-- Method 2: Using NOT IN

SELECT emp\_id, emp\_name, dept\_id, salary

FROM employee e1

WHERE salary = (

SELECT MAX(salary)

FROM employee e2

WHERE e2.dept\_id = e1.dept\_id

AND e2.salary NOT IN (

SELECT MAX(salary)

FROM employee e3

WHERE e3.dept\_id = e1.dept\_id

)

);

-- Method 3: Using LEFT JOIN

SELECT e1.emp\_id, e1.emp\_name, e1.dept\_id, e1.salary

FROM employee e1

JOIN employee e2

ON e1.dept\_id = e2.dept\_id AND e1.salary < e2.salary

LEFT JOIN employee e3

ON e1.dept\_id = e3.dept\_id AND e1.salary < e3.salary AND e2.salary < e3.salary

WHERE e3.emp\_id IS NULL

GROUP BY e1.emp\_id, e1.emp\_name, e1.dept\_id, e1.salary

HAVING COUNT(DISTINCT e2.salary) = 1;

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

SELECT e1.emp\_id, e1.emp\_name, e1.dept\_id, e1.salary

FROM employee e1

WHERE EXISTS (

SELECT 1

FROM employee e2

WHERE e2.dept\_id = e1.dept\_id

AND e2.salary > e1.salary

)

AND NOT EXISTS (

SELECT 1

FROM employee e3

WHERE e3.dept\_id = e1.dept\_id

AND e3.salary > e1.salary

HAVING COUNT(DISTINCT e3.salary) >= 2

);

/\* Does not work in MySQL

-- Method 5: Using EXCEPT (SET operators)

WITH DepartmentSalaries AS (

SELECT emp\_id, emp\_name, dept\_id, salary, RANK() OVER (PARTITION BY dept\_id ORDER BY salary DESC) AS rnk

FROM employee

)

SELECT emp\_id, emp\_name, dept\_id, salary

FROM DepartmentSalaries

EXCEPT

SELECT emp\_id, emp\_name, dept\_id, salary

FROM DepartmentSalaries

WHERE rnk <> 2;

-- Method 6: Using INTERSECT (SET operators)

WITH DepartmentSalaries AS (

SELECT emp\_id, emp\_name, dept\_id, salary, RANK() OVER (PARTITION BY dept\_id ORDER BY salary DESC) AS rnk

FROM employee

)

-- Employees who are NOT the highest earners (rnk > 1)

SELECT emp\_id, emp\_name, dept\_id, salary

FROM DepartmentSalaries

WHERE rnk > 1

INTERSECT

-- Employees who are NOT the 3rd highest or lower earners (rnk <= 2)

SELECT emp\_id, emp\_name, dept\_id, salary

FROM DepartmentSalaries

WHERE rnk <= 2;

\*/

-- 51. Generate a report: Emp Name | Number of skills

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS GetEmployeeSkillsCount;

DELIMITER $$

CREATE PROCEDURE GetEmployeeSkillsCount()

BEGIN

SELECT e.emp\_name, COUNT(es.skill\_id) AS number\_of\_skills

FROM employee e

LEFT JOIN employeeskill es

ON e.emp\_id = es.emp\_id

GROUP BY e.emp\_name;

END $$

DELIMITER ;

CALL GetEmployeeSkillsCount();

-- Method 2: Using LEFT JOIN

SELECT e.emp\_name, COUNT(es.skill\_id) AS number\_of\_skills

FROM employee e

LEFT JOIN employeeskill es

ON e.emp\_id = es.emp\_id

GROUP BY e.emp\_name;

-- Method 3: Using NOT IN

SELECT e.emp\_name, (SELECT COUNT(skill\_id) FROM employeeskill es WHERE es.emp\_id = e.emp\_id) AS number\_of\_skills

FROM employee e

WHERE e.emp\_id NOT IN (

SELECT emp\_id

FROM employeeskill

WHERE emp\_id IS NULL

)

GROUP BY e.emp\_name;

-- Method 4: Using Subquery

SELECT e.emp\_name, (SELECT COUNT(skill\_id) FROM employeeskill es WHERE es.emp\_id = e.emp\_id) AS number\_of\_skills

FROM employee e

GROUP BY e.emp\_name;

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

SELECT e.emp\_name, (SELECT COUNT(skill\_id) FROM employeeskill es WHERE es.emp\_id = e.emp\_id) AS number\_of\_skills

FROM employee e

WHERE EXISTS (

SELECT 1

FROM employeeskill es

WHERE es.emp\_id = e.emp\_id

)

OR NOT EXISTS (

SELECT 1

FROM employeeskill es

WHERE es.emp\_id = e.emp\_id

)

GROUP BY e.emp\_name;

-- Method 6: Using UNION (SET operators)

SELECT e.emp\_name, COUNT(es.skill\_id) AS number\_of\_skills

FROM employee e

JOIN employeeskill es

ON e.emp\_id = es.emp\_id

GROUP BY e.emp\_name

UNION

SELECT e.emp\_name, 0 AS number\_of\_skills

FROM employee e

WHERE e.emp\_id NOT IN (

SELECT emp\_id

FROM employeeskill

)

GROUP BY e.emp\_name;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT e.emp\_name, COUNT(es.skill\_id) AS number\_of\_skills

FROM employee e

JOIN employeeskill es ON e.emp\_id = es.emp\_id

GROUP BY e.emp\_name

EXCEPT

SELECT e.emp\_name, 0 AS number\_of\_skills

FROM employee e

WHERE e.emp\_id NOT IN (

SELECT emp\_id

FROM employeeskill

)

GROUP BY e.emp\_name;

-- Method 8: Using INTERSECT (SET operators)

SELECT e.emp\_name, COUNT(es.skill\_id) AS number\_of\_skills

FROM employee e

JOIN employeeskill es

ON e.emp\_id = es.emp\_id

GROUP BY e.emp\_name

INTERSECT

SELECT e.emp\_name, COUNT(es.skill\_id) AS number\_of\_skills

FROM employeeskill es

JOIN employee e

ON es.emp\_id = e.emp\_id

GROUP BY e.emp\_name;

\*/

-- 52. Generate the below report: Month wise, No. of employees allocated to Project.

-- Project name | Jan | Feb | Mar | Apr | May | Jun

-- Banking 5 6 4 2 3 1

-- Insurance 2 5 4 3 4 3

-- Method 1: Using CASE

SELECT t\_project.project\_name,

SUM(CASE WHEN EXTRACT(month FROM from\_date) = 1 THEN 1 ELSE 0 END) AS Jan,

SUM(CASE WHEN EXTRACT(month FROM from\_date) = 2 THEN 1 ELSE 0 END) AS Feb,

SUM(CASE WHEN EXTRACT(month FROM from\_date) = 3 THEN 1 ELSE 0 END) AS Mar,

SUM(CASE WHEN EXTRACT(month FROM from\_date) = 4 THEN 1 ELSE 0 END) AS Apr,

SUM(CASE WHEN EXTRACT(month FROM from\_date) = 5 THEN 1 ELSE 0 END) AS May,

SUM(CASE WHEN EXTRACT(month FROM from\_date) = 6 THEN 1 ELSE 0 END) AS Jun

FROM allocation

JOIN t\_project ON allocation.project\_id = t\_project.project\_id

GROUP BY t\_project.project\_name;

/\* Does not work in MYSQL

-- Method 2: Using PIVOT

SELECT project\_name, [1] AS Jan, [2] AS Feb, [3] AS Mar, [4] AS Apr, [5] AS May, [6] AS Jun

FROM (

SELECT tp.project\_name, MONTH(from\_date) AS month

FROM allocation AS a

JOIN t\_project AS tp

ON a.project\_id = tp.project\_id

) AS sourcetable

PIVOT (

COUNT(month)

FOR month IN ([1], [2], [3], [4], [5], [6])

) AS pivottable;

Explaination:

The inner query named sourcetable:

- Joins the allocation and t\_project tables.

- Selects each project's name and the month number (1–12) extracted from from\_date column in allocation table.

- Result is a list of project names with their allocation months.

PIVOT clause:

- Pivots the month values into columns.

- Uses COUNT(month) to count how many times each project appears in each month.

- FOR month IN (...) lists the target columns, e.g., [1] for January, [2] for February, etc.

The outer query:

- Renames the numeric month columns to readable names (Jan, Feb, etc.).

- Outputs one row per project with allocation counts for each of the first 6 months.

\*/

-- 53. Generate the report. Month wise, No. of projects allocated to Employee.

-- Employeename | Jan | Feb | Mar | Apr | May | Jun

-- Ram | 5 | 6 | 4 | 2 | 3 | 1

-- Arjun | 2 | 5 | 4 | 3 | 4 | 3

-- Pravin | 6 | 3 | 5 | 2 | 2 | 3

-- Method 1: Using CASE

SELECT employee.emp\_name,

SUM(CASE WHEN EXTRACT(month FROM from\_date) = 1 THEN 1 ELSE 0 END) AS jan,

SUM(CASE WHEN EXTRACT(month FROM from\_date) = 2 THEN 1 ELSE 0 END) AS feb,

SUM(CASE WHEN EXTRACT(month FROM from\_date) = 3 THEN 1 ELSE 0 END) AS mar,

SUM(CASE WHEN EXTRACT(month FROM from\_date) = 4 THEN 1 ELSE 0 END) AS apr,

SUM(CASE WHEN EXTRACT(month FROM from\_date) = 5 THEN 1 ELSE 0 END) AS may,

SUM(CASE WHEN EXTRACT(month FROM from\_date) = 6 THEN 1 ELSE 0 END) AS jun

FROM allocation

JOIN employee ON allocation.emp\_id = employee.emp\_id

GROUP BY employee.emp\_name;

/\* Does not work in MYSQL

-- Method 2: Using PIVOT

SELECT emp\_name, [1] AS Jan, [2] AS Feb, [3] AS Mar, [4] AS Apr, [5] AS May, [6] AS Jun

FROM (

SELECT e.emp\_name, MONTH(from\_date) AS month

FROM allocation AS a

JOIN employee as e

ON a.emp\_id = e.emp\_id

) AS sourcetable

PIVOT (

COUNT(month)

FOR month IN ([1], [2], [3], [4], [5], [6])

) AS pivottable;

\*/