* **QUERIES AND THEIR OUTPUT:**

**1b) Query:**SELECT DISTINCT C.cname, W.pid, W.salary

FROM Company C

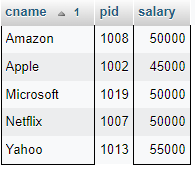
JOIN worksFor W ON C.cname = W.cname

JOIN Knows K ON W.pid = K.pid1

JOIN (SELECT pid FROM personSkill WHERE skill = 'OperatingSystems') OS ON K.pid2 = OS.pid

WHERE W.salary = (SELECT MIN(W2.salary) FROM worksFor W2 WHERE W2.cname = C.cname)

ORDER BY C.cname, W.pid;  
**Output/ Result:**



**2b) Query:**SELECT P.pname, W.salary, P.city

FROM Person P

JOIN worksFor W ON P.pid = W.pid

LEFT JOIN (

SELECT DISTINCT P2.pid, P2.city

FROM Person P2

JOIN personSkill PS ON P2.pid = PS.pid

WHERE PS.skill = 'Networks'

) NetworkCities ON P.city = NetworkCities.city AND P.pid = NetworkCities.pid

WHERE NetworkCities.pid IS NULL

AND W.salary = (

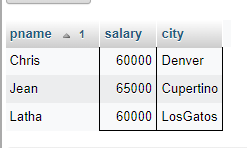
SELECT MAX(W2.salary)

FROM worksFor W2

WHERE W2.cname = W.cname

)

ORDER BY P.pname;  
**Output/ Result:**

****

**3b) Query:**SELECT DISTINCT W1.cname AS c1, W2.cname AS c2

FROM worksFor W1

JOIN worksFor W2 ON W1.cname < W2.cname

LEFT JOIN Person P1 ON P1.pid = W1.pid AND P1.city = 'Chicago'

LEFT JOIN Person P2 ON P2.pid = W2.pid AND P2.city = 'Chicago'

WHERE P1.pid IS NULL AND P2.pid IS NULL

ORDER BY c1, c2;

**Output/ Result:**



**12) Query:**CREATE VIEW CompanyKnownPerson AS

SELECT DISTINCT W1.pid AS known\_person

FROM worksFor W1

JOIN Knows K1 ON W1.pid = K1.pid1

JOIN Knows K2 ON K1.pid2 = K2.pid2 AND K1.pid1 <> K2.pid1

JOIN worksFor W2 ON K2.pid1 = W2.pid AND W1.cname = W2.cname AND W1.salary > W2.salary

WHERE W1.pid IN (1001, 1015)

GROUP BY W1.pid

HAVING COUNT(DISTINCT K1.pid2) >= 2;

SELECT \* FROM CompanyKnownPerson;

**Output/Result:**



**13) Query:**DELIMITER //

CREATE PROCEDURE SkillOnlyOnePerson(IN skill1 TEXT)

BEGIN

-- Create a temporary table to store the result

CREATE TEMPORARY TABLE TempSkillOnlyOnePerson AS

SELECT DISTINCT PS1.pid AS pid1, PS2.pid AS pid2

FROM personSkill PS1

JOIN personSkill PS2 ON PS1.pid <> PS2.pid

LEFT JOIN personSkill PS3 ON PS3.pid = PS2.pid AND PS3.skill = skill1

WHERE PS1.skill = skill1 AND PS3.pid IS NULL;

-- Query the temporary table

SELECT \* FROM TempSkillOnlyOnePerson;

-- Drop the temporary table

DROP TEMPORARY TABLE TempSkillOnlyOnePerson;

END //

DELIMITER ;

-- Call the stored procedure to get the result for 'WebDevelopment'

CALL SkillOnlyOnePerson('WebDevelopment');

**Output/Result:**

****

**14) Query:**

DELIMITER //

CREATE PROCEDURE RecursiveSameGeneration()

BEGIN

CREATE TEMPORARY TABLE TempSameGeneration AS

SELECT parent AS n1, child AS n2

FROM PC;

REPEAT

INSERT IGNORE INTO TempSameGeneration

SELECT P1.n1 AS n1, P2.child AS n2

FROM TempSameGeneration P1

JOIN PC P2 ON P1.n2 = P2.parent;

UNTIL ROW\_COUNT() = 0 END REPEAT;

SELECT \* FROM TempSameGeneration;

DROP TEMPORARY TABLE TempSameGeneration;

END //

DELIMITER ;

-- Call the stored procedure to get the result for sameGeneration

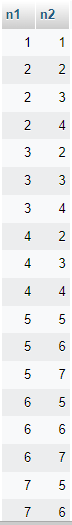
CALL RecursiveSameGeneration();

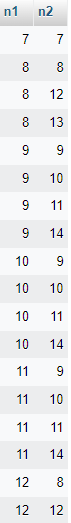
SELECT \* FROM TempSameGeneration;

-- Drop the temporary table

DROP TEMPORARY TABLE TempSameGeneration;

**Output/Result:**







**15) Query:**CREATE VIEW Inheritance AS

WITH RECURSIVE InheritanceCTE AS (

-- Base case: Nodes with missing parents

SELECT child\_id, gold\_accumulated

FROM Hierarchy

WHERE parent\_id IS NULL

UNION ALL

-- Recursive case: Calculate gold inheritance for each child

SELECT H.child\_id, I.gold\_accumulated + H.gold\_accumulated

FROM Hierarchy H

JOIN InheritanceCTE I ON H.parent\_id = I.child\_id

)

-- Final select from the CTE

SELECT \* FROM InheritanceCTE;

SELECT \* FROM Inheritance;

**Output/Result:**

****