* **QUERIES AND THEIR OUTPUT:**

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

FROM Company

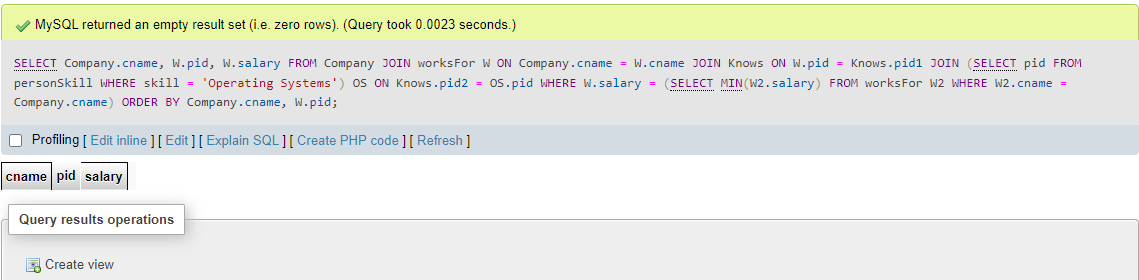
JOIN worksFor W ON Company.cname = W.cname

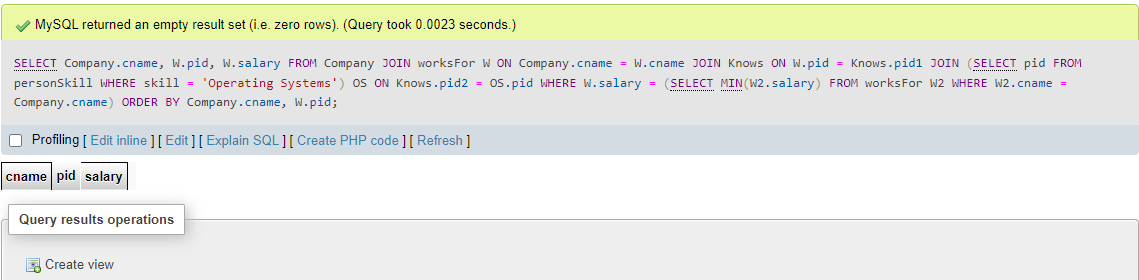
JOIN Knows ON W.pid = Knows.pid1

JOIN (SELECT pid FROM personSkill WHERE skill = 'Operating Systems') OS ON Knows.pid2 = OS.pid

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

ORDER BY Company.cname, W.pid;  
  
**Output/ Result:***Empty Result Set*

****



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

FROM Person P

JOIN worksFor W ON P.pid = W.pid

WHERE P.city NOT IN (

SELECT DISTINCT P2.city

FROM Person P2

JOIN personSkill PS ON P2.pid = PS.pid

WHERE PS.skill = 'Networks'

)

AND W.salary = (

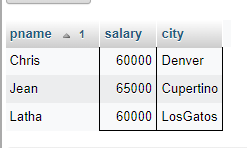
SELECT MAX(W2.salary)

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

WHERE NOT EXISTS (

SELECT 1

FROM Person P1

WHERE P1.pid = W1.pid AND P1.city = 'Chicago'

)

AND NOT EXISTS (

SELECT 1

FROM Person P2

WHERE P2.pid = W2.pid AND P2.city = 'Chicago'

)

ORDER BY c1, c2;  
  
**Output/ Result:**



**12) Query:**CREATE VIEW IF NOT EXISTS 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 K1.pid2 = W2.pid

WHERE W1.cname = W2.cname AND W1.salary > W2.salary;

-- Test your view

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 PS1.pid AS pid1, PS2.pid AS pid2

FROM personSkill PS1

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

WHERE PS1.skill = skill1 AND NOT EXISTS (

SELECT 1

FROM personSkill PS3

WHERE PS3.pid = PS2.pid AND PS3.skill = skill1

);

-- Query the temporary table

SELECT \* FROM TempSkillOnlyOnePerson;

-- Drop the temporary table

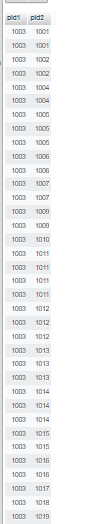
DROP TEMPORARY TABLE IF EXISTS TempSkillOnlyOnePerson;

END //

DELIMITER ;

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

CALL SkillOnlyOnePerson('WebDevelopment');  
**Output/Result:**

****

**14) Query:**-- Drop the existing stored procedure if it exists

DROP PROCEDURE IF EXISTS RecursiveSameGeneration;

-- Create a temporary table to store the intermediate results

CREATE TEMPORARY TABLE TempSameGeneration AS

SELECT parent, child

FROM PC;

-- Create a stored procedure for recursive processing

DELIMITER //

CREATE PROCEDURE RecursiveSameGeneration()

BEGIN

DECLARE done INT DEFAULT FALSE;

REPEAT

-- Insert the next generation into the temporary table

INSERT IGNORE INTO TempSameGeneration

SELECT PC.parent, PC.child

FROM PC

JOIN TempSameGeneration SG ON PC.child = SG.parent;

-- Check if no new rows were inserted

SET done = ROW\_COUNT() = 0;

UNTIL done END REPEAT;

END //

DELIMITER ;

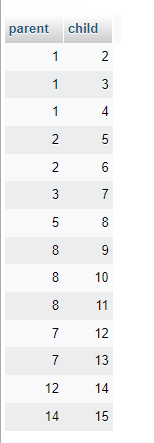
CALL RecursiveSameGeneration ();

-- Query the temporary table

SELECT \* FROM TempSameGeneration;

-- Drop the temporary table

DROP TEMPORARY TABLE IF EXISTS TempSameGeneration;  
 **Output/Result:**



**15) Query:**-- Drop the existing stored procedure if it exists

DROP PROCEDURE IF EXISTS RecursiveInheritance;

-- Create a temporary table to store the intermediate results

CREATE TEMPORARY TABLE TempInheritance AS

SELECT child\_id AS m, gold\_accumulated AS p

FROM Hierarchy

WHERE parent\_id IS NULL;

-- Create a stored procedure for recursive processing

DELIMITER //

CREATE PROCEDURE RecursiveInheritance()

BEGIN

DECLARE done INT DEFAULT FALSE;

REPEAT

-- Insert the next generation into the temporary table

INSERT IGNORE INTO TempInheritance

SELECT H2.child\_id AS m, H1.p + H2.gold\_accumulated AS p

FROM TempInheritance H1

JOIN Hierarchy H2 ON H1.m = H2.parent\_id;

-- Check if no new rows were inserted

SET done = ROW\_COUNT() = 0;

UNTIL done END REPEAT;

END //

DELIMITER ;

CALL RecursiveInheritance();

-- Query the temporary table

SELECT \* FROM TempInheritance;

-- Drop the temporary table

DROP TEMPORARY TABLE IF EXISTS TempInheritance;  
  
**Output/Result:**

****