```
DROP DATABASE IF EXISTS assignment2;

CREATE DATABASE assignment2;

\c assignment2;


CREATE TABLE student(sid integer,
                     sname text,
                     city text,
                     primary key (sid));
CREATE TABLE Company(cname text,
                      headquarter text,
                      primary key (cname));
CREATE TABLE Skill(skill text,
                    primary key (skill));
CREATE TABLE worksFor(sid integer,
                       cname text,
                       salary integer,
                       primary key (sid),
                       foreign key (sid) references student (sid),
                       foreign key (cname) references Company(cname));
CREATE TABLE companyLocation(cname text,
                             city text,
                             primary key (cname, city),
                             foreign key (cname) references Company (cname));
CREATE TABLE studentSkill(sid integer,
                          skill text,
                          primary key (sid, skill),
                          foreign key (sid) references student (sid) on delete
cascade,
                          foreign key (skill) references Skill (skill) on delete
cascade);
CREATE TABLE hasManager(eid integer,
                        mid integer,
                        primary key (eid, mid),
                        foreign key (eid) references student (sid),
                        foreign key (mid) references student (sid));
CREATE TABLE Knows(sid1 integer,
                   sid2 integer,
                   primary key(sid1, sid2),
                   foreign key (sid1) references student (sid),
                   foreign key (sid2) references student (sid));

INSERT INTO student VALUES
     (1001,'Jean','Cupertino'),
     (1002,'Vidya', 'Cupertino'),
     (1003,'Anna', 'Seattle'),
     (1004,'Qin', 'Seattle'),
     (1005,'Megan', 'MountainView'),
     (1006,'Ryan', 'Chicago'),
     (1007,'Danielle','LosGatos'),
     (1008,'Emma', 'Bloomington'),
     (1009,'Hasan', 'Bloomington'),
     (1010,'Linda', 'Chicago'),
     (1011,'Nick', 'MountainView'),
     (1012,'Eric', 'Cupertino'),
     (1013,'Lisa', 'Indianapolis'),
     (1014,'Deepa', 'Bloomington'),
```

```
     (1015,'Chris', 'Denver'),
     (1016,'YinYue', 'Chicago'),
     (1017,'Latha', 'LosGatos'),
     (1018,'Arif', 'Bloomington'),
     (1019,'John', 'NewYork');

INSERT INTO Company VALUES
     ('Apple', 'Cupertino'),
     ('Amazon', 'Seattle'),
     ('Google', 'MountainView'),
     ('Netflix', 'LosGatos'),
     ('Microsoft', 'Redmond'),
     ('IBM', 'NewYork'),
     ('ACM', 'NewYork'),
     ('Yahoo', 'Sunnyvale');


INSERT INTO worksFor VALUES
     (1001,'Apple', 65000),
     (1002,'Apple', 45000),
     (1003,'Amazon', 55000),
     (1004,'Amazon', 55000),
     (1005,'Google', 60000),
     (1006,'Amazon', 55000),
     (1007,'Netflix', 50000),
     (1008,'Amazon', 50000),
     (1009,'Apple',60000),
     (1010,'Amazon', 55000),
     (1011,'Google', 70000),
     (1012,'Apple', 50000),
     (1013,'Yahoo', 55000),
     (1014,'Apple', 50000),
     (1015,'Amazon', 60000),
     (1016,'Amazon', 55000),
     (1017,'Netflix', 60000),
     (1018,'Apple', 50000),
     (1019,'Microsoft', 50000);

INSERT INTO companyLocation VALUES
   ('Apple', 'Bloomington'),
   ('Amazon', 'Chicago'),
   ('Amazon', 'Denver'),
   ('Amazon', 'Columbus'),
   ('Google', 'NewYork'),
   ('Netflix', 'Indianapolis'),
   ('Netflix', 'Chicago'),
   ('Microsoft', 'Bloomington'),
   ('Apple', 'Cupertino'),
   ('Amazon', 'Seattle'),
   ('Google', 'MountainView'),
   ('Netflix', 'LosGatos'),
   ('Microsoft', 'Redmond'),
   ('IBM', 'NewYork'),
   ('Yahoo', 'Sunnyvale');

INSERT INTO Skill VALUES
   ('Programming'),
   ('AI'),
   ('Networks'),
```

```
   ('OperatingSystems'),
   ('Databases'),
   ('WebDevelopment');

INSERT INTO studentSkill VALUES
 (1001,'Programming'),
 (1001,'AI'),
 (1002,'Programming'),
 (1002,'AI'),
 (1004,'AI'),
 (1004,'Programming'),
 (1005,'AI'),
 (1005,'Programming'),
 (1005,'Networks'),
 (1006,'Programming'),
 (1006,'OperatingSystems'),
 (1007,'OperatingSystems'),
 (1007,'Programming'),
 (1009,'OperatingSystems'),
 (1009,'Networks'),
 (1010,'Networks'),
 (1011,'Networks'),
 (1011,'OperatingSystems'),
 (1011,'AI'),
 (1011,'Programming'),
 (1012,'AI'),
 (1012,'OperatingSystems'),
 (1012,'Programming'),
 (1013,'Programming'),
 (1013,'OperatingSystems'),
 (1013,'Networks'),
 (1014,'OperatingSystems'),
 (1014,'AI'),
 (1014,'Networks'),
 (1015,'Programming'),
 (1015,'AI'),
 (1016,'OperatingSystems'),
 (1016,'AI'),
 (1017,'Programming'),
 (1018,'AI'),
 (1019,'Networks'),
 (1003,'WebDevelopment');

INSERT INTO hasManager VALUES
 (1004, 1003),
 (1006, 1003),
 (1015, 1003),
 (1016, 1004),
 (1016, 1006),
 (1008, 1015),
 (1010, 1008),
 (1007, 1017),
 (1002, 1001),
 (1009, 1001),
 (1014, 1012),
 (1011, 1005);

TRUNCATE TABLE Knows;
```

```
INSERT INTO Knows VALUES
 (1011,1009),
 (1007,1016),
 (1011,1010),
 (1003,1004),
 (1006,1004),
 (1002,1014),
 (1009,1005),
 (1018,1009),
 (1007,1017),
 (1017,1019),
 (1019,1013),
 (1016,1015),
 (1001,1012),
 (1015,1011),
 (1019,1006),
 (1013,1002),
 (1018,1004),
 (1013,1007),
 (1014,1006),
 (1004,1014),
 (1001,1014),
 (1010,1013),
 (1010,1014),
 (1004,1019),
 (1018,1007),
 (1014,1005),
 (1015,1018),
 (1014,1017),
 (1013,1018),
 (1007,1008),
 (1005,1015),
 (1017,1014),
 (1015,1002),
 (1018,1013),
 (1018,1010),
 (1001,1008),
 (1012,1011),
 (1002,1015),
 (1007,1013),
 (1008,1007),
 (1004,1002),
 (1015,1005),
 (1009,1013),
 (1004,1012),
 (1002,1011),
 (1004,1013),
 (1008,1001),
 (1008,1019),
 (1019,1008),
 (1001,1019),
 (1019,1001),
 (1004,1003),
 (1006,1003),
 (1015,1003),
 (1016,1004),
 (1016,1006),
 (1008,1015),
 (1010,1008),
```

```
 (1017,1013),
 (1002,1001),
 (1009,1001),
 (1011,1005),
 (1014,1012),
 (1012,1001),
 (1014,1001),
 (1018,1001),
 (1001,1001),
 (1002,1002),
 (1003,1003),
 (1004,1004),
 (1005,1005),
 (1006,1006),
 (1007,1007),
 (1008,1008),
 (1009,1009),
 (1010,1010),
 (1011,1011),
 (1012,1012),
 (1013,1013),
 (1014,1014),
 (1015,1015),
 (1016,1016),
 (1017,1017),
 (1018,1018),
 (1019,1019);


create table PC(parent integer,
                child integer);
insert into PC values
(1,2),
(1,3),
(1,4),
(2,5),
(2,6),
(3,7),
(5,8),
(8,9),
(8,10),
(8,11),
(7,12),
(7,13),
(12,14),
(14,15);


-- Find each triple (c, s, sl) where c is the
-- cname of a company, s is the sid of a student who earns the low-
-- est salary at that company and knows at least someone who has
-- Operating Systems skill, and sl is the salary of s.

\qecho 'Problem-1a'
--Formulate this query in Pure SQL by only using the
--EXISTS or NOT EXISTS set predicates. You can not use the set operations
INTERSECT, UNION, and EXCEPT.
SELECT w.cname, w.sid, w.salary
FROM worksfor w
```

```
WHERE NOT EXISTS(
      SELECT 1
      FROM worksFor w1
      WHERE w1.cname = w.cname
      AND w.salary > w1. salary
)
AND EXISTS(
      SELECT 1
      FROM knows k
      WHERE w.sid = k.sid1
      AND EXISTS(
            SELECT 1
            FROM studentskill sk
            WHERE k.sid2 = sk.sid
            AND sk.skill = 'OperatingSystems'
      )
);


\qecho 'Problem-1b'
--Formulate this query in Pure SQL by only using the IN, NOT IN, SOME, or ALL set
membership predicates.
--You can not use the set operations INTERSECT, UNION, and EXCEPT.
SELECT w.cname, w.sid, w.salary
FROM worksFor w
WHERE w.salary <= ALL (
    SELECT w1.salary
    FROM worksFor w1
    WHERE w1.cname = w.cname
)
AND w.sid IN (
    SELECT k.sid1
    FROM Knows k
    WHERE k.sid2 IN (
        SELECT sk.sid
        FROM studentSkill sk
        WHERE sk.skill = 'OperatingSystems'
    )
);


\qecho 'Problem-1c'
--Formulate this query in Pure SQL by only using the set operations INTERSECT,
UNION, and EXCEPT.
(
    SELECT w.cname AS c, w.sid AS s, w.salary AS sl
    FROM worksFor w
    EXCEPT
    SELECT w1.cname AS c, w1.sid AS s, w1.salary AS sl
    FROM worksFor w1, worksFor w2
    WHERE w1.cname = w2.cname
      AND w1.salary > w2.salary
)
INTERSECT
SELECT w.cname AS c, w.sid AS s, w.salary AS sl
FROM worksFor w, Knows k, StudentSkill sk
WHERE k.sid1 = w.sid
  AND sk.sid = k.sid2
  AND sk.skill = 'OperatingSystems';
```

```
-- Find the name, salary and city of each student who (a) lives in a city where no
one has the Networks skill
-- and (b) earns the highest salary in his/her company.

\qecho 'Problem-2a'
--Formulate this query in Pure SQL by only using the
--EXISTS or NOT EXISTS set predicates. You can not use the set operations
INTERSECT, UNION, and EXCEPT.
SELECT s.sname, w.salary, s.city
FROM Student s, worksFor w
WHERE s.sid = w.sid
AND NOT EXISTS (
    SELECT 1
    FROM Student s2, studentSkill ss
    WHERE s2.city = s.city
    AND ss.sid = s2.sid
    AND ss.skill = 'Networks'
)
AND NOT EXISTS (
    SELECT 1
    FROM worksFor w2
    WHERE w2.cname = w.cname
    AND w2.salary > w.salary
);


\qecho 'Problem-2b'
--Formulate this query in Pure SQL by only using the IN, NOT IN, SOME, or ALL set
membership predicates.
--You can not use the set operations INTERSECT, UNION, and EXCEPT.
SELECT s.sname, w.salary, s.city
FROM Student s, worksFor w
WHERE s.sid = w.sid
AND s.city NOT IN (
    SELECT s2.city
    FROM Student s2, studentSkill ss
    WHERE s2.sid = ss.sid
    AND ss.skill = 'Networks'
)
AND w.salary >= ALL (
    SELECT w2.salary
    FROM worksFor w2
    WHERE w2.cname = w.cname
    AND w2.sid <> w.sid
);


\qecho 'Problem-2c'
--Formulate this query in Pure SQL by only using the set operations INTERSECT,
UNION, and EXCEPT.
SELECT * FROM (
    (
        SELECT s.sname, wf1.salary, s.city
        FROM worksFor wf1, Student s
        WHERE wf1.sid = s.sid
```

```
        EXCEPT

        SELECT s2.sname, wf2.salary, s2.city
        FROM Student s2, worksFor wf2, worksFor wf3
        WHERE wf2.sid = s2.sid
        AND wf2.cname = wf3.cname
        AND wf2.salary < wf3.salary
        AND wf2.sid <> wf3.sid
    )
    EXCEPT

    SELECT s3.sname, wf4.salary, s3.city
    FROM Student s3, worksFor wf4, studentSkill ss
    WHERE s3.sid = wf4.sid
    AND ss.sid = s3.sid
    AND ss.skill = 'Networks'
) AS result
ORDER BY result.sname;



--Find each pair (c1, c2) of cnames of different companies such that
--no employee of c1 and no employee of c2 live in Chicago.

\qecho 'Problem-3a'
--Formulate this query in Pure SQL by only using the
--EXISTS or NOT EXISTS set predicates. You can not use the set operations
INTERSECT, UNION, and EXCEPT.
SELECT c1.cname, c2.cname
FROM Company c1, Company c2
WHERE c1.cname <> c2.cname
AND NOT EXISTS (
    SELECT 1
    FROM worksFor w1, student s1
    WHERE w1.sid = s1.sid
    AND w1.cname = c1.cname
    AND s1.city = 'Chicago'
)
AND NOT EXISTS (
    SELECT 1
    FROM worksFor w2, student s2
    WHERE w2.sid = s2.sid
    AND w2.cname = c2.cname
    AND s2.city = 'Chicago'
);


\qecho 'Problem-3b'
--Formulate this query in Pure SQL by only using the IN, NOT IN, SOME, or ALL set
membership predicates.
--You can not use the set operations INTERSECT, UNION, and EXCEPT.
SELECT c1.cname, c2.cname
FROM Company c1, Company c2
WHERE c1.cname <> c2.cname
AND c1.cname NOT IN (
    SELECT w1.cname
    FROM worksFor w1, student s1
    WHERE w1.sid = s1.sid
    AND s1.city = 'Chicago'
```

```
)
AND c2.cname NOT IN (
    SELECT w2.cname
    FROM worksFor w2, student s2
    WHERE w2.sid = s2.sid
    AND s2.city = 'Chicago'
);


\qecho 'Problem-3c'
--Formulate this query in Pure SQL by only using the set operations INTERSECT,
UNION, and EXCEPT.
SELECT c1.cname, c2.cname
FROM Company c1, Company c2
WHERE c1.cname <> c2.cname

EXCEPT
(
      SELECT c1.cname, c2.cname
      FROM worksFor w1, Student s1, Company c1, Company c2
      WHERE w1.sid = s1.sid
      AND w1.cname = c1.cname
      AND c1.cname <> c2.cname
      AND s1.city = 'Chicago'

      UNION

      SELECT c1.cname, c2.cname
      FROM worksFor w2, Student s2, Company c1, Company c2
      WHERE w2.sid = s2.sid
      AND w2.cname = c2.cname
      AND c1.cname <> c2.cname
      AND s2.city = 'Chicago'
);



--Reconsider Problem 1. Formulate this query in RA SQL
\qecho 'Problem-4b'
SELECT wf1.cname, s.sid, wf1.salary
FROM worksFor wf1
JOIN Student s ON wf1.sid = s.sid
JOIN Knows k ON k.sid1 = s.sid
JOIN studentSkill ss ON k.sid2 = ss.sid
WHERE ss.skill = 'OperatingSystems'

EXCEPT

SELECT wf1.cname, s.sid, wf1.salary
FROM worksFor wf1
JOIN Student s ON wf1.sid = s.sid
JOIN worksFor wf2 ON wf1.cname = wf2.cname
      AND wf2.salary < wf1.salary
      AND wf1.sid <> wf2.sid
ORDER BY cname;


--Reconsider Problem 2. Formulate this query in RA SQL
\qecho 'Problem-5b'
```

```
SELECT * FROM
(
      SELECT s.sname, w1.salary, s.city
      FROM Student s
      JOIN worksFor w1 ON s.sid = w1.sid

      EXCEPT

      SELECT s.sname, w1.salary, s.city
      FROM worksFor w1
      JOIN worksFor w2 ON w1.cname = w2.cname
          AND w1.sid <> w2.sid
          AND w2.salary > w1.salary
      JOIN Student s ON s.sid = w1.sid

      EXCEPT

      SELECT s.sname, w1.salary, s.city
      FROM Student s
      JOIN worksFor w1 ON s.sid = w1.sid
      JOIN Student s2 ON s.city = s2.city
      JOIN studentSkill ss ON ss.sid = s2.sid
      WHERE ss.skill = 'Networks'
) AS result
ORDER BY result.sname;


--Reconsider Problem 3. Formulate this query in RA SQL
\qecho 'Problem-6b'
SELECT C1.cname AS c1, C2.cname AS c2
FROM Company C1
JOIN Company C2 ON C1.cname < C2.cname
WHERE C1.cname NOT IN (
    SELECT W.cname
    FROM worksFor W
    JOIN student S ON W.sid = S.sid
    WHERE S.city = 'Chicago'
)
AND C2.cname NOT IN (
    SELECT W.cname
    FROM worksFor W
    JOIN student S ON W.sid = S.sid
    WHERE S.city = 'Chicago'
)
UNION
SELECT C2.cname AS c1, C1.cname AS c2
FROM Company C1
JOIN Company C2 ON C1.cname < C2.cname
WHERE C1.cname NOT IN (
    SELECT W.cname
    FROM worksFor W
    JOIN student S ON W.sid = S.sid
    WHERE S.city = 'Chicago'
)
AND C2.cname NOT IN (
    SELECT W.cname
    FROM worksFor W
    JOIN student S ON W.sid = S.sid
    WHERE S.city = 'Chicago'
```

```
)
ORDER BY c1, c2;



\qecho 'Problem 10'

CREATE MATERIALIZED VIEW CompanyKnownStudent AS
SELECT DISTINCT W1.sid
FROM worksFor W1
JOIN Knows K ON W1.sid = K.sid2
JOIN worksFor W2 ON K.sid1 = W2.sid
WHERE W1.cname = W2.cname
  AND W1.salary > W2.salary
GROUP BY W1.sid
HAVING COUNT(DISTINCT K.sid1) >= 2;

SELECT * FROM CompanyKnownStudent;



\qecho 'Problem 11'
CREATE OR REPLACE FUNCTION SkillOnlyOneStudent(skill1 TEXT)
RETURNS TABLE(sid1 INTEGER, sid2 INTEGER) AS
$$
SELECT DISTINCT S1.sid AS sid1, S2.sid AS sid2
FROM studentSkill S1
CROSS JOIN student S2
WHERE S1.skill = skill1
  AND S2.sid <> S1.sid
  AND NOT EXISTS (
    SELECT 1
    FROM studentSkill SS
    WHERE SS.sid = S2.sid AND SS.skill = skill1
  )
  AND EXISTS (
    SELECT 1
    FROM studentSkill SS
    WHERE SS.sid = S2.sid
  );
$$
 LANGUAGE SQL;

 SELECT * FROM SkillOnlyOneStudent('WebDevelopment');



\qecho 'Problem 12'
CREATE OR REPLACE VIEW sameGeneration AS
WITH RECURSIVE sameGen(n1, n2) AS (
    -- Base rule: every node is in the same generation as itself
    SELECT parent AS n1, parent AS n2
    FROM PC
    UNION
    SELECT child AS n1, child AS n2
    FROM PC
    UNION
    -- Inductive rule
    SELECT PC1.child AS n1, PC2.child AS n2
```

```
    FROM PC PC1
    JOIN sameGen SG ON PC1.parent = SG.n1
    JOIN PC PC2 ON PC2.parent = SG.n2
    WHERE PC1.child <> PC2.child
)
SELECT DISTINCT * FROM sameGen
ORDER BY n1, n2;

SELECT * FROM sameGeneration;


-- Connect to default database
\c postgres

-- Drop database created for this assignment
DROP DATABASE assignment2;
```