```
DROP TABLE IF EXISTS Person;

DROP TABLE IF EXISTS Knows;

DROP TABLE IF EXISTS Company;

DROP TABLE IF EXISTS WorksFor;

DROP TABLE IF EXISTS JobSkill;

DROP TABLE IF EXISTS PersonSkill;

CREATE TABLE Person(pid integer, name text, city text, birthYear
integer, primary key(pid));

CREATE TABLE Knows(pid1 integer, pid2 integer, primary key(pid1,
pid2), foreign key(pid1) references Person(pid), foreign
key(pid2) references Person(pid));

CREATE TABLE Company(cname text, city text, primary key(cname,
city));

CREATE TABLE WorksFor(pid integer, cname text, salary integer,
primary key(pid), foreign key(pid) references Person(pid));

CREATE TABLE JobSkill(skill text, primary key(skill));

CREATE TABLE PersonSkill(pid integer, skill text, primary
key(pid, skill), foreign key(pid) references Person(pid), foreign
key(skill) references JobSkill(skill));

TABLE Person;

TABLE Knows;

TABLE Company;

TABLE WorksFor;

TABLE JobSkill;

TABLE PersonSkill;

INSERT INTO Person VALUES
(1,'Nick','NewYork',1990),
```

```
 (2,'Deepa','Indianapolis',1985),
 (3,'Eric','NewYork',1990),
 (4,'Ryan','Indianapolis',1995),
 (5,'Hasan','Indianapolis',1990),
 (6,'Arif','Indianapolis',1980),
 (7,'Ryan','Chicago',1980),
 (8,'Jean','SanFransisco',2000),
 (9,'Aya','SanFransisco',1985),
 (10,'Lisa','NewYork',2000),
 (11,'Arif','Chicago',1990),
 (12,'Deepa','Bloomington',1990),
 (13,'Nick','SanFransisco',1980),
 (14,'Ryan','Indianapolis',1990),
 (15,'Nick','Indianapolis',1990),
 (16,'Anna','Chicago',1980),
 (17,'Lisa','Bloomington',1990),
 (18,'Ryan','Bloomington',1995),
 (19,'Lisa','Chicago',1980),
 (20,'Danielle','Indianapolis',1985),
 (21,'Eric','Chicago',1980),
 (22,'Anna','Indianapolis',1985),
 (23,'Chris','Bloomington',1990),
 (24,'Aya','NewYork',1995),
 (25,'Arif','SanFransisco',1990),
 (26,'Anna','Bloomington',2000),
 (27,'Latha','SanFransisco',2000),
 (28,'Eric','Bloomington',2000),
 (29,'Linda','Bloomington',1990),
 (30,'Aya','NewYork',1995),
 (31,'Aya','NewYork',1996),
 (32,'Anna','Bloomington',1985);

INSERT INTO Knows VALUES
(5,22),
 (15,28),
 (10,27),
 (11,27),
 (13,14),
 (11,14),
 (5,28),
 (1,26),
 (18,24),
 (24,5),
 (6,26),
 (15,7),
```

(15,25),
(19,27),
(10,5),
(11,19),
(20,22),
(27,23),
(24,29),
(4,10),
(26,12),
(13,15),
(19,4),
(20,10),
(10,6),
(1,7),
(17,23),
(9,26),
(3,10),
(21,29),
(27,15),
(12,13),
(16,3),
(14,24),
(14,28),
(12,4),
(15,8),
(4,28),
(18,11),
(12,16),
(30,12),
(4,9),
(4,8),
(29,13),
(29,20),
(24,18),
(16,13),
(30,17),
(23,22),
(7,16),
(29,22),
(26,3),
(28,30),
(25,10),
(3,22),
(22,21),
(30,3),

(1,20),
(19,11),
(29,15),
(13,30),
(11,12),
(1,5),
(13,18),
(24,19),
(30,10),
(4,12),
(24,11),
(18,22),
(3,2),
(4,3),
(12,23),
(25,24),
(17,20),
(28,10),
(8,17),
(15,13),
(1,9),
(6,18),
(3,4),
(4,19),
(24,23),
(27,3),
(12,5),
(12,2),
(26,22),
(30,15),
(20,13),
(28,14),
(14,5),
(1,10),
(7,9),
(27,22),
(12,11),
(16,20),
(12,3),
(17,7),
(2,14),
(18,25),
(16,24),
(16,15),
(31,14),

```
 (32,14),
 (32,7),
 (31,7);

INSERT INTO Company VALUES
 ('Amazon','NewYork'),
 ('IBM','NewYork'),
 ('Amazon','Indianapolis'),
 ('Amazon','Bloomington'),
 ('Intel','NewYork'),
 ('Netflix','Indianapolis'),
 ('Yahoo','Indianapolis'),
 ('Google','Bloomington'),
 ('Apple','Indianapolis'),
 ('Hulu','Chicago'),
 ('Hulu','NewYork'),
 ('Yahoo','Chicago'),
 ('Intel','Bloomington'),
 ('Google','Chicago'),
 ('Zoom','Chicago'),
 ('Yahoo','NewYork'),
 ('Yahoo','Bloomington'),
 ('Netflix','Bloomington'),
 ('Microsoft','Chicago'),
 ('Netflix','NewYork'),
 ('Microsoft','Indianapolis'),
 ('Zoom','SanFransisco'),
 ('Netflix','SanFrancisco'),
 ('Yahoo','SanFrancisco'),
 ('IBM','SanFrancisco'),
 ('Uber','Bloomington');

INSERT INTO WorksFor VALUES
 (1,'IBM',60000),
 (2,'Hulu',50000),
 (3,'Amazon',45000),
 (4,'Microsoft',60000),
 (5,'Amazon',40000),
 (6,'IBM',50000),
 (7,'IBM',50000),
 (8,'Netflix',45000),
 (9,'Yahoo',50000),
 (10,'Hulu',40000),
 (11,'Apple',40000),
 (12,'Netflix',55000),
```

```
 (13,'Apple',40000),
 (14,'IBM',50000),
 (15,'IBM',40000),
 (16,'Apple',55000),
 (17,'Google',45000),
 (18,'Amazon',45000),
 (19,'Zoom',45000),
 (20,'Microsoft',55000),
 (21,'Intel',55000),
 (22,'IBM',40000),
 (23,'Apple',40000),
 (24,'Google',45000),
 (25,'Hulu',50000),
 (26,'Intel',55000),
 (27,'Intel',50000),
 (28,'Intel',50000),
 (29,'Google',60000),
 (30,'Intel',60000),
 (31,'Uber',50000),
 (32,'Uber',60000);

INSERT INTO JobSkill VALUES
 ('Programming'),
 ('Databases'),
 ('AI'),
 ('Networks'),
 ('Mathematics'),
 ('Accounting');

insert into personskill values
 (27,'Programming'),
 (18,'Mathematics'),
 (10,'AI'),
 (29,'Networks'),
 (23,'AI'),
 (4,'AI'),
 (1,'Databases'),
 (10,'Networks'),
 (9,'Programming'),
 (13,'Networks'),
 (9,'AI'),
 (27,'Mathematics'),
 (20,'AI'),
 (29,'Databases'),
 (5,'Programming'),
```

```
(26,'Databases'),
(1,'Networks'),
(28,'AI'),
(15,'Programming'),
(16,'Mathematics'),
(12,'Databases'),
(15,'Databases'),
(24,'Programming'),
(14,'AI'),
(25,'Networks'),
(13,'AI'),
(12,'Programming'),
(22,'Programming'),
(7,'Mathematics'),
(10,'Programming'),
(16,'Databases'),
(19,'Programming'),
(7,'Programming'),
(22,'AI'),
(5,'Databases'),
(2,'Mathematics'),
(14,'Programming'),
(26,'Networks'),
(19,'Networks'),
(21,'Programming'),
(14,'Mathematics'),
(19,'AI'),
(2,'Networks'),
(8,'Databases'),
(13,'Mathematics'),
(29,'Programming'),
(3,'AI'),
(16,'Networks'),
(5,'Networks'),
(17,'AI'),
(24,'Databases'),
(2,'Databases'),
(27,'Networks'),
(28,'Databases'),
(30,'Databases'),
(4,'Networks'),
(6,'Networks'),
(17,'Networks'),
(23,'Programming'),
(20,'Programming'),
```

```
 (31,'Programming'),
 (32,'Databases'),
 (32,'Accounting'),
 (6, 'Databases');


TABLE Person;

TABLE Knows;

TABLE Company;

TABLE WorksFor;

TABLE JobSkill;

TABLE PersonSkill;

\qecho 'Problem 1'

-- Find the ID and name of each person who works for IBM and
whose salary is lower
-- than another person who works for IBM as well and has
Programming skill.

-- For Reference
(SELECT p1.pid, p1.name FROM Person p1, WorksFor w1 WHERE p1.pid
= w1.pid AND w1.cname = 'IBM' AND w1.salary <= ALL (SELECT
MIN(s.salary) FROM WorksFor s));

-- Formulate this query in SQL without using subqueries and set
predicates.
-- You are allowed to use the SQL operators INTERSECT, UNION, and
EXCEPT.
-- Problem 1a

SELECT p1.pid, p1.name FROM Person p1, WorksFor w1 WHERE p1.pid =
w1.pid AND w1.cname = 'IBM' AND w1.salary <= 40000;

\qecho 'Problem 1b'

-- Formulate this query in SQL by only using the IN or NOT IN set
predicates.

SELECT DISTINCT p.pid, p.name FROM Person p, WorksFor w1 WHERE
```

```
p.pid IN (SELECT w.pid FROM WorksFor w WHERE p.pid = w.pid AND
w.cname = 'IBM' AND w.salary <= ALL (SELECT MIN(s.salary) FROM
WorksFor s));

\qecho 'Problem 1c'

-- Formulate this query in SQL by only using the SOME or ALL set
predicates.

(SELECT p1.pid, p1.name FROM Person p1, WorksFor w1 WHERE p1.pid
= w1.pid AND w1.cname = 'IBM' AND w1.salary <= ALL (SELECT
MIN(s.salary) FROM WorksFor s));

\qecho 'Problem 1d'

-- Formulate this query in SQL by only using the EXISTS
or NOT EXISTS set predicates.

SELECT p.pid, p.name FROM Person p WHERE EXISTS (SELECT w.pid
FROM WorksFor w WHERE p.pid = w.pid AND w.cname = 'IBM' AND
w.salary <= ALL (SELECT MIN(s.salary) FROM WorksFor s));


\qecho 'Problem 1.2'

-- Find the ID and name of each person who knows another person
who works for ,
-- but who does not know a person who works at and has the skill.

\qecho 'Problem 1.2a'
-- Formulate this query in SQL without using subqueries and set
predicates. You are allowed to use the SQL operators INTERSECT,
UNION, and EXCEPT.

SELECT DISTINCT  p.name, k.pid1 FROM Person p, Person p2, Knows
k, WorksFor w WHERE p.pid = k.pid1 AND p2.pid = k.pid2 AND p2.pid
= w.pid AND w.cname = 'Hulu' INTERSECT SELECT DISTINCT p.name,
k.pid1 FROM Person p, Person p2, Knows k, WorksFor w, JobSkill js
WHERE p.pid = k.pid1 AND p2.pid = k.pid2 AND p2.pid <> w.pid AND
w.cname = 'Intel' AND js.skill = 'Networks' ORDER BY 1, 2;

\qecho 'Problem 1.2b'
-- Formulate this query in SQL by only using the IN or NOT IN set
predicates.
```

```
SELECT DISTINCT p.name, k.pid1 FROM Person p, Person p2, Person
p3, Knows k, WorksFor w, JobSkill j WHERE p.pid = k.pid1 AND
p2.pid = k.pid2 AND p2.pid = w.pid AND w.cname = 'Hulu' AND
k.pid2 NOT IN (SELECT k.pid2 FROM Person p, Person p2, Knows k,
WorksFor w, JobSkill j  WHERE  p.pid = k.pid1 AND p2.pid = k.pid2
AND p2.pid <> w.pid AND w.cname = 'Intel' AND j.skill =
'Networks');

\qecho 'Problem 1.2c'

\qecho 'Problem 1.2d'

\qecho 'Problem 1.3'

\qecho 'Problem 1.3a'

\qecho 'Problem 1.3b'

\qecho 'Problem 1.3c'

\qecho 'Problem 1.3d'

\qecho 'Problem 2'

\qecho 'Problem 2.1a'
-- Define a view SalaryAbove50000 that defines the sub relationof
Person consisting of the employees whose
-- salary is strictly above 50000. Test your view

CREATE VIEW SalaryAbove50000 AS SELECT DISTINCT p.pid, p.name,
p.city, p.birthYear FROM Person p, WorksFor w WHERE p.pid = w.pid
AND w.salary > 50000 ORDER BY p.pid ASC;

SELECT * FROM SalaryAbove50000;

\qecho 'Problem 2.1b'
-- Define  a  view Programmerthat  returnsthe  set  of IDsof
persons  whose  job  skill  is Programming.
-- Test your view.

CREATE VIEW Programmers AS SELECT ps.pid FROM PersonSkill ps
WHERE ps.skill = 'Programming';

SELECT * FROM Programmers;
```

```
\qecho 'Problem 2.1c'

-- Using the views SalaryAbove50000 and Programmer, write the
following query in SQL:
-- Netflix


SELECT DISTINCT s.pid, s.name FROM SalaryAbove50000 s,
SalaryAbove50000 s2, Programmers p,  WorksFor w, Knows k WHERE
s.pid = p.pid
AND p.pid = w.pid AND w.cname = 'Netflix' AND s.pid = k.pid1 AND
s2.pid <> k.pid2;


\qecho 'Problem 2.2a'

-- Define a parameterized view SalaryAbove(amount integer) that
returns, for a given value for the amount parameter, the
subrelation of
-- Person consisting of the employees whose salary is strictly
above that of this value. Test your view for the parameter values
30000,
-- 50000, and 55000.

CREATE OR REPLACE FUNCTION SalaryAbove(amount integer) RETURNS
TABLE (pid integer, name text, city text, birthYear integer) AS
$$
SELECT DISTINCT p.pid, p.name, p.city, p.birthYear FROM Person p,
WorksFor w, WorksFor w2 WHERE p.pid = w.pid AND w.salary <>
amount AND w.salary > amount ORDER BY p.pid ASC;
$$ LANGUAGE SQL;

SELECT * FROM SalaryAbove(30000);
SELECT * FROM SalaryAbove(50000);
SELECT * FROM SalaryAbove(55000);


\qecho 'Problem 2.2b'

-- Define  a  view  KnowsEmployeeAtCompany(cname  text)  that
returns  the  set  of  pids  of persons  who know a  person who
works   -- at the  company given  by the value  of the  parameter
cname.
-- Test you view for the parameters , , and .
```

```
CREATE OR REPLACE FUNCTION KnowsEmployeeAtCompany(cname text)
RETURNS TABLE (pid integer) AS
$$
SELECT DISTINCT p.pid FROM Person p, Person p2, Knows k, Company
c, WorksFor w  WHERE k.pid1 = p.pid AND k.pid2 = p2.pid AND
c.cname = cname AND k.pid1 = w.pid OR k.pid2 = w.pid AND w.cname
= c.cname;
$$ LANGUAGE SQL;

CREATE OR REPLACE FUNCTION KnowsEmployeeAtCompany(compname text)
RETURNS TABLE (pid integer) AS
$$
SELECT DISTINCT p.pid FROM Person p, Person p2, Knows k, Company
c, WorksFor w  WHERE p.pid = k.pid1 AND p2.pid = k.pid2 AND
k.pid2 = w.pid AND w.cname = compname ORDER BY p.pid ASC;
$$ LANGUAGE SQL;

SELECT * FROM KnowsEmployeeAtCompany('Yahoo');
SELECT * FROM KnowsEmployeeAtCompany('Google');
SELECT * FROM KnowsEmployeeAtCompany('Amazon');

\qecho 'Problem 3'
```