B461 Exam 1 Practice Problems with Solutions

The following are a list of practice problems for Exam 1.

1 Preamble

1.1 Pure SQL and General SQL

In this exam, we distinguish between Pure SQL and general SQL. Below we list the features that are allowed in Pure SQL and general SQL, respectively. **Comment**: JOIN operations are **not** allowed in Pure SQL nor general SQL. In other words, JOIN operations can not be used in this exam.

Features allowed in Pure SQL

SELECT ... FROM ... WHERE
UNION, INTERSECT, EXCEPT IN
and NOT IN predicates
ALL and SOME predicates
EXISTS and NOT EXISTS predicates
VIEWs and user-defined FUNCTIONs that can only use the above SQL features

Features allowed in general SQL

---

all the features of Pure SQL
aggregate functions COUNT, SUM, AVERAGE, MIN and MAX
GROUP BY and HAVING clauses
VIEWs and user-defined FUNCTIONs that can use all of the above SQL features

2 Database schema used in this exam

For most problems, we will use the following database schema:

Person(pid, pname, city)
Company(cname,city)
Job Skill(skill)
Works(pid, cname, salary)
Person Skill(pid,skill)

Manages(mid,eid)

In this database, we maintain a set of persons (Person), a set of companies (Company), and a set of job skills (Job_Skill). The city attribute in Person specifies the city in which the person lives. The city attribute in Company indicates a city in which the company is located. (Companies may be located in multiple cities.) A person can be employed by at most one company. (We permit that a person is not employed.) A person can have multiple job skills (Person_Skill). A job skill can be the job skill of multiple persons. (A person may not have any job skills, and a job skill may have no persons with that skill.) A pair (*m,e*) in Manages indicates that *m* is the pid of a person who is a manager of an employee who is a person with pid *e*. We permit that a manager manages multiple employees and that an employee can have multiple managers. (It is possible that an employee has no manager and that an employee is not a manager.) We further require that an employee and his or her managers must work in the same company. The attributes mid and eid refer to the pid of the employee and the manager, respectively.

1. Consider the following Pure SQL query:

   ```
   SELECT DISTINCT w1.cname
   FROM      works w1, works w2
   WHERE w1.cname = w2.cname AND w1.sid <> w2.sid
   ```

   Formulate an equivalent query in Pure SQL that does not use the DISTINCT clause.

   ```
   SELECT w1.cname
   FROM      works w1
   WHERE w1.cname IN (SELECT w2.cname
                FROM  works w2
                WHERE w1.sid <> w2.sid)
   ```

2. Let **R**(A) and **S**(A,B) be two relations. Consider the following query:

   ```
   SELECT r.A
   FROM R r
   WHERE r.A NOT IN (SELECT s1.A
   ```

```
FROM  S s1, S s2
WHERE s1.A <> s2.A AND s1.B = SOME (SELECT s.B
                                   FROM       S s
                                   WHERE      s.B <> s2.B))
```

Formulate this query in Pure SQL without using any of the set predicates (NOT) EXISTS, (NOT) IN, θ ALL, and θ SOME predicates. You can use the set operations UNION, INTERSECT, and EXCEPT.

```
SELECT r.A
FROM      R r
EXCEPT
SELECT s1.A
FROM      S s1, S s2, S s
WHERE s1.A <> s2.A AND s1.B = s.B AND s.B <> s2.B
```

3. Let **R**(A) and **S**(A,B) be two relations. Consider the following query:

```
SELECT r.A
FROM   R r
WHERE EXISTS (SELECT s1.A
FROM   S s1, S s2
WHERE r.A = s2.A
AND s1.A <> s2.A
AND s1.B <> ALL (SELECT s.B
                 FROM S s
                 WHERE          s.A = r.A))
```

Translate this query into an equivalent SQL query wherein the set predicates are simulated with the SQL COUNT aggregate function.

```
SELECT r.A
FROM       R r
WHERE (SELECT COUNT(1)
          FROM     S s1, S s2
           WHERE r.A = s2.A AND s1.A <> s2.A AND
                            (SELECT COUNT(1)
                            FROM    S s
```

WHERE s.A = r.A AND NOT(s1.B <> s.B)) = 0) >= 1

4. Consider the following constraint: *Each manager must have all the skills of each of the employees that he or she manages.*

   Write a Pure SQL boolean query that returns true if this constraint is satisfied in the database, and returns false if this is not the case.

```
select not exists (select m.mid from manages m, job_skill je
        where m.eid = je.pid and je.skill not in(select jm.skill
                                from    job_skill jm
                                where jm.pid = m.mid))
```

5. Formulate the following queries in Pure SQL (in particular, you cannot use SQL aggregate functions in your solution):

   (c) Find the pid and name of each person who lives in the same city as one or more of his or her managers.

```
select p.pid, p.name
from        person p
where p.city in (select pm.city
            from        manages m, person pm
            where m.eid = p.pid
            and m.mid = pm.pid)
```

   Alternatively,

```
select p.pid, p.name from    person p, manages m, person
pm where m.eid = p.pid and m.mid= pm.pid and p.city =
pm.city
```

   (d) Find the mid of each manager who has a higher salary than at least two employees he or she manages.

```
select distinct m.mid from    manages m, works w where m.mid =
w.pid and exists(select 1 from        works w1, works w2, manages
```

```
m1, manages m2 where w1.pid <> w2.pid and w1.pid = m1.eid and
w2.pid = m2.eid and m.mid = m1.mid and m.mid = m2.sid and
                                    w.salary > w1.salary and w.salary > w2.salary)
```

(e) Find the pairs (*c,e*) where *c* is the name of a company and *e* is the pid of a person who works at the company and who has the highest salary of all the employees working for that company.

```
select c.name, w.pid
from          works w
where w.salary >= ALL (select w1.salary
                                    from works w1 where
                                    w1.cname = c.cname)
```

(f) Find each pairs (*c,e*) where *c* is the name of a company and *e* is the pid of an employee who works for that company and who has at least one manager who lacks at least one of that employee's job skills.

```
select w.cname, w.pid from  works w where exists
(select m.mid from   manages m where w.pid = m.eid
and exists (select je.skill from          Person_Skill je
where je.pid = w.pid and je.skill not in (select jm.skill
from          Person_Skill jm where jm.pid = m.mid)))
```

7. Formulate the following queries in general SQL. So now you can use the aggregate functions COUNT, SUM, MIN, MAX, and AVERAGE, and the operations GROUP BY and HAVING. (Again however, you can not use SQL's JOIN operations.) You should also consider creating and using user-defined functions.

(a) Find, for each person, that person's pid and name along with the number of persons he or she manages. (Make sure that your solution works also if a person is not a manager.)

```
create function numberOfEmployeesManagedBy(person
int) returns int as $$ select count(m.eid) from
      manages m where m.mid = person; $$ language sql;

select p.pid, p.pname, numberOfEmployeesManagedBy(p.pid)
```

```
from     person p;
```

(c) Find the name of each city that has the highest number of employed persons.

```
create function numberOfEmployeesWorkingInCity(city
text) returns int as $$ select count(w.pid) from          works
w, company c where w.cname = c.cname and c.city = city;
$$ language sql;

select c.city from
       company c
where   NumberOfEmployeesWorkingInCity(c.city) >=
                              ALL (select NumberofEmployeesWorkingInCity(c1.city)
                                    from company c1);
```

(d) Find the name of each company that employees more managers than non-managers.

```
create function NumberOfManagersAtCompany(company
text) returns int as $$ select count(distinct m.mid) from
       manages m where m.mid in (select w.pid
                          from         works w where
                          w.cname = company);
$$ language sql;

create function
NumberOfNonManagersAtCompany(company text) returns
int as $$ select count(w.pid) from      works w where
w.cname = company and
             w.pid not in (select m.mid from
                                    manages m, works w1
                              where         m.mid = w1.pid
                              and w1.cname = company);
$$ language sql;

select c.cname
from
       company c
where NumberofOfManagersAtCompany(c.cname) >
                    NumberofNonManagersAtCompany(c.cname);
```

8. In the following queries with quantifiers you have to use the method of **Venn diagrams with (non-counting) conditions**. In particular, you need to use views and parameterized views to specify the relevant sets that are involved in this queries.

   Using this method, formulate the following queries in Pure SQL.

   (b) Find the pid of each employee who has all the job skills of at least one of his or her managers.

```
create function
job_skillsOfPerson(person int)
returns table (skill text) as
$$ select skill
  from
      job_skill
  where pid =
  person;
$$ language sql;

select w.pid
from works w where exists (select 1 from manages m
where m.eid = w.pid and not exists (select skill from
job_skillsOfPerson(m.mid) except select skill from
job_skillsOfPerson(w.pid)));
```

   (d) Find each pairs (*c,m*) where *c* is the name of a company that is located in Bloomington and *m* is the mid a manager who works for that company and who manages at least two employees who make more than $50,000.

```
create function
employeesManagedByManager(manager
int) returns table (pid int) as
$$ select eid
from
      manages
where mid =
manager; $$
language sql;
```

```
create function
employeesWhoMakeMoreThan50KAtCompany(
company) returns table(pid int) as
(select pid from works where
 salary > 50K and cname =
 company);

select distinct c.cname, m.mid from  company c, manager m, works w
where c.city = 'Bloomington' and m.mid = w.works and w.cname =
c.cname and exists (select 1 from      person p1, person p2 where p1.pid
<> p2.pid and p1.pid in (select pid from
      EmployeesManagedBy(m.mid) intersect select pid from
      employeesWhoMakeMoreThan50KAtCompany(c.cname))
                                        and
                                              p2.pid in (select pid from
                                            EmployeesManagedBy(m.mid)
                                                 intersect select pid from
                                          employeesWhoMakeMoreThan50K
                                                AtCompany(c.cname))
```

9. In the following queries with quantifiers you have to use the method of **Venn diagrams with counting conditions**. In particular, you need to use views and parameterized views to specify the relevant sets that are involved in this queries and make appropriate use of the SQL COUNT aggregate function.

   Using this method, formulate the following queries in Pure SQL.

   (a) Repeat all the problems in Question 8

   ii. Find the pid of each employee who has all the job skills of at least one of his or her managers.

```
create function job_skillsOfPerson(person int)
returns table (skill text) as
$$
 select skill from     job_skill where pid = person;
$$ language sql;
select w.pid
from          works w
```

```
where exists (select 1
        from
        manages m
        where m.eid = w.pid
        and (select count(1)
            from (select skill
                from job_skillsOfPerson(m.mid)
                except
                select skill
                from job_skillsOfPerson(w.pid)) q) = 0);
```

iv. Find each pairs (*c,m*) where *c* is the name of a company that is located in Bloomington and *m* is the mid a manager who works for that company and who manages at least two employees who make more than $50,000.

```
create function
employeesManagedByManager(manager
int) returns table (pid int) as
$$ select eid
from
        manage
s where mid =
manager; $$
language sql;

create function
employeesWhoMakeMoreThan50KAtCompany
(company) return as
(select pid from works where
 salary > 50K and cname =
 company);

select distinct c.cname, m.mid from company c,
manager m, works w where c.city = 'Bloomington'
and m.mid = w.works and w.cname = c.cname and
   (select count (1) from (select pid from
    EmployeesManagedBy(m.mid) intersect select
    pid from
```

```
        employeesWhoMakeMoreThan50KAtCompan
        y(c.cname))) >= 2
```

(b) Find each pair of persons pids (p_1,p_2) such that p_1 has at least 3 of the job skills of person p_2.

```
create function
job_skillsOfPerson(person int) returns
table (skill text) as
$$ select skill
  from
      job_skill
  where pid =
  person;
$$ language sql;

select p1.pid,
p2.pid from
      person p1,
person p2 where
(select count(1)
from
      (select skill
            from
            job_skillsOfPerson(p1
            .pid) intersect select
            skill
            from job_skillOfPerson(p2.pid)) q) >= 3;
```

(c) Find each pair of persons pids (p_1,p_2) such that p_1 has all-but three of the job skills of person p_2.

```
create function
job_skillsOfPerson(person int) returns
table (skill text) as
$$ select skill
  from
      job_skill
  where pid =
  person;
$$ language sql;
```

```
select p1.pid,
p2.pid from
      person p1,
person p2 where
(select count(1)
from
      (select skill
            from
            job_skillsOfPerson(p2
            .pid) except select
            skill
            from job_skillOfPerson(p1.pid)) q) = 3;
```

10. Let $P(x)$ be a polynomial. For example, $P(x)$ could be the polynomial $3x^3 - 2x^2 + 5$.

    We can represent a polynomial $P(x)$ with a binary relation **P**(coefficient, degree) wherein each pair (c,d) represents the term cx^d in $P(x)$. For example, $P(x) = 3x^3 - 2x^2 + 5$ is represented in **P** as follows:

P

coefficient	degree
3	3
−2	2
5	0

    Write a SQL function

```
CREATE FUNCTION P value(x numeric) RETURNS numeric AS
$$ ...
$$ LANGUAGE SQL; such that, for
```

    an input number x_0,

```
SELECT P value(x 0);
```

    returns the value $P(x_0)$. For example, for the polynomial $P(x) = 3x^3 - 2x^2 + 5$ and $x_0 = 7$, $P(7) = 3 \times 7^3 - 2 \times 7^2 + 5 = 936$ and so

```
SELECT P value(7);
```

should return the value 936.

Of course, your solution should work for any polynomial $P(x)$. Observe that if $P(x) = c_0+c_1x+c_2x^2+\cdots+c_nx^n$, then $P(x_0)$ is an aggregated sum, i.e., $P(x_0) = \sum_{d=0}^{n} c_d(x_0)^d$.

```
create function P_value(x int) returns int as
$$ select sum(p.coefficient * power(x, p.degree)) from
  polynomial p; $$ LANGUAGE SQL;
```