1. SQL

A large software company maintains the following database about its projects and developers

Project(pid,name,startYear)
Developer(did,name,hireYear)
WorksOn(pid,did,year)

Each project has a name and a start year. Each developer has a name and the year when she/he was hired. WorksOn records whether a developer worked on a project in a given year. Note that a developer may work on the same project in multiple years, and a project may have multiple developers in any given year.

(a) Write the sequence of SQL statements necessary to create the tables above. Assume that name is a TEXT type, and all other attributes are integers. Include all keys or foreign keys declarations.

Solution:

```
DROP TABLE IF EXISTS Workson;
DROP TABLE IF EXISTS Project;
DROP TABLE IF EXISTS Developer;

CREATE TABLE Project(pid INT PRIMARY KEY, name TEXT, startYear INT);
CREATE TABLE Developer(did INT PRIMARY KEY, name TEXT, hireYear INT);
CREATE TABLE WorksOn(pid INT REFERENCES Project, did INT REFERENCES Developer, year INT);
```

(b) Write a SQL query that returns for each project and for each year the total number of developers who worked on that project in that year. Your query should return the project's ID, name, year, and number of developers, sorted by year in increasing order, and, within each year, sorted in decreasing order of the number of developers. (Years in which no developers worked need not be included as "0 developers.")

Solution:

```
SELECT X.pid, X.name, Y.year, count(Y.pid) AS cnt
FROM Project AS X, WorksOn AS Y
WHERE X.pid = Y.pid
GROUP BY X.pid, X.name, Y.year
ORDER BY Y.year, cnt desc;
```

(c) A project is called inactive if no developer worked on it since 2010

(including and after 2010). Write a SQL query to find all inactive projects. You should return their IDs and names. (A project is active if a developer worked on it in 2010.)

Solution: Solution 1:

```
SELECT X.pid, X.name
FROM Project AS X
WHERE NOT EXISTS (SELECT *
                  FROM WorksOn AS Y
                   WHERE X.pid = Y.pid and Y.year >= 2010);
```

-1 point for unnecessary joins
-3 points for missing correlation in the subquery X.pid=Y.pid

Solution 2:

```
SELECT X.pid, X.name
FROM Project X left outer join WorksOn Y
      ON X.pid = Y.pid and Y.year >= 2010
GROUP BY X.pid, X.name
HAVING count(Y.pid) = 0;
```

Solution 3 (imperfect because it misses projects without any developers):

```
SELECT X.pid, X.name
FROM Project X, WorksOn Y
WHERE X.pid = Y.pid
GROUP BY X.pid, X.name
HAVING 2010 > max(Y.year);
```

-1 point because incomplete answer

Solution 4

```
SELECT X.pid, X.name
FROM Project X
     WHERE 2010 > ALL (SELECT Y.year from WorksOn Y WHERE Y.pid =
                                                          X.pid);
```

(d) For each year since 1990 (including and after 1990), return the project(s) that the most developers worked on during that year. Return the year and project ID. In case of a tie (if multiple projects had the maximum number of developers) return all of them.

Solution:

```
-- Solution 1: Universal Quantifier Approach with ALL
SELECT W.year, W.pid
FROM WorksOn AS W
WHERE W.year >= 1990
GROUP BY W.year, W.pid
HAVING count(*) >= ALL (SELECT count(*)
                        FROM WorksOn AS W2
                        WHERE W2.year = W.year
                            GROUP BY W.pid);

-- Solution 2: Witness Approach
WITH A AS (SELECT W.year AS year, W.pid AS pid, count(*) AS count
           FROM WorksOn AS W
           WHERE W.year >= 1990
           GROUP BY W.year, W.pid),
     B AS (SELECT A.year AS year, max(A.count) AS count
           FROM A
           GROUP BY A.year)
SELECT A.year, A.pid
FROM A, B
WHERE A.year = B.year AND A.count = B.count;

-- Solution 3: variant on the Witness
WITH A ... -- same as above
SELECT x.year, x.pid
FROM A x, A y
WHERE x.year = y.year
GROUP BY x.year, x.pid, x.count
HAVING x.count = max(y.count);
```

-5 points for answers containing the expression max(count(*)) -4 points for an expression like HAVING x.count = max(x.count) (same x)

(e) 'SystemX' is the oldest project of the company. Write a SQL query that returns all developers who worked every year on 'SystemX', from when it started until 2015. Your query should return the developers' ID and name.

Solution:

```
-- Solution 1
SELECT Z.did, Z.name
FROM Project AS X, WorksOn AS Y, Developer AS Z
WHERE X.pid = Y.pid AND Y.did = Z.did
          AND X.startYear <= Y.year
          AND Y.year <= 2015
          AND X.name = 'SystemX'
GROUP BY Z.did, Z.name, X.startYear
HAVING count(*) = 2015 - X.startYear + 1;

-- Solution 2 (imperfect, but got full credit where present) SELECT
Z.did, Z.name
FROM Developer Z
WHERE NOT EXISTS ( -- find a year where Z didn't work on SystemX
   SELECT *
   FROM WorksOn Y -- proxy for the set of all years (which we don'tWHERE
   NOT EXISTS ( -- check if Z worked that year on SystemX SELECT * FROM
   WorksOn Y2, Project X
       WHERE Y.year = Y2.year and Y2.pid = X.pid and X.name = 'SystemX')
```

solution without any negation got 2-3 points max.

A solution with only 1 negation got 5 points max.

1 point off for unnecessary join

(f) For each question below indicate whether the two SQL queries are equivalent. Assume that the database does not contain any NULL values.

i. Are Q3 and Q4 equivalent?

```
Q3: SELECT DISTINCT Z.did, Z.name
        FROM Project X1, Project X2, Workson Y1, Workson Y2,
      Developer Z WHERE X1.pid = Y1.pid AND Y1.did = Z.did
        AND X2.pid = Y2.pid AND Y2.did = Z.did
        AND X1.startYear < 2010 AND Y1.year > 2018
        AND X2.startYear < 2012 AND Y2.year > 2015;

Q4: SELECT DISTINCT Z.did, Z.name
        FROM Project X, Workson Y, Developer Z
      WHERE X.pid = y.pid AND y.did = Z.did
        AND X.startYear < 2010 AND y.year > 2018;
```

Yes/No: Yes

ii. (1 point) Are Q5 and Q6 equivalent?

Q5:
```
SELECT X.did, count(*)
    FROM WorksOn X
GROUP BY X.did;
```

Q6:
```
SELECT X.did, (SELECT count(*)
                FROM WorksOn Y
               WHERE X.did = Y.did)
    FROM WorksOn X;
```

Yes/No: No

iv. Are Q7 and Q8 equivalent?

Q7:
```
SELECT X.did, count(*)
    FROM WorksOn X
WHERE X.year > 2010
GROUP BY X.did;
```

Q8:
```
SELECT X.did, (SELECT count(*)
                FROM WorksOn y
          WHERE X.did = y.did AND y.year > 2010)
    FROM WorksOn X;
```

Yes/No: No

v. Are Q11 and Q12 equivalent?

Q11:
```
SELECT X.name, Z.name
    FROM Developer X
        LEFT OUTER JOIN
    WorksOn Y ON X.did = Y.did
        LEFT OUTER JOIN
    Project Z ON Y.pid = Z.pid;
```

Q12:
```
SELECT X.name, Z.name
    FROM Developer X
        LEFT OUTER JOIN
    WorksOn Y on X.did = Y.did
        JOIN
    Project Z on Y.pid = Z.pid;
```

Yes/No: Yes

(g) Consider the following database instance:

Project

pid	name	startYea
10 20	Syste mX Syste mY	NULL 2016

WorksOn

pid	did	year
10 20 20	1 1 2	NULL 2015 NULL

Developer

did	nam e	startYear
1 2	Alice Bob	2015 NULL

Indicate for each query below what answers it returns; write ”empty” if the answer is the emptyset.

i. (1 point) What does query Q2 return?:

```
SELECT X.name AS proj, Z.name AS dev
  FROM Project X, Workson Y, Developer Z
 WHERE X.pid = Y.pid AND Y.did = Z.did
       AND (Z.hireyear = 2015 AND (Y.year = 2015 OR X.startyear = 2016));
```

Solution:

proj	dev
SystemY	Alice

ii. (1 point) What does query Q3 return?:

```
SELECT X.name AS proj, Z.name AS dev
  FROM Project X, Workson Y, Developer Z
 WHERE X.pid = Y.pid AND Y.did = Z.did
       AND (Z.hireyear = 2015 OR (Y.year = 2015 AND NOT(X.startyear = 2016)));
```

Solution:

proj	dev
SystemX	Alice
SystemY	Alice

iii. What does query Q4 return?:

```
SELECT X.name AS proj, Z.name AS dev
  FROM Project X, Workson Y, Developer Z
 WHERE X.pid = Y.pid AND Y.did = Z.did
          AND (NOT(Z.hireyear = 2015) OR (Y.year = 2015 AND X.startyear = 2016));
```

Solution:

proj	dev
SystemY	Alice

Q. i. Which of the following is the most accurate English interpretation of the SQL query below?

```
select distinct z.city
from Customer z
where not exists
          (select *
           from Orders y
           where y.cid = z.cid and y.qnt < 100);
```

Returns cities that . . .

(A) had at least one order for < 100 units of a product.

(B) have a customer who placed at least one order for more < 100 units of a product.

(C) had only orders with ≥ 100 units.

(D) have a customer that placed only orders with ≥ 100 units.

i. D

2. Relational Algebra

Consider the same relational schema as before, including the key/foreign key constraints:

Project(pid,name,startYear)
Developer(did,name,hireYear)
WorksOn(pid,did,year)

(a) (5 points) Write a Relational Algebra expression in the form of a logical query plan (i.e., draw a tree) that is equivalent to the SQL query below. Your query plan does not have to be necessarily “optimal”: however, points will be taken off for overly
complex solutions.

```
SELECT X.did, X.name, count(*)
FROM Developer X, WorksOn y, Project Z
WHERE X.did = y.did and y.pid = Z.pid
  AND y.year < Z.startYear + 2
GROUP BY X.did, X.name, X.hireYear
HAVING X.hireYear + 10 < max(startYear);
```

Solution:

$$\pi_{(X.did,X.name,C)}($$
$$\sigma_{(X.hireYear+10<M)}($$
$$\gamma_{(X.did,X.name,X.hireYear,count(*)->C,max(Z.startYear)->M)}($$
$$\sigma_{(Y.year<Z.startYear+2)}($$
$$\rho_X(Developer) \bowtie_{(X.did=Y.did)} \rho_Y(WorkOn) \bowtie_{(Y.pid=Z.pid)} \rho_Z(Project)))))$$

(b)

i. Write a Relational Algebra expression in the form of a logical query plan (i.e., draw a tree) that is equivalent to the SQL query in the previous question (reproduced below). Your query plan does not have to be necessarily "optimal": however, points will be taken off for overly complex solutions.

```
SELECT X.did
FROM Developer X
WHERE NOT EXISTS
          (SELECT *
           FROM Project Z
           WHERE NOT EXISTS
                   (SELECT *
                    FROM WorksOn Y
                    WHERE X.did = Y.did AND Y.pid = Z.pid
                      AND Y.year = 2015));
```

Solution:

$$\pi_{(X3.did)}(\rho_{X3}(Developer)) - \pi_{(X.did)}(A - B)$$

Where A is

$$\pi_{(Y2.pid,Y2.did)}($$
$$(\rho_{Y2}(WorksOn))$$

Where B is

$$\pi_{(Z.pid,X.did)}($$
$$\sigma_{(Y.year=2015)}($$
$$\rho_X(Developer) \bowtie_{(X.did=Y.did)} \rho_Y(WorksOn) \bowtie_{(Y.pid=z.pid)} \rho_Z(Project)))$$

(c) Assume the database schema in Question 1 ; assume the key and foreign keys defined there, and that the database instance does not contain NULL's. Answer the questions below, assuming all expressions have set semantics.

i. (2 points) The notation |*S*| means the cardinality of a set *S* (number of tuples in *S*). Does the following always hold?

$$|\texttt{Developer} \bowtie_{\texttt{did=did}} \sigma_{\texttt{year}<2015}(\texttt{WorksOn})| \leq |\texttt{Developer}|$$

i. No

Yes/No:

ii. (2 points) Does the following always hold?

$$|\texttt{Developer} \bowtie_{\texttt{did=did}} \sigma_{\texttt{year}<2015}(\texttt{WorksOn})| \leq |\texttt{WorksOn}|$$

ii. Yes

Yes/No: