CS143: Database Systems
Homework #2

1. Assume the following tables for this problem:

```
Employee(person-name, age, street, city)
Work(person-name, company-name, salary)
Company(company-name, city)
Manage(person-name, manager-name)
```

A person's name is unique, but a person may work for more than one company. A company name is unique, but a company may be located in more than one city.

(a) Write a query in SQL to find the names of persons who work in one or more companies where their salary < 22000

ANSWER:

```
SELECT person-name    FROM  Work
WHERE  Work.salary <22000
```

(b) Write the same query in Relational Algebra.

ANSWER:

$\Pi_{person-name}(\sigma_{salary<22000}(Work))$

(c) Compare the results of (a) and (b), are they the same? Why?

ANSWER:

```
No, because there might be duplicates in the result of the SQL query.
```

2. Assume the database of the previous problem and write the following queries in SQL. You should use at least one subquery in each of your answers and write each query in two significantly different ways (e.g., using different operators such as `EXISTS`, `IN`, and `ALL`)

(a) Find the name(s) of the employee(s) whose *total* salary is higher than those of all employees living in Los Angeles.

ANSWER:

```
SELECT person-name FROM Work
GROUP BY person-name
HAVING SUM(salary)>ALL
(SELECT SUM(salary) FROM Work, Employee
WHERE Work.person-name=Employee.person-name AND city='Los Angeles'
GROUP BY Work.person-name)
```

```
SELECT person-name FROM Employee E
WHERE NOT EXISTS
(SELECT Work.person-name FROM Work, Employee
WHERE Work.person-name=Employee.person-name AND city='Los Angeles'
GROUP BY Work.person-name
HAVING SUM(salary)>=
(SELECT SUM(salary) FROM Work W
WHERE W.person-name=E.person-name))
```

(b) Find the name(s) of the manager(s) whose *total* salary is higher than that of at least one employee that they manage.

ANSWER:

```
SELECT manager-name
FROM Manage M,
(SELECT person-name, SUM(salary) total-salary
FROM Work GROUP BY person-name) S1
WHERE M.manager-name=S1.person-name AND
S1.total-salary > SOME (SELECT total-salary
FROM (SELECT person-name, SUM(salary) total-salary
FROM Work GROUP BY person-name) S2
WHERE S2.person-name = M.person-name)

SELECT manager-name
FROM Manage M
WHERE EXISTS (SELECT *
FROM (SELECT person-name, SUM(salary) total-salary
FROM Work GROUP BY person-name) S1,
(SELECT person-name, SUM(salary) total-salary
FROM Work GROUP BY person-name) S2
WHERE M.manager-name=S1.person-name AND
M.person-name = S2.person-name AND
S1.total-salary > S2.total-salary)
```

3. Assume the following tables for this problem:

```
MovieStar(name, address, gender)
MovieExec(name, address, company, netWorth)
```

(a) We want to find the names and addresses of all female movie stars (`gender = 'F'` in the `MovieStar` relation) who are also movie executives with a net worth over $2,000,000 (`netWorth > 2000000` in the `MovieExec` relation).

i. Write the query using `INTERSECT` operator.
ANSWER:

```
SELECT name, address FROM MovieStar WHERE gender='F' INTERSECT SELECT name, address
FROM MovieExec WHERE netWorth>1000000
```

ii. Write the query without using INTERSECT operator.
ANSWER:

```
SELECT name, address FROM MovieStar WHERE gender=’F’ AND (name, address) in (SELECT
name, address FROM MovieExec WHERE netWorth>1000000)
```

(b) We want to find the movie stars who are not movie executives.

i. Write the query using EXCEPT operator.
ANSWER:

```
SELECT name FROM MovieStar EXCEPT SELECT name FROM MovieExec
```

ii. Write the query without using EXCEPT operator.
ANSWER:

```
SELECT name FROM MovieStar WHERE name not in (SELECT name FROM MovieExec)
```

4. Assume the following tables for this problem:

```
ComputerProduct(manufacturer, model, price)
Desktop(model, speed, ram, hdd)
Laptop(model, speed, ram, hdd, weight)
```

A computer product is either a desktop or a laptop.

(a) Find the average speed of all desktop computers.
ANSWER:

```
SELECT AVG(speed) FROM Desktop
```

(b) Find the average price of all laptops with weight below 2kg.
ANSWER:

```
SELECT AVG(price) FROM ComputerProduct CP, Laptop L WHERE CP.model=L.model AND weight<=2
```

(c) Find the average price of PC’s and laptops made by “Dell.”
ANSWER:

```
SELECT AVG(price) FROM ComputerProduct WHERE manufacturer=‘DELL’
```

(d) For each different CPU speed, find the average price of a laptop.
ANSWER:

```
SELECT AVG(price) FROM Laptop GROUP BY speed
```

(e) Find the manufacturers that make at least three different computer models.
ANSWER:

```
SELECT manufacturer FROM ComputerProduct GROUP BY manufacturer HAVING COUNT(model)>=3
```

5. Assume the computer-product database of the previous problem, and write the following database modifications.

(a) Using two `INSERT` statements, insert a desktop computer manufactured by HP, with model number 1200, price $1000, speed 1.2Ghz, 256MB RAM, and an 80GB hard drive.
ANSWER:

```
INSERT INTO ComputerProduct VALUES (‘HP’, 1200, 1000); INSERT INTO Desktop VALUES
(1200, ‘1.2GHz’, ‘256MB’, ‘80GB’);
```

(b) Using two `DELETE` statements, delete all desktops manufactured by IBM with price below $1000. (*Comments: Be careful with the order of your two* `DELETE` *statements.*)
ANSWER:

```
DELETE FROM Desktop WHERE model IN (SELECT model FROM ComputerProduct WHERE manufacturer
= ‘IBM’ AND price < 1000); DELETE FROM ComputerProduct WHERE manufacturer=‘IBM’ AND
price<1000 AND model NOT IN (SELECT model FROM Laptop);
```

(c) For each laptop made by Gateway, add one kilogram to the weight. (*Hint: The* `WHERE` *clause in a* `UPDATE` *statement may contain complex conditions, including subqueries.*)
ANSWER:

```
UPDATE Laptop SET weight=weight+1 WHERE model IN (SELECT model FROM ComputerProduct
WHERE manufacturer=‘Gateway’);
```

6. Returning to the Enroll(sid,dept,cnum,sec) example where **enroll** show t
he enrollment for this quarter:

(a) Write an SQL query to find the students who are only enrolled in the CS classes offered this quarter.

```
SELECT   E1.sid
FROM Enroll AS E1
WHERE  E1.sid  NOT IN   (SELECT  Sid  FROM  Enroll  WHERE dept<> 'CS')
```

(b) Write and SQL query to find the students who are enrolled in all the CS classes offered this quarter.

```
(SELECT   E0.sid  /* an enrolled student who is not missing any CS class */
FROM Enroll AS E0
WHERE  E0.sid  NOT IN       /* a E1.sid is a student  who is missing some   CS class */
       (SELECT   E1.sid   FROM Enroll AS E1
                    WHERE  (E1.Stid, E1.dept, E1.cnum)
                       NOT IN   (SELECT E1.Stid,  'CS', E2.cnum
                                   FROM  Enroll AS  E2 WHERE E2.dept='CS'))
```

(c) Write the previous queries using different SQL constructs. In particular can you express those queries using the count aggregate? Please explain

```
SELECT   E1.sid
FROM Enroll AS E1
WHERE E1.dept ='CS'
GROUP BY E1   HAVING  count(dept,cnum)=
                    (SELECT  count(dept, cnum)
                     FROM  Enroll  WHERE dept= 'CS')
```