

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 they make a salary that is less than \$22,000.

ANSWER:

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

(b) Write an SQL query to find the names of persons who work in one or more companies and make less than \$22,000 in the majority (i.e., 50% or more) of the companies they work for.

ANSWER:

```
SELECT Total.person-name
FROM (SELECT person-name, count(*) as cnt FROM Work GROUP BY person-name) as Total,
(SELECT person-name, count(*) as cnt FROM Work WHERE Work.salary<22000 GROUP BY person-name) as Less
WHERE Total.person-name = Less.person-name and Less.cnt >= 0.5 * Total.cnt
```

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 Barstow.

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='Barstow'
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='Barstow'
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>2000000

```

- 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>2000000)
```

- (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 which shows the 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 an 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.sid
HAVING count(*) = (SELECT count(*) FROM Enroll WHERE dept= 'CS')
```

7. Write SQL queries for Hw1 problems 3 (b), (c), (d), (e) using different constructs from those you used for HW1 (e.g., if you used NOT EXIST or NOT IN, you might use COUNT or ALL, or viceversa.)

Customer (customer-name, street, city)

Branch (branch-name, city)

Account (customer-name, branch-name, account-number)

(b) Find the names of all customers who have an account in a branch NOT located in the same city that they live in.

```
SELECT C.customer_name FROM Customer C
WHERE EXISTS (SELECT *
              FROM Account A, Branch B
              WHERE C.customer_name = A.customer_name
                  AND B.branch_name = A.branch_name AND C.city != B.city)
```

(c) Find the branches that do not have any accounts.

```
SELECT branch_name
FROM (SELECT B.branch_name, count(account_number) as numAccounts
      FROM Branch B LEFT JOIN Account A ON A.branch_name = B.branch_name
      Group By B.branch_name) as BranchAccounts
WHERE BranchAccounts.numAccounts = 0
```

(d) Find the customer names who do not have any account in the 'Region12' branch.

```
SELECT R12_AccountCounts.customer_name FROM
(SELECT A.customer_name, count(*) AS numAccounts
 FROM Account A LEFT JOIN
      (SELECT * FROM Account WHERE branch_name = "Region12") AS R12_Accounts
      ON A.customer_name = R12_Accounts.customer_name
 GROUP BY A.customer_name) AS R12_AccountCounts
WHERE R12_AccountCounts.numAccounts = 0
```

(e) Find the customer names who have accounts in all the branches located in 'Los Angeles'. You are not allowed to use the division operator directly for this question.

```
SELECT customer_name
FROM (SELECT customer_name, count(*) AS numAccounts
      FROM (SELECT A.customer_name, A.branch_name
            FROM Account A, Branch B
            WHERE A.branch_name = B.branch_name and city = "Los Angeles"
            /* group by to merge cases when one has multiple accounts in a branch */
            GROUP BY A.customer_name, A.branch_name) AS T
      GROUP BY customer_name) AS LA
WHERE LA.numAccounts =
      (SELECT count(*) FROM Branch WHERE city="Los Angeles")
```