# DBA 372: Database Management Systems

## Assignment 2

**1) Consider the SQL query whose answer is shown in Figure 1.**

**1. Modify this query so that only the *login* column is included in the answer.**

**2. If the clause WHERE *S.gpa >*= *2* is added to the original query, what is the set of tuples in the answer?**



Figure 1

**Answer:** The answers are as follows:

1. Only *login* is included in the answer:

SELECT S.login

FROM Students S

WHERE S.age *<* 18

2. The answer tuple for Madayan is omitted then.

**2) Consider the following relations:**

**Student(*snum:* integer, *sname:* string, *major:* string, *level:* string, *age:* integer)**

**Class(*name:* string, *meets at:* string, *room:* string, *fid:* integer)**

**Enrolled(*snum:* integer, *cname:* string)**

**Faculty(*fid*: integer, *fname:* string, *deptid:* integer)**

**The meaning of these relations is straightforward; for example, Enrolled has one record per student-class pair such that the student is enrolled in the class.**

**Write the following queries in SQL. No duplicates should be printed in any of the answers.**

**1. Find the names of all Juniors (level = JR) who are enrolled in a class taught by I. Teach.**

**2. Find the age of the oldest student who is either a History major or enrolled in a course taught by I. Teach.**

**3. Find the names of all classes that either meet in room R128 or have five or more students enrolled.**

**4. Find the names of all students who are enrolled in two classes that meet at the same time.**

**5. Find the names of faculty members for whom the combined enrollment of the courses that they teach is less than five.**

**6. For each level, print the level and the average age of students for that level.**

**7. For all levels except JR, print the level and the average age of students for that level.**

**8. For each faculty member that has taught classes only in room R128, print the faculty member’s name and the total number of classes she or he has taught.**

**9. Find the names of students not enrolled in any class.**

**Answer:** The answers are given below:

1. SELECT DISTINCT S.Sname

 FROM Student S, Class C, Enrolled E, Faculty F

 WHERE S.snum = E.snum AND E.cname = C.name AND C.fid = F.fid AND

 F.fname = ‘I.Teach’ AND S.level = ‘JR’;

2. SELECT MAX(S.age)

 FROM Student S

 WHERE (S.major = ‘History’)

 OR S.snum IN (SELECT E.snum

 FROM Class C, Enrolled E, Faculty F

 WHERE E.cname = C.name AND C.fid = F.fid

 AND F.fname = ‘I.Teach’ );

3. SELECT C.name

 FROM Class C

 WHERE C.room = ‘R128’

 OR C.name IN (SELECT E.cname

 FROM Enrolled E

 GROUP BY E.cname

 HAVING COUNT (\*) *>*= 5);

4. SELECT DISTINCT S.sname

 FROM Student S

 WHERE S.snum IN (SELECT E1.snum

 FROM Enrolled E1, Enrolled E2, Class C1, Class C2

 WHERE E1.snum = E2.snum AND E1.cname *<>* E2.cname

 AND E1.cname = C1.name

 AND E2.cname = C2.name AND C1.meets at = C2.meets at);

5. SELECT DISTINCT F.fname

 FROM Faculty F

 WHERE 5 *>* (SELECT COUNT (E.snum)

 FROM Class C, Enrolled E

 WHERE C.name = E.cname

 AND C.fid = F.fid);

6. SELECT S.level, AVG(S.age)

 FROM Student S

 GROUP BY S.level;

7. SELECT S.level, AVG(S.age)

 FROM Student S

 WHERE S.level *<>* ‘JR’

 GROUP BY S.level;

8. SELECT F.fname, COUNT(\*) AS CourseCount

 FROM Faculty F, Class C

 WHERE F.fid = C.fid

 GROUP BY F.fid, F.fname

 HAVING ( C.room = ‘R128’ );

9. SELECT DISTINCT S.sname

 FROM Student S

 WHERE S.snum NOT IN (SELECT E.snum

 FROM Enrolled E);

**3) The following relations keep track of airline flight information:**

**Flights(*flno:* integer, *from:* string, *to:* string, *distance:* integer,**

***departs:* time, *arrives:* time, *price:* real)**

**Aircraft(*aid:* integer, *aname:* string, *cruisingrange:* integer)**

**Certified(*eid:* integer, *aid:* integer)**

**Employees(*eid:* integer, *ename:* string, *salary:* integer)**

**Note that the Employees relation describes pilots and other kinds of employees as well; every pilot is certified for some aircraft, and only pilots are certified to fly. Write each of the following queries in SQL.**

**1. Find the names of aircraft such that all pilots certified to operate them have salaries more than $80,000.**

**2. For each pilot who is certified for more than three aircraft, find the *eid* and the maximum *cruisingrange* of the aircraft for which she or he is certified.**

**3. Find the names of pilots whose *salary* is less than the price of the cheapest route from Los Angeles to Honolulu.**

**4. For all aircraft with *cruisingrange* over 1000 miles, find the name of the aircraft and the average salary of all pilots certified for this aircraft.**

**5. Find the names of pilots certified for some Boeing aircraft.**

**6. Find the *aid*s of all aircraft that can be used on routes from Los Angeles to Chicago.**

**7. Identify the routes that can be piloted by every pilot who makes more than $100,000.**

**8. Print the *ename*s of pilots who can operate planes with *cruisingrange* greater than 3000 miles but are not certified on any Boeing aircraft**

**9. Compute the difference between the average salary of a pilot and the average salary of all employees (including pilots).**

**10. Print the name and salary of every nonpilot whose salary is more than the average salary for pilots.**

**11. Print the names of employees who are certified only on aircrafts with cruising range longer than 1000 miles.**

**12. Print the names of employees who are certified only on aircrafts with cruising range longer than 1000 miles, but on at least two such aircrafts.**

**13. Print the names of employees who are certified only on aircrafts with cruising range longer than 1000 miles and who are certified on some Boeing aircraft.**

**Answer:** The answers are given below:

1. SELECT DISTINCT A.aname

 FROM Aircraft A

 WHERE A.Aid IN (SELECT C.aid

 FROM Certified C, Employees E

 WHERE C.eid = E.eid AND

 NOT EXISTS ( SELECT \*

 FROM Employees E1;

 WHERE E1.eid = E.eid AND E1.salary *<* 80000 ))

2. SELECT C.eid, MAX (A.cruisingrange)

 FROM Certified C, Aircraft A

 WHERE C.aid = A.aid

 GROUP BY C.eid

 HAVING COUNT (\*) *>* 3;

3. SELECT DISTINCT E.ename

 FROM Employees E

 WHERE E.salary *<* ( SELECT MIN (F.price)

 FROM Flights F

 WHERE F.from = ‘Los Angeles’ AND F.to = ‘Honolulu’ );

4. Observe that *aid* is the key for Aircraft, but the question asks for aircraft names;

we deal with this complication by using an intermediate relation Temp:

 SELECT Temp.name, Temp.AvgSalary

 FROM ( SELECT A.aid, A.aname AS name, AVG (E.salary) AS AvgSalary

 FROM Aircraft A, Certified C, Employees E

 WHERE A.aid = C.aid AND C.eid = E.eid AND A.cruisingrange *>* 1000

 GROUP BY A.aid, A.aname ) AS Temp;

5. SELECT DISTINCT E.ename

 FROM Employees E, Certified C, Aircraft A

 WHERE E.eid = C.eid AND C.aid = A.aid AND A.aname = ‘Boeing%’;

6. SELECT A.aid

 FROM Aircraft A

 WHERE A.cruisingrange *>* ( SELECT MIN (F.distance)

 FROM Flights F

 WHERE F.from = ‘Los Angeles’ AND F.to = ‘Chicago’ );

7. SELECT DISTINCT F.from, F.to

 FROM Flights F

 WHERE NOT EXISTS ( SELECT \*

 FROM Employees E

 WHERE E.salary *>* 100000

 AND NOT EXISTS (SELECT \*

 FROM Aircraft A, Certified C

 WHERE A.cruisingrange *>* F.distance

 AND E.eid = C.eid

 AND A.aid = C.aid) );

8. SELECT DISTINCT E.ename

 FROM Employees E

 WHERE E.eid IN ( ( SELECT C.eid

 FROM Certified C

 WHERE EXISTS ( SELECT A.aid

 FROM Aircraft A

 WHERE A.aid = C.aid

 AND A.cruisingrange *>* 3000 )

 AND NOT EXISTS ( SELECT A1.aid

 FROM Aircraft A1

 WHERE A1.aid = C.aid

 AND A1.aname = ‘Boeing%’ ));

9. SELECT Temp1.avg - Temp2.avg

 FROM (SELECT AVG (E.salary) AS avg

 FROM Employees E

 WHERE E.eid IN (SELECT DISTINCT C.eid

 FROM Certified C )) AS Temp1,

 (SELECT AVG (E1.salary) AS avg

 FROM Employees E1 ) AS Temp2;

10. SELECT E.ename, E.salary

 FROM Employees E

 WHERE E.eid NOT IN ( SELECT DISTINCT C.eid

 FROM Certified C )

 AND E.salary *>* ( SELECT AVG (E1.salary)

 FROM Employees E1

 WHERE E1.eid IN ( SELECT DISTINCT C1.eid

 FROM Certified C1 ) );

11. SELECT E.ename

 FROM Employees E, Certified C, Aircraft A

 WHERE C.aid = A.aid AND E.eid = C.eid

 GROUP BY E.eid, E.ename

 HAVING (A.cruisingrange *>* 1000);

12. SELECT E.ename

 FROM Employees E, Certified C, Aircraft A

 WHERE C.aid = A.aid AND E.eid = C.eid

 GROUP BY E.eid, E.ename

 HAVING (A.cruisingrange *>* 1000) AND COUNT (\*) *>* 1;

13. SELECT E.ename

 FROM Employees E, Certified C, Aircraft A

 WHERE C.aid = A.aid AND E.eid = C.eid

 GROUP BY E.eid, E.ename

 HAVING (A.cruisingrange *>* 1000) AND ANY (A.aname = ’Boeing’);

**4)** **Consider the instance of the Sailors relation shown in Figure 5.1.**

**1. Write SQL queries to compute the average rating, using AVG; the sum of the ratings, using SUM; and the number of ratings, using COUNT.**

**2. If you divide the sum just computed by the count, would the result be the same as the average? How would your answer change if these steps were carried out with respect to the *age* field instead of *rating*?**

**3. Consider the following query: *Find the names of sailors with a higher rating than all sailors with age <* 21*.* The following two SQL queries attempt to obtain theanswer to this question. Do they both compute the result? If not, explain why.Under what conditions would they compute the same result?**

 **SELECT S.sname**

 **FROM Sailors S**

 **WHERE NOT EXISTS ( SELECT \***

 **FROM Sailors S2**

 **WHERE S2.age *<* 21**

 **AND S.rating *<*= S2.rating )**

 **SELECT \***

 **FROM Sailors S**

 **WHERE S.rating *>* ( SELECT S2.rating**

 **FROM Sailors S2**

 **WHERE S2.age *<* 21 )**

**4. Consider the instance of Sailors shown in Figure 2. Let us define instance S1 of**

**Sailors to consist of the first two tuples, instance S2 to be the last two tuples, and**

**S to be the given instance.**

**(a) Show the left outer join of S with itself, with the join condition being *sid=sid*.**

**(b) Show the right outer join of S with itself, with the join condition being**

***sid=sid*.**

**(c) Show the full outer join of S with itself, with the join condition being *sid=sid*.**

**(d) Show the left outer join of S1 with S2, with the join condition being *sid=sid*.**

**(e) Show the right outer join of S1 with S2, with the join condition being *sid=sid*.**

**(f) Show the full outer join of S1 with S2, with the join condition being *sid=sid*.**

****

Figure

**Answer:** The answers are shown below:

1. SELECT AVG (S.rating) AS AVERAGE

 FROM Sailors S

 SELECT SUM (S.rating)

 FROM Sailors S

 SELECT COUNT (S.rating)

 FROM Sailors S

2. The result using SUM and COUNT would be smaller than the result using AVERAGE if there are tuples with rating = NULL. This is because all the aggregate operators, except for COUNT, ignore NULL values. So the first approach would compute the average over all tuples while the second approach would compute the average over all tuples with non-NULL rating values. However, if the aggregation is done on the age field, the answers using both approaches would be the same since the age field does not take NULL values.

3. Only the first query is correct. The second query returns the names of sailors with a higher rating than *at least one* sailor with age *<* 21. Note that the answer to the second query does not necessarily contain the answer to the first query. In particular, if all the sailors are at least 21 years old, the second query will return an empty set while the first query will return all the sailors. This is because the NOT EXISTS predicate in the first query will evaluate to *true* if its subquery evaluates

4.



to an empty set, while the ANY predicate in the second query will evaluate to *false* if its subquery evaluates to an empty set. The two queries give the same results if and only if one of the following two conditions hold:

* The *Sailors* relation is empty, or
* There is at least one sailor with age *>* 21 in the *Sailors* relation, and for every sailor s, either s has a higher rating than all sailors under 21 or s has a rating no higher than all sailors under 21.