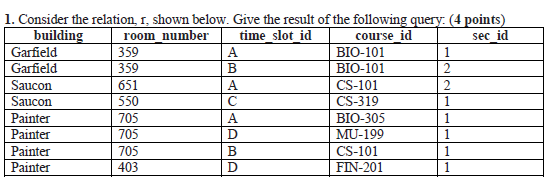
**ERIC WEBB**

**Assignment#2**

**MSIT 630 Database Systems (Summer, 2019)**

**Total: 50 points**

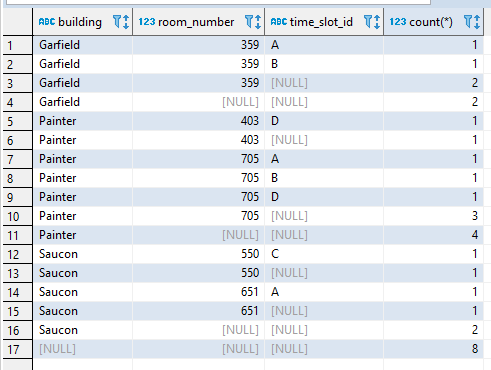
***Due: 6/16/2019 11:59PM***

****

**SELECT building, room\_number, time\_slot\_id, count(\*)**

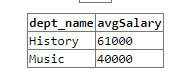
**FROM r**

**GROUP BY building, room\_number, time\_slot\_id with rollup;**

****

**2. In the university database, write a query to find departments whose instructors earn a lower salary, on average, than the average salary at “Biology”. Use user defined SQL functions (create function command) as appropriate to answer the above query, the function takes the department name as the input and returns the average salary of the given department. (6 points)**

**SELECT dept\_name,AVG(salary) as avgSalary FROM instructor group by dept\_name having avgSalary < (select AVG(salary)from instructor where dept\_name = "Biology");**

****

**use** school;

**drop** **function** deptAVGsalary;

**create** **function** deptAVGsalary(dept\_name **varchar**(20))

**returns** **integer**

**DETERMINISTIC**

**begin**

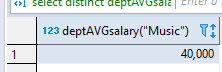
**declare** avgSalary **integer**;

**select** **distinct** **AVG**(salary) **into** avgSalary **FROM** instructor **where** instructor.dept\_name = dept\_name **group** **by** instructor.dept\_name;

**return** avgSalary;

**end**

**select** **distinct** deptAVGsalary("Music") **from** instructor;



**3. Write the following queries in relational algebra, using the university schema. (Appendix A, page 1271) (16 points, 4 points each)**

**a. Find the names of all students who have taken at least one Elec. Eng. course.**

∏name(student ⟗ takes ⟗ ∏ course\_id(σdept\_name=’Elec. Eng.’(course)))

**b. Find the IDs and names of all students who have not taken any course offering before 2010.**

∏ID,name(student) - ∏ID,name (σyear <2010(student ⟗ takes))

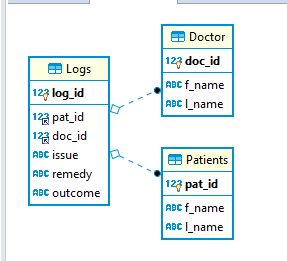
**c. For each department, find the average salary of instructors in that department. You may assume that every department has at least one instructor.**

dept\_nameGavg(salary)(instructor)

**d. Find the lowest, across all departments, of the per-department average salary computed by the preceding query.**

Gmin(avgsal)(dept\_nameGavg(salary)asavgsal(instructor))

**4. Construct an E-R diagram for a hospital with a set of patients and a set of medical doctors. Associate with each patient a log of the various tests and examinations conducted.(6 points)**

****

**Doctor.doc\_id and Patients.pat\_id are both foreign keys to Logs.**

**Logs.log\_id is Primary key to Logs.**

**5. Explain the distinction between disjoint and overlapping constraints. Provide an example for each constraint. (3 points)**

**In disjoint constraints the parent super type can only be one of its sub types.**

**While in Overlapping constraints it can be a number of its sub types.**

**For example, for Disjoint let’s say we have an “ATM Card” this “ATM Card” has two sub-types “Credit Card” or “Debit Card”, because the “ATM Card” can only be only one. (Either a credit card or debit card.) This is what would make it a disjoint constraint.**

**For an Overlapping constraint, let’s say your “ATM Card” has BOTH the functionalities of a Debit Card and a Credit card attached to it. (It is one card but can be both a credit and debit card at will.)Then this is would make it an Overlapping constraint.**

**6. Explain the distinction between total and partial constraints. Provide an example for each constraint. (3 points)**

**A total constraint is when one entity is expected to have at least one relation to another entity. For example, in a school database the “Grades” table will have a “Student” and a “Class”. Without “Students” and a “Class” there will be no grade.**

**For a partial constraint, an entity may not need a relation to another entity. An example of this would be an after school program for students. Not all students would be required to have a relation with the after school program table.**

**7. Consider the following set F of functional dependencies on the relation schema r(A,B,C,D,E,F): (12 points, 4 points each.)**

**A🡪BCD**

**BC🡪DE**

**B 🡪D**

**D🡪A**

1. **Compute B+.**

**Step 1) B+ = Closure of B.**

**Step 2) We find B 🡪 D so that B+ = (BD).**

**Step 3) From (BD) we find D 🡪 A, So that B+ = (BDA).**

**Step 4) From (BDA) we find that A 🡪 BCD, So that B+ = (BDAC).**

**Step 5) From (BDAC) we find that BC 🡪DE so that B+ = (BDACE).**

**No other Functional Dependencies for closure of B, so B+ is (BDACE).**

1. **Compute D+.**

**Step 1) D+ = Closure of D.**

**Step 2) We find D 🡪 A so that D+ = (DA).**

**Step 3) From (DA) we find A 🡪 BCD, So that D+ = (DABC).**

**Step 4) From (DABC) we find that BC 🡪 DE AND B🡪 D, So that D+ = (DABCE).**

**No other Functional Dependencies for closure of D, so D+ is (DABCE).**

1. **Prove (using Armstrong’s axioms) that AF is a super key.**

**Step 1) AF+ = Closure of AF.**

**Step 2) We find A 🡪 BCD so that AF+ = (AFBCD). (Axiom of augmentation)**

**Step 3) From (AFBCD) we find BC🡪 DE AND B🡪D AND D🡪A, So that AF + = (AFBCDE). (Axiom of reflexivity)**

**Step 4) From (DABC) we find that BC 🡪 DE AND B🡪 D, So that AF + = (AFBCDE). (Axiom of transitivity)**

**No other Functional Dependencies for closure of AF, so AF+ is (AFBCDE).**

**Since the closure of AF has all the attributes of the given relation along with the Candidate keys hence AF is the Super key.**