university

November 9, 2018

1 Exercises

```
In [1]: %load_ext sql
        %load_ext autoreload
        %autoreload 2
```

```
In [2]: %sql use SUni;
```

```
Done.
```

```
Out[2]: []
```

1.1 Database schema

```
In [3]: %%sql
        select name + ' (' + (
            select STRING_AGG(COLUMN_NAME,', ')
            from INFORMATION_SCHEMA.COLUMNS
            where table_name = name) + ') ' as [Relation Schema]
        from sys.tables;
```

```
 * mssql+pyodbc://sa:***@E7450/AP?driver=ODBC Driver 17 For SQL Server
Done.
```

```
Out[3]: [('classroom (building, capacity, room_number) ',),
         ('department (budget, building, dept_name) ',),
         ('course (course_id, credits, dept_name, title) ',),
         ('instructor (dept_name, ID, name, salary) ',),
         ('section (building, course_id, room_number, sec_id, semester, time_slot_id, year) ',),
         ('teaches (course_id, ID, sec_id, semester, year) ',),
         ('student (dept_name, ID, name, tot_cred) ',),
         ('takes (course_id, grade, ID, sec_id, semester, year) ',),
         ('advisor (i_ID, s_ID) ',),
         ('time_slot (day, end_hr, end_min, start_hr, start_min, time_slot_id) ',),
         ('prereq (course_id, prereq_id) ',),
         ('sysdiagrams (definition, diagram_id, name, principal_id, version) ',)]
```

1.2 Ex. 1

1.2.1 a. Find the titles of courses in the Comp. Sci. department that have 3 credits.

$\Pi_{title}(\sigma_{dept\_name='Comp.Sci.' \wedge\ credits=3}(course))$

```
In [4]: %%sql
        select distinct title
        from course
        where dept_name = 'Comp. Sci.' and credits = 3;
```

```
 * mssql+pyodbc://sa:***@E7450/AP?driver=ODBC Driver 17 For SQL Server
Done.
```

```
Out[4]: [('Database System Concepts',), ('Image Processing',), ('Robotics',)]
```

1.2.2 b. Find the IDs of all students who were taught by an instructor named Einstein; make sure there are no duplicates in the result.

$\Pi_{s\_ID}(\sigma_{name='Einstein'}(instructor) \bowtie (teaches \bowtie \rho_{takes2[s\_ID,\_,\_,,,]}(takes)))$

```
In [5]: %%sql
        select distinct tk.ID as Student_ID
        from instructor i join (teaches t join takes tk
                                on t.course_id = tk.course_id
                                and t.sec_id = tk.sec_id
                                and t.semester = tk.semester
                                and t.year = tk.year)
                            on i.ID = t.ID
        where i.name = 'Einstein';
```

```
 * mssql+pyodbc://sa:***@E7450/AP?driver=ODBC Driver 17 For SQL Server
Done.
```

```
Out[5]: [('44553',)]
```

1.2.3 c. Find the highest salary of all instructors, (not using aggregate functions)

$\Pi_{salary}(instructor) - \Pi_{salary}(instructor \bowtie_{salary<d.salary} \rho_{d[d.ID,d.name,d.dept\_name,d.salary]}(instructor))$

```
In [6]: %%sql
        select distinct salary from instructor
        except
        select distinct i.salary
        from instructor i join instructor d
            on i.salary < d.salary;
```

```
 * mssql+pyodbc://sa:***@E7450/AP?driver=ODBC Driver 17 For SQL Server
Done.
```

Out[6]: [(Decimal('95000.00'),)]

```
In [7]: %%sql
        select distinct salary
        from instructor
        where salary >= all(
            select salary
            from instructor
        );
```

```
 * mssql+pyodbc://sa:***@E7450/AP?driver=ODBC Driver 17 For SQL Server
Done.
```

Out[7]: [(Decimal('95000.00'),)]

```
In [8]: %%sql
        select distinct i.salary
        from instructor i
        where not exists(
                select *
                from instructor i2
                where i2.salary > i.salary
        );
```

```
 * mssql+pyodbc://sa:***@E7450/AP?driver=ODBC Driver 17 For SQL Server
Done.
```

Out[8]: [(Decimal('95000.00'),)]

$\mathcal{G}_{max(salary)as\ salary}(instructor)$

```
In [9]: %sql select max(salary) [Max Salary] from instructor;
```

```
 * mssql+pyodbc://sa:***@E7450/AP?driver=ODBC Driver 17 For SQL Server
Done.
```

Out[9]: [(Decimal('95000.00'),)]

1.2.4 d. Find all instructors earning the highest salary (there may be more than one with the same salary)

$\Pi_{ID,name}(instructor) - \Pi_{ID,name}(instructor \bowtie_{salary<d.salary} \rho_{d[d.ID,d.name,d.dept\_name,d.salary]}(instructor))$

```
In [10]: %%sql
         select distinct ID, name from instructor
         except
         select distinct i.ID, i.name
         from instructor i join instructor d
             on i.salary < d.salary;
```

```
 * mssql+pyodbc://sa:***@E7450/AP?driver=ODBC Driver 17 For SQL Server
Done.
```

```
Out[10]: [('22222', 'Einstein')]
```

```
In [11]: %%sql
         select distinct ID, name from instructor
         where ID not in (
             select distinct i.ID
             from instructor i join instructor d on
                 i.salary < d.salary);
```

```
 * mssql+pyodbc://sa:***@E7450/AP?driver=ODBC Driver 17 For SQL Server
Done.
```

```
Out[11]: [('22222', 'Einstein')]
```

```
In [12]: %%sql
         select distinct ID, name
         from instructor
         where salary >= all (
             select i.salary
             from instructor i);
```

```
 * mssql+pyodbc://sa:***@E7450/AP?driver=ODBC Driver 17 For SQL Server
Done.
```

```
Out[12]: [('22222', 'Einstein')]
```

```
In [13]: %%sql
         select distinct ID, name
         from instructor i
         where not exists (
             select * from instructor j
             where j.salary > i.salary);
```

```
 * mssql+pyodbc://sa:***@E7450/AP?driver=ODBC Driver 17 For SQL Server
Done.
```

```
Out[13]: [('22222', 'Einstein')]
```

$\Pi_{ID,name}(instructor \bowtie \mathcal{G}_{max(salary)as\ salary}(instructor))$

```
In [14]: %%sql
         select distinct i.ID, i.name
         from instructor i join
             (select max(salary) as salary from instructor) g
             on i.salary = g.salary;
```

```
 * mssql+pyodbc://sa:***@E7450/AP?driver=ODBC Driver 17 For SQL Server
Done.
```

```
Out[14]: [('22222', 'Einstein')]
```

```
In [15]: %%sql
         select distinct i.ID, i.name
         from instructor i
         where i.salary = (select max(j.salary) from instructor j);
```

```
 * mssql+pyodbc://sa:***@E7450/AP?driver=ODBC Driver 17 For SQL Server
Done.
```

```
Out[15]: [('22222', 'Einstein')]
```

1.2.5 e. Find the enrollment of each section that was offered in Autumn 2009

$_{course\_id,sec\_id}\mathcal{G}_{count(ID)as\ enrollment}(\sigma_{semester='Autumn' \wedge year=2009}(takes))$

```
In [16]: %%sql
         select course_id, sec_id, count(ID) as enrollment from takes
         where semester = 'Fall' and year = 2009
         group by course_id, sec_id;
```

```
 * mssql+pyodbc://sa:***@E7450/AP?driver=ODBC Driver 17 For SQL Server
Done.
```

```
Out[16]: [('CS-101', '1', 6), ('CS-347', '1', 2), ('PHY-101', '1', 1)]
```

1.2.6 f. Find the maximum enrollment, across all sections, in Autumn 2009.

$\mathcal{G}_{max(enrollment)as\ enrollment}(_{course_id,sec\_id}\mathcal{G}_{count(ID)as\ enrollment}(\sigma_{semester='Autumn' \wedge year=2009}(takes)))$

```
In [17]: %%sql
         select max(c.enrollment) as enrollment from
             (select course_id, sec_id, count(ID) as enrollment from takes
             where semester = 'Fall' and year = 2009
             group by course_id, sec_id) c;
```

```
 * mssql+pyodbc://sa:***@E7450/AP?driver=ODBC Driver 17 For SQL Server
Done.
```

```
Out[17]: [(6,)]
```

1.2.7 g. Find the sections that had the maximum enrollment in Autumn 2009.

$\Pi_{course\_id,sec\_id}((_{course\_id,sec\_id}\mathcal{G}_{count(ID)as\ enrollment}(\sigma_{semester='Autumn'\wedge year=2009}(takes))) \bowtie (\mathcal{G}_{max(enrollment)as\ enrollment}(_{course_id,sec\_id}\mathcal{G}_{count(ID)as\ enrollment}(\sigma_{semester='Autumn'\wedge year=2009}(takes)))))$

```
In [18]: %%sql
         select r.course_id, r.sec_id
         from (select course_id, sec_id, count(ID) as enrollment from takes
             where semester = 'Fall' and year = 2009
             group by course_id, sec_id) r join
             (select max(t.enrollment) as enrollment from
                 (select course_id, sec_id, count(ID) as enrollment from takes
                 where semester = 'Fall' and year = 2009
                 group by course_id, sec_id) t) s
                 on r.enrollment = s.enrollment;

 * mssql+pyodbc://sa:***@E7450/AP?driver=ODBC Driver 17 For SQL Server
Done.


Out[18]: [('CS-101', '1')]
```

1.3 Database schema

```
In [19]: %%sql
         select name + ' (' + (
             select STRING_AGG(COLUMN_NAME,', ')
             from INFORMATION_SCHEMA.COLUMNS
             where table_name = name) + ') ' as [Relation Schema]
         from sys.tables;

 * mssql+pyodbc://sa:***@E7450/AP?driver=ODBC Driver 17 For SQL Server
Done.


Out[19]: [('classroom (building, capacity, room_number) ',),
          ('department (budget, building, dept_name) ',),
          ('course (course_id, credits, dept_name, title) ',),
          ('instructor (dept_name, ID, name, salary) ',),
          ('section (building, course_id, room_number, sec_id, semester, time_slot_id, year) '
          ('teaches (course_id, ID, sec_id, semester, year) ',),
          ('student (dept_name, ID, name, tot_cred) ',),
          ('takes (course_id, grade, ID, sec_id, semester, year) ',),
          ('advisor (i_ID, s_ID) ',),
          ('time_slot (day, end_hr, end_min, start_hr, start_min, time_slot_id) ',),
          ('prereq (course_id, prereq_id) ',),
          ('sysdiagrams (definition, diagram_id, name, principal_id, version) ',)]
```

1.4 Ex. 2

1.4.1 a. Find the names of all students who have taken at least one Comp. Sci. course.

$\Pi_{name}(student \bowtie takes \bowtie \Pi_{course\_id}(\sigma_{dept\_name='Comp.Sci.'}(course)))$

```
In [20]: %%sql
         select distinct s.name
         from (student s join takes t on
             s.ID = t.ID) join course c on
                 t.course_id = c.course_id
         where c.dept_name = 'Comp. Sci.';

 * mssql+pyodbc://sa:***@E7450/AP?driver=ODBC Driver 17 For SQL Server
Done.


Out[20]: [('Bourikas',), ('Brown',), ('Levy',), ('Shankar',), ('Williams',), ('Zhang',)]
```

1.4.2 b. Find the IDs and names of all students who have not taken any course offering before Spring 2009.

$\Pi_{ID,name}(student \bowtie (\Pi_{ID}(student) - \Pi_{ID}(\sigma_{year<=2008}(takes))))$

```
In [21]: %%sql
         select distinct s.ID, s.name
         from student s
         where s.ID not in (
                 select distinct ID
                 from takes
                 where year <= 2008);

 * mssql+pyodbc://sa:***@E7450/AP?driver=ODBC Driver 17 For SQL Server
Done.


Out[21]: [('00128', 'Zhang'),
          ('12345', 'Shankar'),
          ('19991', 'Brandt'),
          ('23121', 'Chavez'),
          ('44553', 'Peltier'),
          ('45678', 'Levy'),
          ('54321', 'Williams'),
          ('55739', 'Sanchez'),
          ('70557', 'Snow'),
          ('76543', 'Brown'),
          ('76653', 'Aoi'),
          ('98765', 'Bourikas'),
          ('98988', 'Tanaka')]
```

```
In [22]: %%sql
         select distinct s.ID, s.name
         from student s
         where not exists(
                 select * from takes t
                 where t.ID = s.ID and year <= 2008);
```

```
 * mssql+pyodbc://sa:***@E7450/AP?driver=ODBC Driver 17 For SQL Server
Done.
```

```
Out[22]: [('00128', 'Zhang'),
          ('12345', 'Shankar'),
          ('19991', 'Brandt'),
          ('23121', 'Chavez'),
          ('44553', 'Peltier'),
          ('45678', 'Levy'),
          ('54321', 'Williams'),
          ('55739', 'Sanchez'),
          ('70557', 'Snow'),
          ('76543', 'Brown'),
          ('76653', 'Aoi'),
          ('98765', 'Bourikas'),
          ('98988', 'Tanaka')]
```

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

$_{dept\_name}\mathcal{G}_{max(salary)}(instructor)$

```
In [23]: %%sql
         select dept_name, max(salary) as max_salary
         from instructor
         group by dept_name;
```

```
 * mssql+pyodbc://sa:***@E7450/AP?driver=ODBC Driver 17 For SQL Server
Done.
```

```
Out[23]: [('Biology', Decimal('72000.00')),
          ('Comp. Sci.', Decimal('92000.00')),
          ('Elec. Eng.', Decimal('80000.00')),
          ('Finance', Decimal('90000.00')),
          ('History', Decimal('62000.00')),
          ('Music', Decimal('40000.00')),
          ('Physics', Decimal('95000.00'))]
```

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

$\mathcal{G}_{min(max\_dept\_sal)}(_{dept\_name}\mathcal{G}_{max(salary)as\ max\_dept\_sal}(instructor))$

```
In [24]: %%sql
         select min(r.max_salary) min_dept_max_sal
         from (select i.dept_name, max(i.salary) as max_salary
              from instructor i
              group by i.dept_name) r;

 * mssql+pyodbc://sa:***@E7450/AP?driver=ODBC Driver 17 For SQL Server
Done.


Out[24]: [(Decimal('40000.00'),)]
```

1.5 Database schema

```
In [25]: %%sql
         select name + ' (' + (
             select STRING_AGG(COLUMN_NAME,', ')
             from INFORMATION_SCHEMA.COLUMNS
             where table_name = name) + ') ' as [Relation Schema]
         from sys.tables;

 * mssql+pyodbc://sa:***@E7450/AP?driver=ODBC Driver 17 For SQL Server
Done.


Out[25]: [('classroom (building, capacity, room_number) ',),
          ('department (budget, building, dept_name) ',),
          ('course (course_id, credits, dept_name, title) ',),
          ('instructor (dept_name, ID, name, salary) ',),
          ('section (building, course_id, room_number, sec_id, semester, time_slot_id, year) '
          ('teaches (course_id, ID, sec_id, semester, year) ',),
          ('student (dept_name, ID, name, tot_cred) ',),
          ('takes (course_id, grade, ID, sec_id, semester, year) ',),
          ('advisor (i_ID, s_ID) ',),
          ('time_slot (day, end_hr, end_min, start_hr, start_min, time_slot_id) ',),
          ('prereq (course_id, prereq_id) ',),
          ('sysdiagrams (definition, diagram_id, name, principal_id, version) ',)]
```

1.6 Ex 3. Write relational-algebra queries to find the course sections taught by more than one instructor in the following ways:

1.6.1 a. Using an aggregate function.

$\Pi_{course\_id,sec\_id,semester,year}(\sigma_{ins\_cnt>1}(_{course\_id,sec\_id,semester,year}\mathcal{G}_{count(ID)as\ ins\_cnt}(teaches)))$

```
In [26]: %%sql
         select course_id, sec_id, semester, year
         from teaches
         group by course_id, sec_id, semester, year
         having count(ID) > 1;
```

```
 * mssql+pyodbc://sa:***@E7450/AP?driver=ODBC Driver 17 For SQL Server
0 rows affected.
```

```
Out[26]: []
```

1.6.2 b. Without using any aggregate functions.

using natual join

$$\pi_{course\_id,grade,sec\_id,semester,year}(\sigma_{ID<>ID2}(teaches \bowtie \rho_{[ID2,course\_id,sec\_id,semester,year]}(teaches)))$$

using theta join

$$\pi_{course\_id,grade,sec\_id,semester,year}(\quad teaches \bowtie_{ID<>ID2 \wedge course\_id=course\_id2 \wedge sec\_id=sec\_id2 \wedge semester=semester2 \wedge year=year2} \rho_{[ID2,course\_id2,sec\_id2,semester2,year2]}(teaches))$$

```
In [27]: %%sql
         select distinct t.course_id, t.sec_id, t.semester, t.year
         from teaches t join teaches t2
             on t.course_id = t2.course_id
                 and t.sec_id = t2.sec_id
                 and t.semester = t2.semester
                 and t.year = t2.year
                 and t.ID != t2.ID;
```

```
 * mssql+pyodbc://sa:***@E7450/AP?driver=ODBC Driver 17 For SQL Server
0 rows affected.
```

```
Out[27]: []
```