29/36 Study +2 Bump +1

Mary

Denison CS-181/DA-210 Quiz 3

April 14, 2021

Instructions

- Quiz is to be completed synchronously and within a total of 50 minutes.
- **No** electronic resources, including, but not limited to Notebooks from this or any prior semester or class, online help or cheatsheets, web pages, stack overflow or tutorials, or **any execution environment** (cloud or local).
- You **are** permitted a **handwritten** 4x6 index card two-sided, with whatever notes you choose to include.
- If in class, you must hand write your answers on the provided hard copy of the quiz.
- If you are remote, you should be in the class Zoom and in a breakout room with your screen shared **for the duration of the quiz**.
- If you are remote, you will be given both a PDF and a text file containing Markdown for the test. You can do one of the following:
  - Use a PDF editor to add text blocks to annotate the PDF with your answers, submitting the annotated PDF to Notebowl, or
  - Print the PDF, and then hand write your answers on the hard copy. Submit by scanning or taking pictures of the quiz pages and uploading to Notebowl, or
  - Use **only** a text editor and the Markdown version of the quiz, and type your answers at the appropriate place after each of the questions, and submit by uploading the (saved) markdown file. In this last case, you are permitted to also display/read the PDF version of the quiz.
- Each student should **upload to Notebowl** evidence (i.e. a picture) showing their extra credit study group. This must be done by noon on the day of the quiz.

Q1: XPath Operations

10 1/2

3 parts by 4 pts apiece (12 points total)

You are to give me an XPath expression that solves the given query problem, where the underlying structure is based on the `school.xml` and `school_small.xml`, and an abridged version of the structure is given in the last pages of this quiz.

Your solution should **work in general**, as long as the same tree structure is employed, and should not only work for the abridged data given.

1-A: Write an XPath expression to extract all subject ids. Note that, within the `departments` subtree, a department that offers only one subject (major) does not have `subject` children.

2 1/2 /4

You made this more complicated than I intended. The note was to help you not worry about getting values for ones.

```
# Write answer here
'/school/departments/department[contains 'subject) subject/text() else @id]'
```

if we would to do this, would have an '|' and build good XPaths for the 2 cases

inside XPath

```
"""//department/subject/@id"""
```

1-B: Use a single XPath expression to query for the last names of all instructors for whom city is **not** 'Granville'.

```
# Write answer here
'/school/instructors/instructor[city != 'Granville']/last/text()'
```

4/4

1-C: Write an XPath query to make a list of the titles of all courses whose subject is CS and whose course number is less than 300

```
# Write answer here
'/school/courses/course[@subject =='CS' and num < 300]/title/text()'
```

4/4

Q2: XML Procedural Operations

8 points

This question will continue to use the `schools.xml` structure. Suppose we want a table that has columns:

```
dept_id , dept_name , subject_count
```

So we want a table that has a row per department, telling us the id and name of the department and how many subjects (i.e. majors) offered by that department. For a department that only offers one major/subject, there are no subject children of the department node. Note CINE in the example.

Write a function

```
dept_table(dept_root)
```

where the `dept_root` is the lxml Element associated with `departments`, and whose children are the individual department nodes. You are welcome to represent the rows as either dictionaries or as lists.

```
# Write answer here
```

5½/8

```
def dept_table(dept_root)
    header = [dept_id, dept_name, subject_count]
    L = []
    columns = '/school/departments/*'
    for item in columns
        row = []
        row.append(item.attrib('id'))
        row.append(item.text('name'))
        try:
            row.append(count(subjects))
        L.append(row)
    table = pd.DataFrame(L, columns=header)
    return table
```

Would need an xpath, but dept_root already has the root of the departments subtree, which we can then iterate over.

item.find('name').text

a global count function? Soo

```
def dept_table(dept_root):
    LoL = []
    for dept in dept_root:
        row = []
        row.append(dept.get('id'))
        row.append(dept.findtext('name'))
        nsubj = len(dept.findall('subject'))
        row.append(1 if nsubj == 0 else nsubj)
        LoL.append(row)
    return LoL
```

Q3: SQL Operations 13 | 16

These questions all involve the `school` database, whose schema is reproduced at the end of this quiz writeup.

4 parts at 4 points apiece.

3-A: Write an SQL query to obtain the instructor name (a single string composed of `lastname`, a comma and a space, and `firstname`), and city, for all instructors who live in Granville or whose last name ends in `"son"`. Present your result alphabetically by your generated `name` column.

```
# Write answer here
SELECT instructorlast || ', ' || instructorfirst AS name, instructorcity AS city
FROM instructors
WHERE instructorcity = 'Granville' OR instructorlast LIKE '%son'
ORDER BY name ASC
```

3 | 4

OR LIKE '%son'

```
SELECT instructorlast || ', ' || instructorfirst AS name, instructorcity
FROM instructors
WHERE instructorcity = 'Granville' OR instructorlast LIKE '%son'
ORDER BY name
```

3-B Write an SQL query to obtain a list of students whose majors are in the MATH or EDUC **departments**, including their last and first names, their major, and the id of the department. Your predicate should demonstrate the use of an `IN` clause. Results should be in alphabetical order of student last name and then student first name.

```
# Write answer here
SELECT studentlast, studentfirst, studentmajor, departmentid
FROM students JOIN INNER
    (SELECT * FROM subjects JOIN INNER departments
     USING(departmentid))
    ON students.studentmajor = subject.subjectname)
WHERE studentmajor IN ('MATH', 'EDUC')
ORDER BY studentlast ASC, studentfirst ASC
```

3 1/2 | 4

departmentid

No harm, but no bonus pt, unless I asked for department name or division

```
SELECT studentlast, studentfirst, studentmajor, departmentid
FROM students AS ST INNER JOIN subjects AS SU
                    ON (ST.studentmajor = SU.subjectid)
WHERE departmentid IN ('MATH', 'EDUC')
ORDER BY studentlast, studentfirst
```

3-C Write an SQL query to obtain the class id, course subject, course number, and count of students enrolled in the SPRING term. Only include rows where the count is less than 10. The table should be listed in descending enrollment counts.

2 1/2 / 4

```
# Write answer here
SELECT classid, coursesubject, coursenum, COUNT(classid) AS count
FROM classes LEFT JOIN student_class
    USING (classid)
WHERE COUNT(classid) < 10
ORDER BY count DESC
```

GROUP BY → HAVING a

Spring?

needs GROUP BY o/w only 1 row

```
SELECT classid, coursesubject AS subject, coursenum AS num, COUNT(*) AS count
FROM classes C INNER JOIN student_class C USING (classid)
                    INNER JOIN courses CO USING (coursesubject, coursenum)
WHERE classterm = 'SPRING'
GROUP BY classid, subject, num
HAVING count < 10
ORDER BY count DESC
```

1-D: Student's Choice. Either answer the short answer question **or** the SQL query. **Do not do both.**

Short Answer: Explain why, with specific detail, we never do a JOIN without either an ON or a USING clause.

SQL Query: Assume that the `students` table has a `gpa` field. Write a query to list the students (first and last name) whose gpa is above the average gpa.

4 / 4

```
# Write answer here
```

You cannot JOIN without an ON or USING because that will cause a cartesian product. This is where you multiply the tables together instead of adding them, which causes your data to get out of order and give you way more rows than you need. When working with extremely large datasets and tables, this can cause your code to take a really long time to run and it is inefficient.