COMS W4111: Introduction to Databases
Spring 2024, Sections 002/V02

Homework 2: Common

Introduction

This notebook contains HW2 Common. **Students on both tracks should complete this part.** To ensure everything runs as expected, work on this notebook in Jupyter.

Submission instructions:

- You will submit **a PDF** for this assignment
  - The most reliable way to save as PDF is to go to your browser's menu bar and click `File -> Print`. **Switch the orientation to landscape mode**, and hit save.
  - **MAKE SURE ALL YOUR WORK (CODE AND SCREENSHOTS) IS VISIBLE ON THE PDF. YOU WILL NOT GET CREDIT IF ANYTHING IS CUT OFF.** Reach out for troubleshooting.

Written Questions

W1

Explain Codd's 3rd Rule.

- What are some interpretations of a NULL value?
- An alternative to using NULL is some other value for indicating missing data, e.g., using -1 for the value of a weight column. Explain the benefits of NULL relative to other approaches.

NULL can be interpreted as one the following – data is missing, data is not known, or data is not applicable.

Without NULL, we have to write explicitly "weight != -1" for each query, which will cause us lots of trouble. And this will vary from table to table.

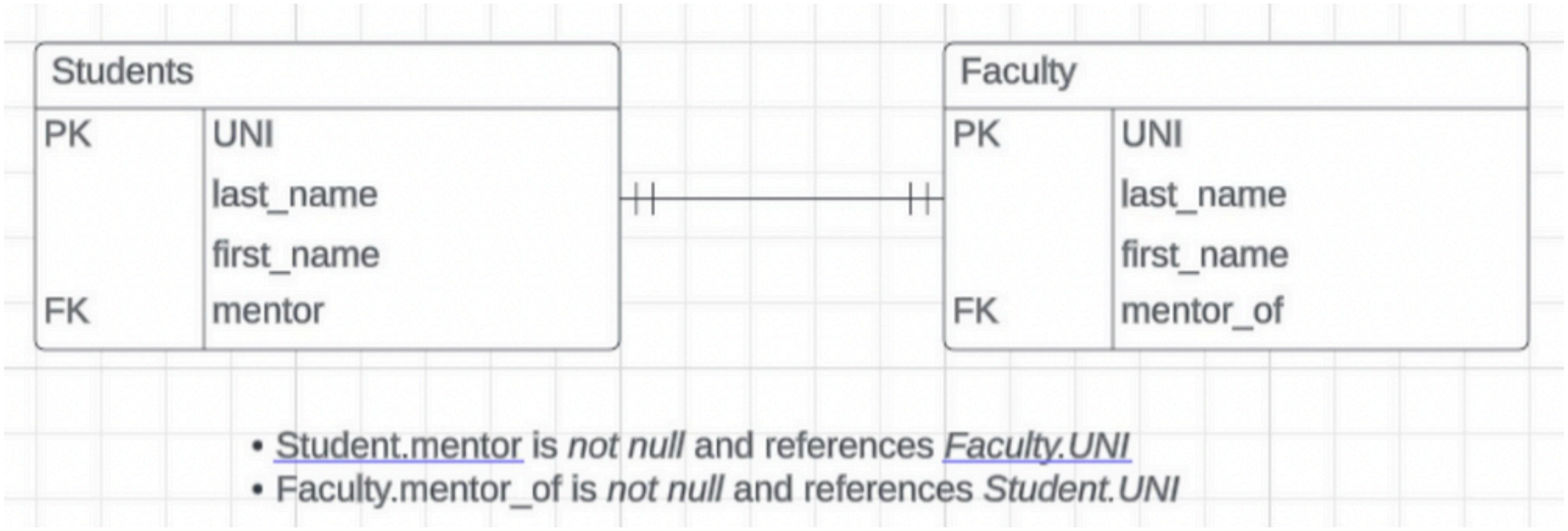
W2

Briefly explain the following concepts:

1. Primary key
2. Candidate key
3. Super key
4. Alternate key
5. Composite key
6. Unique key
7. Foreign key

1. Primary Key: uniquely identifies each row in that table.
2. Candidate Key: can uniquely identify each row in a table and from which one is chosen as the primary key.
3. Super Key: can uniquely identify rows in a table, which may include additional columns not necessary for uniqueness.
4. Alternate Key: can serve as a primary key but is not chosen as the primary key, essentially a candidate key that is not the primary key.
5. Composite Key: made up of two or more columns used together as a primary key to uniquely identify rows in a table.
6. Unique Key: a constraint that ensures all values in a column or a combination of columns are unique across all rows in the table, allowing null values.
7. Foreign Key: refers to the primary key columns in another table, establishing a link between the two tables.

W3



- Student.mentor is *not null* and references *Faculty.UNI*
- Faculty.mentor_of is *not null* and references *Student.UNI*

Consider the logical data model above. The one-to-one relationship is modeled using two foreign keys, one in each table.

- Why does this make it difficult to insert data into the tables?
- What is a (simple) fix for this, i.e., how would you model a one-to-one relationship?

- There is a circular reference problem when inserting rows. For example, when inserting rows into `students`, you have to know the mentor UNI, which has not been added into the `faculty` yet.
- To fix this, we can allow the FK to be NULL temporarily.

W4

The relational model places restrictions on attributes. Many data scenarios have more complex types of attributes. Briefly explain the following types of attributes:

1. Simple attribute
2. Composite attribute

3. Derived attribute
4. Single-value attribute
5. Multi-value attribute

1. Simple Attribute: cannot be divided any further and consists of a single atomic value.
2. Composite Attribute: made up of multiple simple attributes that can be divided into smaller parts. For example, a full name attribute could be divided into first name and last name.
3. Derived Attribute: not stored in the table(database) but is derived from other stored attributes.
4. Single-value Attribute: holds only one value for a given attribute at a time.
5. Multi-value Attribute: can hold multiple values for a single attribute.

W5

The slides associated with the recommended textbook list six basic relational operators:

1. select: σ
2. project: π
3. union: ∪
4. set difference: -
5. Cartesian product: ×
6. rename: ρ

The list does not include join: ⋈. This is because it is possible to derive join using more basic operators. Explain how to derive join from the basic operators.

Like the example in the slide, assume there are two tables `instructor` and `teacher`, and we want to join with `ID`. Then we can do it with the following operation:

$$\sigma_{instructor.ID=teacher.ID}(instructor \times teacher) = instructor \bowtie teacher$$

W6

Explain how using a natural join may produce an incorrect/unexpected answer.

If there are columns that have the same name but are not intended to join on, a natural join will make mistakes.

W7

The UNION and JOIN operations both combine two tables. Describe their differences.

UNOIN is used to append rows from one query with the rows from one another, concating the two tables "vertically".

JOIN is used to append two tables by matching values in the column, concating them "horizentally".

W8

Briefly explain the importance of integrity constraints. Why do non-atomic attributes cause problems/difficulties for integrity constraints?

Integrity constraints are designed to keep the database accurate and consistent. Without integrity constraints, it will require lots of users know what to do and never make mistakes.

Non-atomic attributes make it difficult to do intefrity because the individual elements can not be uniquely identified.

W9

What is the primary reason for creating indexes? What are the negative effects of creating unnecessary indexes?

- The primary reason for indexes is to improve query efficiency by only querying a subset of the atrributes.
- If creating unnecessary indexes, it will cost extra storage spaces and slower our query.

W10

Consider the table `time_slot` from the sample database associated with the recommended textbook.

- The data type for the column `day` is `char(1)`. Given the data types MySQL supports, what is a better data type for `day`?
- What is a scenario that would motivate creating an index on `day`?

- It is better to use ENUM type for `day`. Like ENUM('M', 'T', 'W', 'R', 'F', 'S', 'U') in order to keep integrity.
- If queries frequently filter or join `day`, it will be better to create an index on `day`.

Relational Algebra

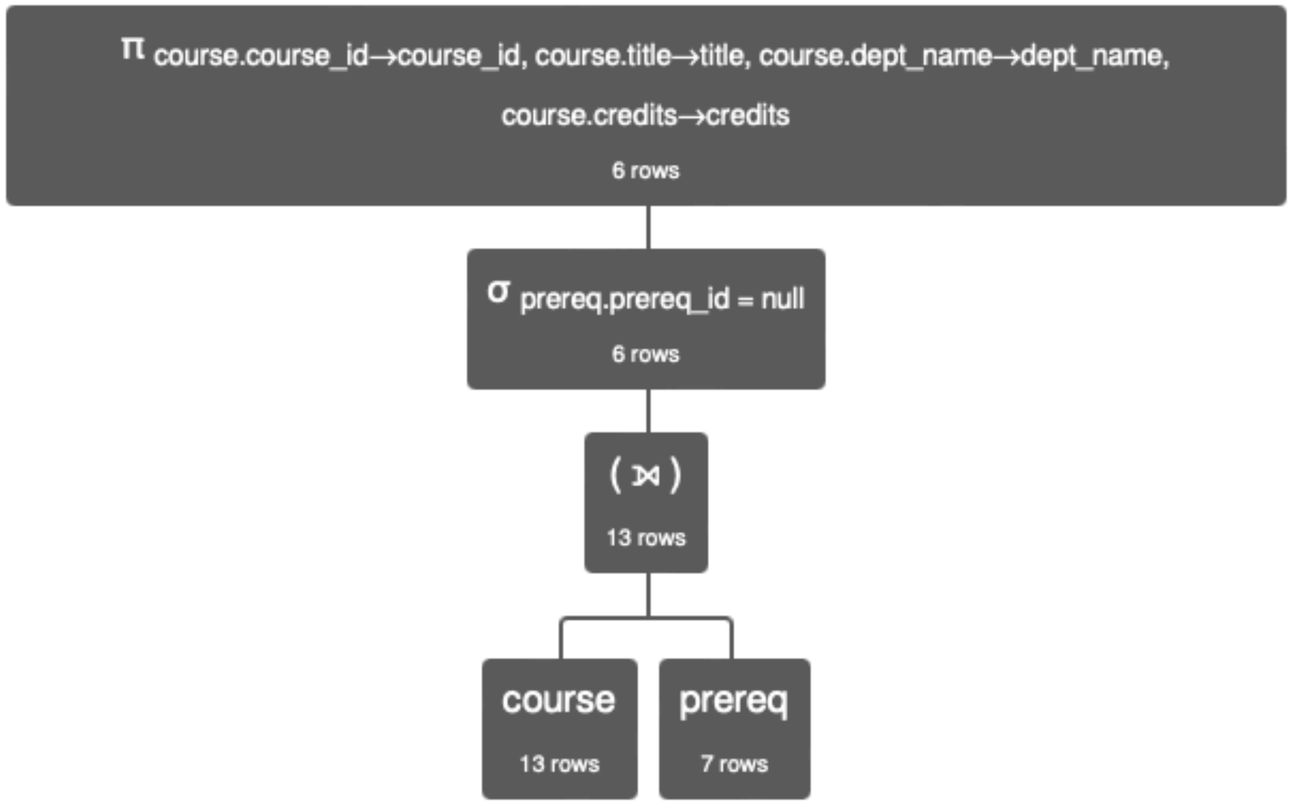
R1

- Write a relational algebra statement that produces a relation showing **courses that do not have a prereq**
- Your output should have the following columns (names should match exactly; there should be no prefixes):
    - `course_id`
    - `title`
    - `dept_name`
    - `credits`
- You may not use the anti-join: ▷ operator
- You should use the `course` and `prereq` tables

Algebra statement:

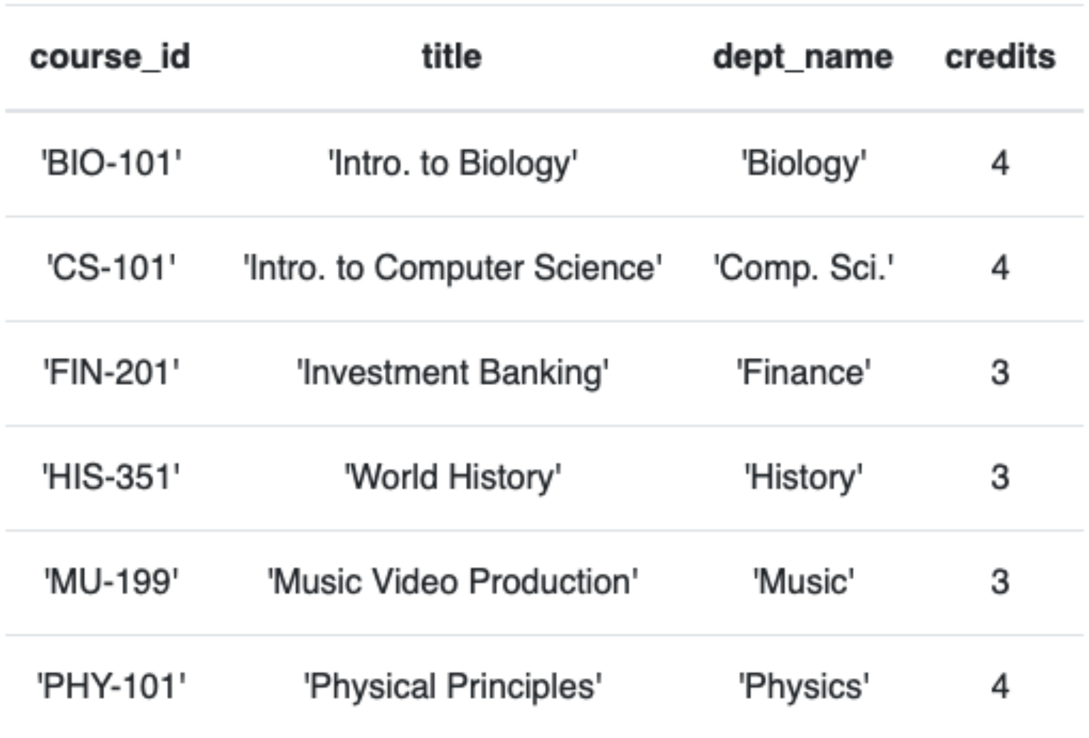
```
π course_id←course.course_id, title←course.title, dept_name←course.dept_name, credits←course.credits
(σ prereq.prereq_id=NULL (course⟕prereq))
```

Execution:


π course.course_id→course_id, course.title→title, course.dept_name→dept_name, course.credits→credits
6 rows
σ prereq.prereq_id = null
6 rows
(⋈)
13 rows
course
13 rows
prereq
7 rows

$\pi_{\text{course.course\_id}\rightarrow\text{course\_id},\ \text{course.title}\rightarrow\text{title},\ \text{course.dept\_name}\rightarrow\text{dept\_name},\ \text{course.credits}\rightarrow\text{credits}}\ (\ \sigma_{\text{prereq.prereq\_id = null}}\ (\ \text{course} \rtimes \text{prereq}\ )\ )$

Execution time: 5 ms



course_id	title	dept_name	credits
'BIO-101'	'Intro. to Biology'	'Biology'	4
'CS-101'	'Intro. to Computer Science'	'Comp. Sci.'	4
'FIN-201'	'Investment Banking'	'Finance'	3
'HIS-351'	'World History'	'History'	3
'MU-199'	'Music Video Production'	'Music'	3
'PHY-101'	'Physical Principles'	'Physics'	4

R1 Execution Result

R2

- Write a relational algebra query that produces a relation showing **students who have taken sections taught by their advisors**
    - A section is identified by `(course_id, sec_id, semester, year)`
- Your output should have the following columns (names should match exactly; there should be no prefixes):
    - `student_name`
    - `instructor_name`
    - `course_id`
    - `sec_id`
    - `semester`

    - `year`
    - `grade`
- You should use the `takes`, `teaches`, `advisor`, `student`, and `instructor` tables
- As an example, one row you should get is

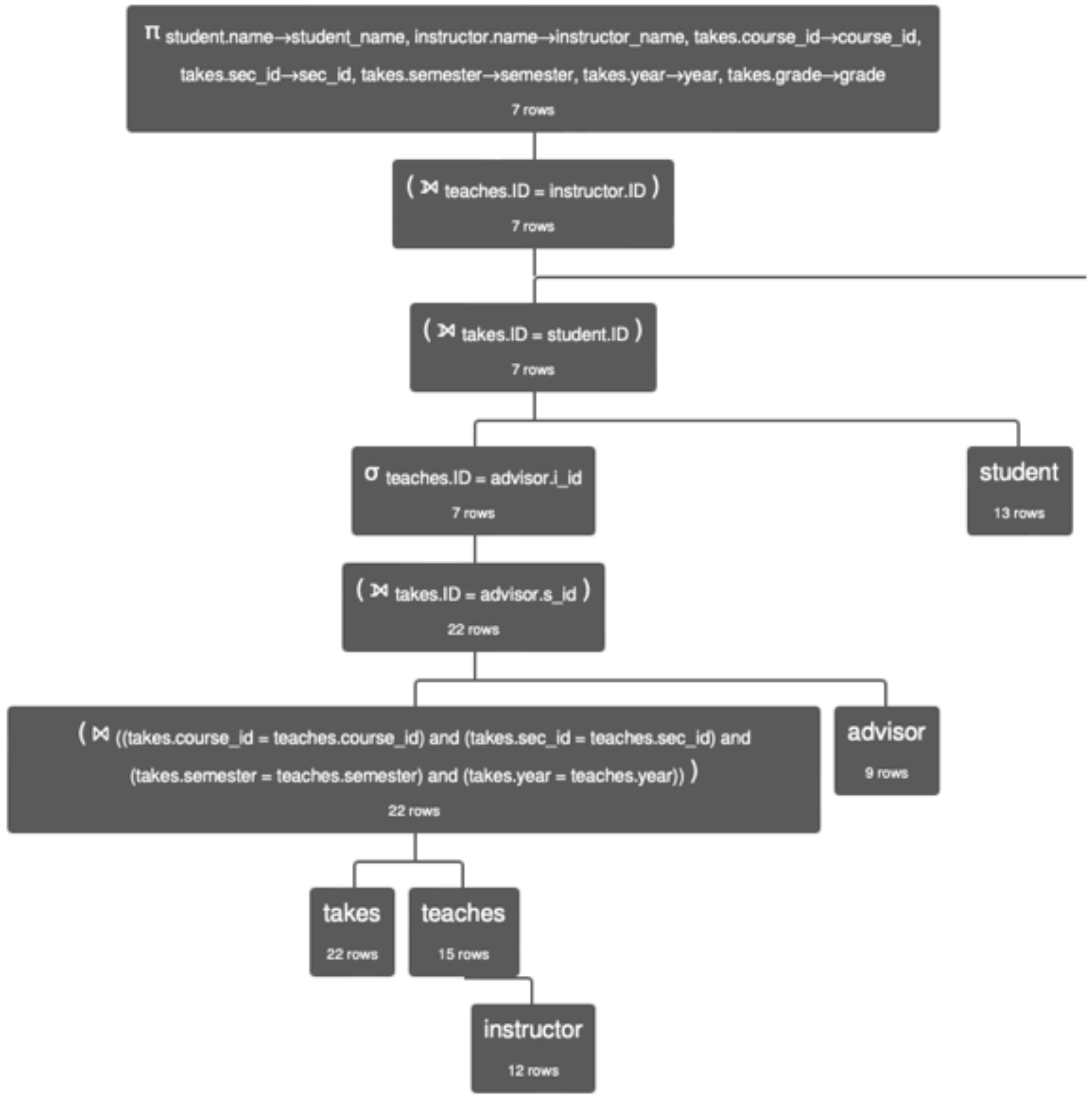
student_name	instructor_name	course_id	sec_id	semester	year	grade
'Shankar'	'Srinivasan'	'CS-101'	1	'Fall'	2009	'C'

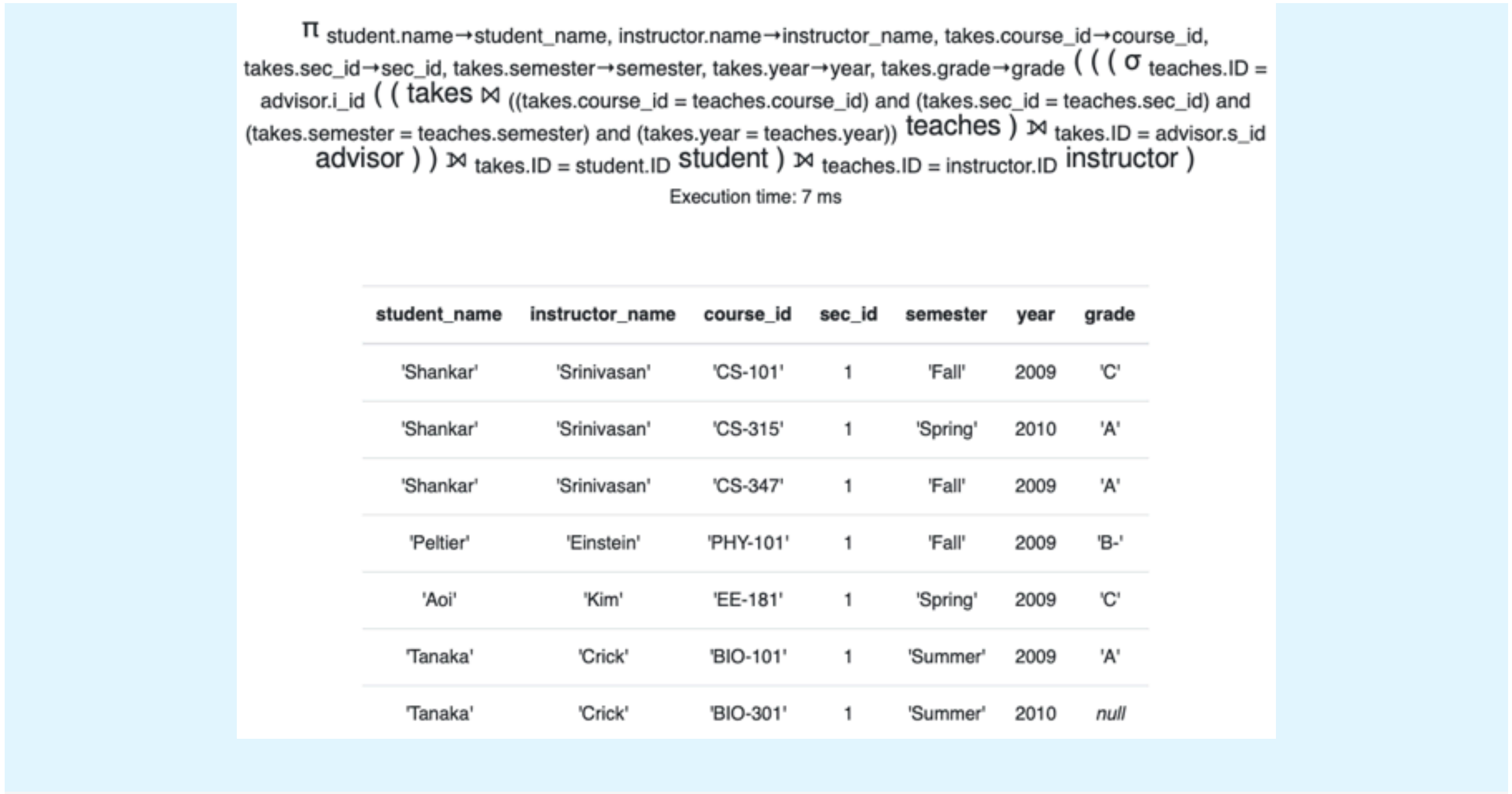
- Shankar took CS-101, section 1 in Fall of 2009, which was taught by Srinivasan. Additionally, Srinivasan advises Shankar

Algebra statement:

```
π student_name←student.name, instructor_name←instructor.name, course_id←takes.course_id,
sec_id←takes.sec_id, semester←takes.semester, year←takes.year, grade←takes.grade
(((σ teaches.ID=advisor.i_id
((takes⋈ ((takes.course_id=teaches.course_id)∧(takes.sec_id=teaches.sec_id)∧
(takes.semester=teaches.semester)∧(takes.year=teaches.year)) teaches)
⋈ takes.ID=advisor.s_id advisor)
)⋈ takes.ID=student.ID student)⋈ teaches.ID=instructor.ID instructor)
```

Execution:


π student.name→student_name, instructor.name→instructor_name, takes.course_id→course_id, takes.sec_id→sec_id, takes.semester→semester, takes.year→year, takes.grade→grade
7 rows
(⋈ teaches.ID = instructor.ID)
7 rows
(⋈ takes.ID = student.ID)
7 rows
σ teaches.ID = advisor.i_id
7 rows
student
13 rows
(⋈ takes.ID = advisor.s_id)
22 rows
(⋈ ((takes.course_id = teaches.course_id) and (takes.sec_id = teaches.sec_id) and (takes.semester = teaches.semester) and (takes.year = teaches.year)))
22 rows
advisor
9 rows
takes
22 rows
teaches
15 rows
instructor
12 rows



$$\pi_{\text{student.name}\rightarrow\text{student\_name},\ \text{instructor.name}\rightarrow\text{instructor\_name},\ \text{takes.course\_id}\rightarrow\text{course\_id},\ \text{takes.sec\_id}\rightarrow\text{sec\_id},\ \text{takes.semester}\rightarrow\text{semester},\ \text{takes.year}\rightarrow\text{year},\ \text{takes.grade}\rightarrow\text{grade}} (((\sigma_{\text{teaches.ID} = \text{advisor.i\_id}} ((\text{takes} \bowtie_{((\text{takes.course\_id} = \text{teaches.course\_id}) \text{ and } (\text{takes.sec\_id} = \text{teaches.sec\_id}) \text{ and } (\text{takes.semester} = \text{teaches.semester}) \text{ and } (\text{takes.year} = \text{teaches.year}))} \text{teaches}) \bowtie_{\text{takes.ID} = \text{advisor.s\_id}} \text{advisor})) \bowtie_{\text{takes.ID} = \text{student.ID}} \text{student}) \bowtie_{\text{teaches.ID} = \text{instructor.ID}} \text{instructor})$$

Execution time: 7 ms

student_name	instructor_name	course_id	sec_id	semester	year	grade
'Shankar'	'Srinivasan'	'CS-101'	1	'Fall'	2009	'C'
'Shankar'	'Srinivasan'	'CS-315'	1	'Spring'	2010	'A'
'Shankar'	'Srinivasan'	'CS-347'	1	'Fall'	2009	'A'
'Peltier'	'Einstein'	'PHY-101'	1	'Fall'	2009	'B-'
'Aoi'	'Kim'	'EE-181'	1	'Spring'	2009	'C'
'Tanaka'	'Crick'	'BIO-101'	1	'Summer'	2009	'A'
'Tanaka'	'Crick'	'BIO-301'	1	'Summer'	2010	*null*

R2 Execution Result

R3

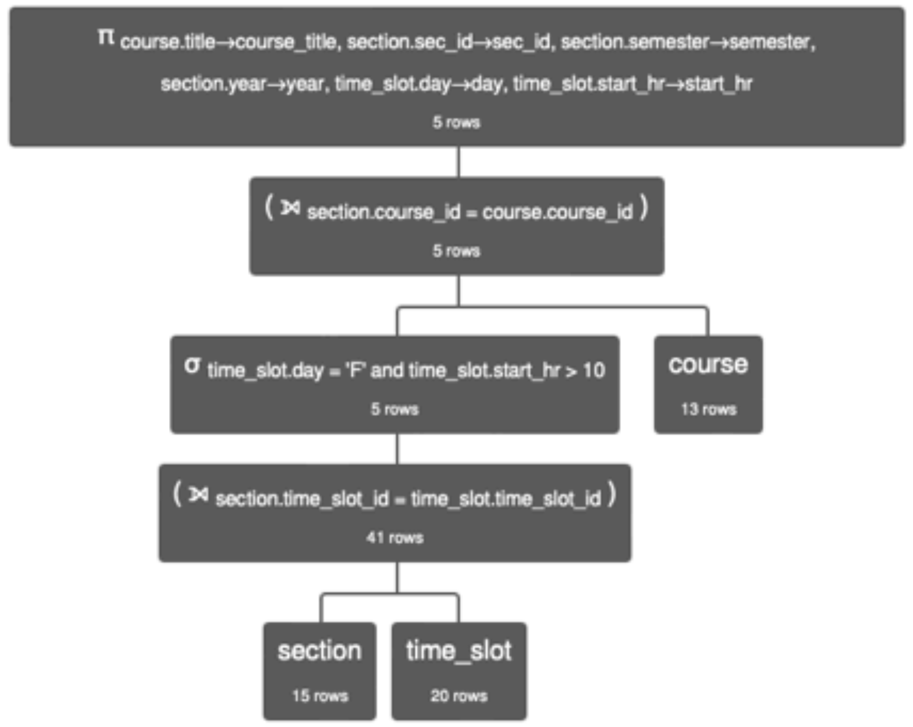
- Write a relational algebra query that produces a relation showing **sections that occur on Friday and start after 10 AM**
- Your output should have the following columns (names should match exactly; there should be no prefixes):
  - `course_title`
  - `sec_id`
  - `semester`

    - `year`
    - `day`
    - `start_hr`
- You should use the `course`, `section`, and `time_slot` tables

Algebra statement:

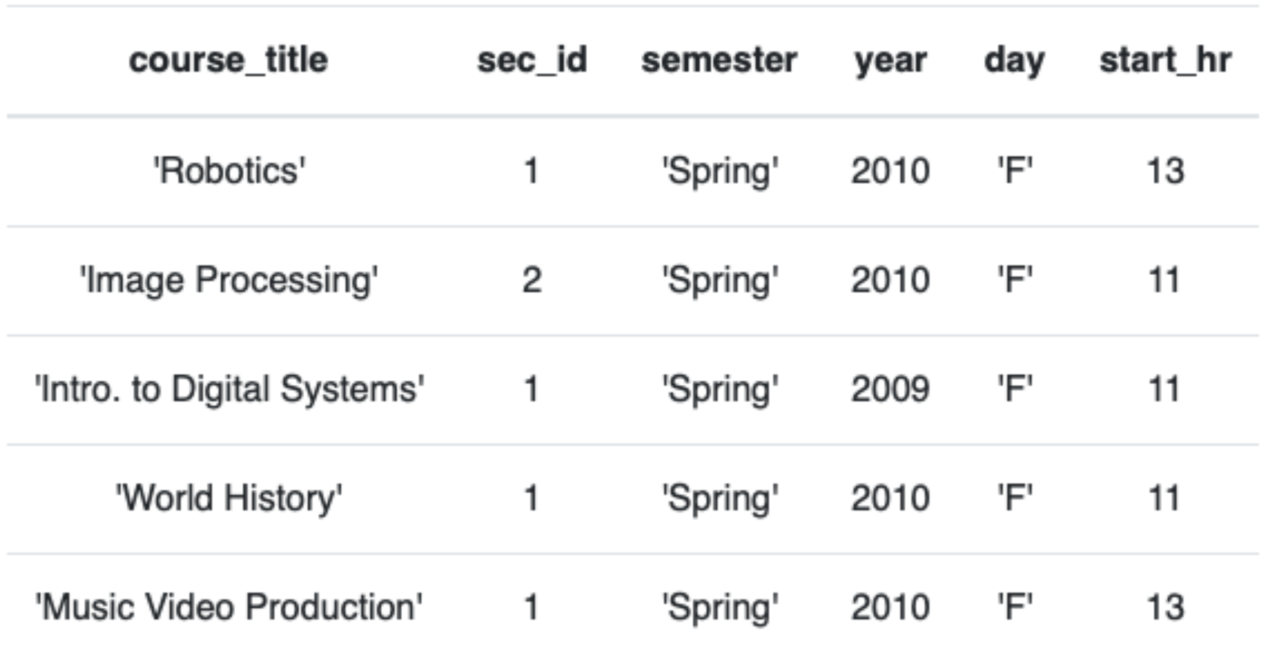
```
π course_title←course.title, sec_id←section.sec_id, semester←section.semester, year←section.year, day←time_slot.day,
start_hr←time_slot.start_hr
((σ time_slot.day='F' ∧ time_slot.start_hr>10
(section⋈ section.time_slot_id=time_slot.time_slot_id time_slot))
⋈ section.course_id=course.course_id course)
```

Execution:


π course.title→course_title, section.sec_id→sec_id, section.semester→semester, section.year→year, time_slot.day→day, time_slot.start_hr→start_hr
5 rows
(⋈ section.course_id = course.course_id)
5 rows
σ time_slot.day = 'F' and time_slot.start_hr > 10
5 rows
course
13 rows
(⋈ section.time_slot_id = time_slot.time_slot_id)
41 rows
section
15 rows
time_slot
20 rows

$$\pi_{\text{course.title}\rightarrow\text{course\_title, section.sec\_id}\rightarrow\text{sec\_id, section.semester}\rightarrow\text{semester, section.year}\rightarrow\text{year, time\_slot.day}\rightarrow\text{day, time\_slot.start\_hr}\rightarrow\text{start\_hr}} \left(\left(\sigma_{\text{time\_slot.day = 'F' and time\_slot.start\_hr > 10}} \left(\text{section} \bowtie_{\text{section.time\_slot\_id = time\_slot.time\_slot\_id}} \text{time\_slot} \right) \right) \bowtie_{\text{section.course\_id = course.course\_id}} \text{course} \right)$$

Execution time: 3 ms



course_title	sec_id	semester	year	day	start_hr
'Robotics'	1	'Spring'	2010	'F'	13
'Image Processing'	2	'Spring'	2010	'F'	11
'Intro. to Digital Systems'	1	'Spring'	2009	'F'	11
'World History'	1	'Spring'	2010	'F'	11
'Music Video Production'	1	'Spring'	2010	'F'	13

R3 Execution Result