W4111_2025_002_1: Introduction to Databases: Homework 2

Overview

Scope

The material in scope for this homework is:

- The content of lectures 1, 2 and 3.
- The slides associated with the recommended textbook for
    - Chapter 1.
    - Chapter 2.
    - Chapter 3.
    - Chapter 4 slides 4.4 to 4.13, 4.36 to 4.50 except for slide 4.35 (Transactions).
    - Chapter 6 slides 6.1 to 6.53.

Submission Instructions

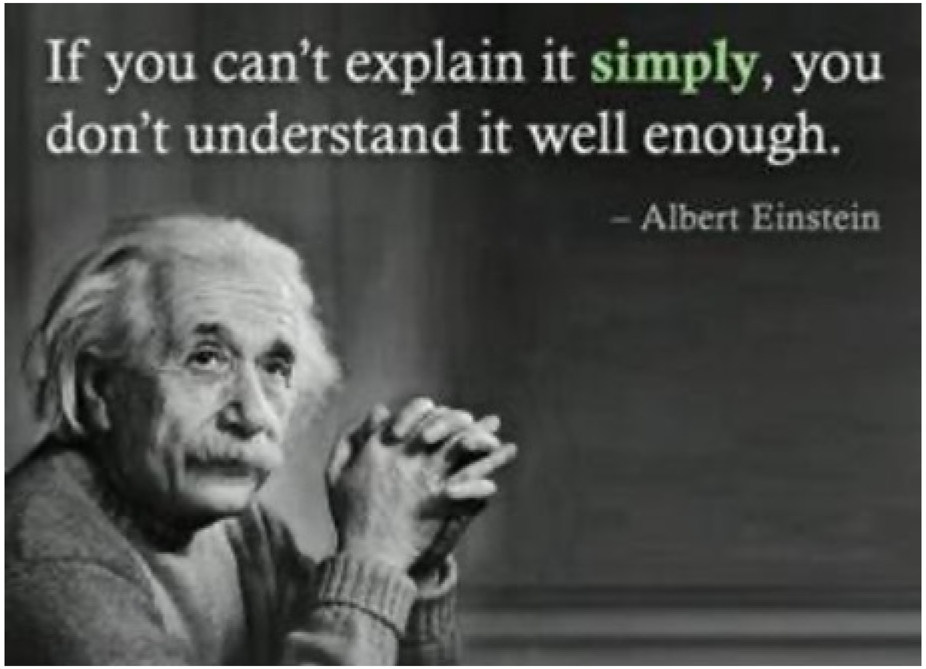
- Due date: 2025-Feb-23, 11:59 PM EDT on GradeScope.

- You submit on GradeScope. We will create a GradeScope submission for the homework.

- Your submission is a PDF of this notebook. You must tag the submission with locations in the PDF for each question.

There is a post/mega-thread on Ed Discussions that we will use to resolve questions and issues with respect to homework 2.

Brevity


If you can't explain it simply, you don't understand it well enough.
– Albert Einstein

Brevity

Students sometimes just write a lot of words hoping to get something right. We will deduct points if your answer is too long.

Initialization

In [4]:
```
import copy
```

In [7]:
```
import json
```

In [5]:
```
import pandas
```

In [366…
```
# You should have installed the packages for previous homework assignments
#
import pymysql
import sqlalchemy
```

In [30]:
```
import numpy
```

In [367…
```
# You have installed and configured ipython-sql for previous assignments.
# https://pypi.org/project/ipython-sql/
```

```
#
%load_ext sql
```

```
The sql extension is already loaded. To reload it, use:
  %reload_ext sql
```

In [368…

```
# This is a hack to fix a version problem/incompatibility  with some of the packages and magics.
#
%config SqlMagic.style = '_DEPRECATED_DEFAULT'
```

In [369…

```
# Make sure that you set these values to the correct values for your installation and
# configuration of MySQL
#
db_user = "root"
db_password = "dbuserdbuser"
```

In [370…

```
# Create the URL for connecting to the database.
# Do not worry about the local_infile=1, I did that for wizard reasons that you should not have to use.
#
db_url = f"mysql+pymysql://{db_user}:{db_password}@localhost?local_infile=1"
```

In [371…

```
# Initialize ipython-sql
#
%sql $db_url
```

In [372…

```
# Your answer will be different based on the databases that you have created on your local MySQL instance.
#
%sql show tables from db_book
```

```
 * mysql+pymysql://root:***@localhost?local_infile=1
15 rows affected.
```

Out[372...

Tables_in_db_book
advisor
classroom
course
course2
department
instructor
person
person_section
prereq
section
section2
student
takes
teaches
time_slot

In [37]:

```
from sqlalchemy import create_engine
default_engine = create_engine(db_url)
```

In [374...

```
result_df = pandas.read_sql(
    "show tables from db_book", con=default_engine
)
result_df
```

Out[374…

	Tables_in_db_book
0	advisor
1	classroom
2	course
3	course2
4	department
5	instructor
6	person
7	person_section
8	prereq
9	section
10	section2
11	student
12	takes
13	teaches
14	time_slot

Written Questions

Data Types and Domains

Question

Columbia University has an online directory of classes. One of the properties in the data defining a class is the section key. The section key for our database class this spring is “20251COMS4111W002.” The section key for one of this spring’s Calculus I classes is “20251MATH1101V002.” The “data type” for section key is clearly a text string. The domain of this attribute is related to the data type but is different. Briefly explain the concept of a domain and how it differs from a data type. Use section key and your knowledge of Columbia University to provide examples of the difference.

Answer

Domains are the set of permitted values an attribute can take. This may be a smaller set than data type, which only constrains the 'type' of value. For example boolean or number. However, domains can be more restrictive and be informed by context and meaning. For example in the case of section key, while the data type is a text string, the domain is a string with the first 4 characters being a valid year, the 6th to 9th characters are 4 letter codes of a valid department, and so on.

Associative Entity

Question

When modeling a relationship between two entity sets using Crow’s Foot Notation or implementing in SQL, what are the two reasons that you must use an associative entity?

Answer

When the relationship is many to many, and when the relationship has properties/attributes. Crow's foot and SQL only has entities and hence these relationships must be represented as an entity, and in the case of many-to-many the relationship cannot be represented using foreign keys, hence necessitating representation as an entity that contains tuples of keys.

Recognizing Entity Types

Question

Examine the schema/SQL DDL for the sample database associated with the recommended textbook. Which tables are associative entities, and which tables are weak entities? Briefly explain your answer.

Answer

advisor: this is an associative entity, it represents a many-to-many relationship (students can have multiple instructors as advisors, and vice versa for advisees)

prereq: this is an associative entity, it represents a many-to-many relationship (courses can have many prereqs, or be a prereq of many courses)

section: this is a weak entity, the existence of sections depends on course entities

takes: this is an associative entity, it represents a many-to-many relationship with attributes (students can take many courses, and courses can have many students, plus such a relationship also have grades attributes) teaches: this is an associative entity, it represents a many-to-many relationship (an instructor can teach multiple sections, and sections can have multiple instructors)

Atomic Domains

Question

The lecture 3 slides contained the following:

- “Every domain must contain atomic values(smallest indivisible units) which means composite and multi-valued attributes are not allowed."
- This is sometimes known as “First Normal Form.” We will cover normalization later in the semester.

Briefly explain this concepts and give examples of atomic and non-atomic domains using people's names.

Answer

If a domain is atomic, then means that the values of the domain cannot be subdivided into more attributes. For example, a full name such as Will A. Smith is not atomic because it can be split into a first name, middle name, and last name. These three are atomic because it would not make sense to split them further.

arity

Question

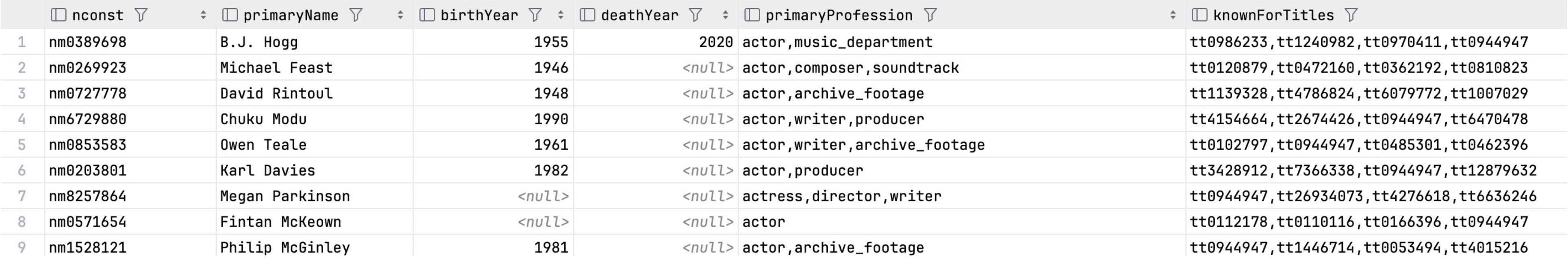
For set operations in the relational algebra, the relations must have the same arity. Briefly explain the concept of arity. The relational scheme definitions for student and instructor for the data schema associated with the recommended textbook are $student(ID, name, dept\_name, tot\_cred)$ and $instructor(ID, name, dept\_name, salary)$. Do these relations have the same arity?

Answer

Arity is the number of attributes a relation has. The relations of 'student' and 'instructor' both have arity of 4, so they have the same arity.

Complex Attributes

Question



	nconst	primaryName	birthYear	deathYear	primaryProfession	knownForTitles
1	nm0389698	B.J. Hogg	1955	2020	actor,music_department	tt0986233,tt1240982,tt0970411,tt0944947
2	nm0269923	Michael Feast	1946	<null>	actor,composer,soundtrack	tt0120879,tt0472160,tt0362192,tt0810823
3	nm0727778	David Rintoul	1948	<null>	actor,archive_footage	tt1139328,tt4786824,tt6079772,tt1007029
4	nm6729880	Chuku Modu	1990	<null>	actor,writer,producer	tt4154664,tt2674426,tt0944947,tt6470478
5	nm0853583	Owen Teale	1961	<null>	actor,writer,archive_footage	tt0102797,tt0944947,tt0485301,tt0462396
6	nm0203801	Karl Davies	1982	<null>	actor,producer	tt3428912,tt7366338,tt0944947,tt12879632
7	nm8257864	Megan Parkinson	<null>	<null>	actress,director,writer	tt0944947,tt26934073,tt4276618,tt6636246
8	nm0571654	Fintan McKeown	<null>	<null>	actor	tt0112178,tt0110116,tt0166396,tt0944947
9	nm1528121	Philip McGinley	1981	<null>	actor,archive_footage	tt0944947,tt1446714,tt0053494,tt4015216

Typical Input Data

There are six attributes in the sample data above.

1. For each attribute, specify if the attribute is: *simple* or *composite, single valued* or *multi-valued* and *derived* or *not serived.* Explain your choices.
2. For which attributes would you use a `check constraint` and explain the constraint.

Answer

1. nconst: simple, single valued, not derived.

I would check constraint of "first two chars are nm" and "last 7 chars are numeric", and since it seems to be used as primary key it would contrain it as not null

2. primaryName: composite, single valued, not derived.

3. birthYear: simple, single valued, not derived.

I would check constraint of "is a year less than current year" and "4 digit numeric"

4. deathYear: simple, single valued, not derived.

I would check constraint of "is a year less than current year" and "4 digit numeric"

5. primaryProfession: simple, multi-valued, not derived.

I would check constraint that the values are valid (possibly reference another relation containing all valid professions)

6. knownForTitles: simple, multi-valued, possibly derived (maybe aggregated from title entities containing the actors involved).

I would check constraint that the values are valid (possibly reference another relation containing all valid title keys)

Relational Algebra Assignment Operator

Question

One explanation for the assignment operator is, "With the assignment operation, a query can be written as a sequential program consisting of a series of assignments followed by an expression whose value is displayed as the result of the query."

Use the assignment operator to write a program using assignments to rewrite the query

```
π course_id, course_title, prereq_course_id, prereq_title
(
        (π course_id, course_title←title, prereq_id (course ⋈ prereq)
)
⋈ prereq_id=prereq_course_id
(π prereq_course_id←course_id, prereq_title←title (course)))
```

What are two benefits of writing complex queries using a set of statements?

Answer

The benefits are:

1. Makes the queries more readable and easy to understand and debug
2. Improves performance by storing results. These results do not need to be recalculated every time it is referenced.

```
courseAndCorrespondingPrereqId = (π course_id, course_title←title, prereq_id (course ⋈ prereq))

coursesAsPrereqs = π prereq_course_id←course_id, prereq_title←title (course)

courseAndPrereq = courseAndCorrespondingPrereqId ⋈ prereq_id=prereq_course_id (coursesAsPrereqs)

π course_id, course_title, prereq_course_id, prereq_title
(courseAndPrereq)
```

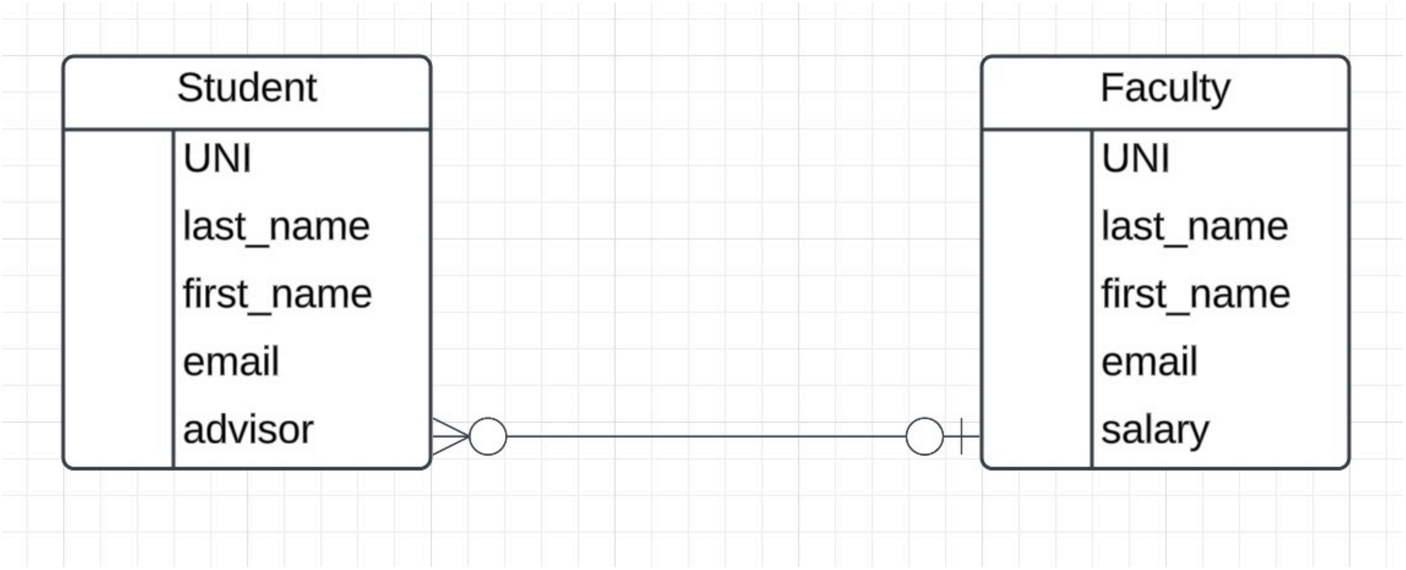
Constraints

Question

What are four types of constraints that may apply to a single relation/table? What type of constraint can apply to more than one table?

Consider the partial logical schema below. A student *may* or *may not* have an advisor.

Briefly explain which constraints you would apply.



Constraints

Answer

The four types of constrains are:

1. Domain Constraints
2. Key/Uniqueness Contraints
3. Enitity Integrity Constraints
4. Referential Integrity Constraints

Referential Integrity Constraints applies to more than one table since it is a constraint on tuples between multiple tables

I would apply Domain Constraints on UNI (text), first_name (atomic string), last_name (atomic string), email (string), advisor (text), salary (number)

Key/Uniqueness Contraints on UNI

Entity Integrity Constraints on UNI

Referential Integrity Constraints on advisor (must reference an actual existing faculty)

SELECT versus UNION

Question

In SQL, `SELECT` and `UNION` behave differently with respect to duplicates in the result set. Explain the difference.

Taking a step back, if tables have primary keys, how are duplicates in a query result even possible?

Answer

If the result of SELECT has duplicates, then SELECT keeps them. However, when the result of UNION has duplicates, the duplicates are removed such that only one remains. This is because UNION is a set operation and the result needs to be a set.

If a SELECT operation omits the primary keys, duplicates in the result are possible. Tables may have also been misdefined, ie a primary key contraint was forgotten. Using natural join with attributes that are not the primary key may also lead to this happening.

Associative Entity

Question

Consider the query below. What is required of the result of the two subqueries? What is the name for the type of subquery?

```
select
    s_id as student_id,
    (select name from student where student.ID=s_id) as student_name,
    i_id as advisor_id,
    (select name from instructor where instructor.ID=i_id) as instructor_name
from
    advisor;
```

Answer

The first subquery returns the name of the student with student id s_id

The second subquery returns the name of the instructor with instructor id i_id

They are known as scalar subqueries since a single value is expected.

Practical Questions

Set Operations in SQL

Question

Using the sample data associated with the recommended textbook,

1. What is wrong with the query below.
2. Write and execute a query that produces accurate results that contains all of the information.

```
select * from student where dept_name='Comp. Sci.'
union
select * from instructor where dept_name='Comp. Sci.'
```

Answer

While 'student' and 'instructor' have the same arity, they have different attributes of 'tot_cred' and 'salary' which are not compatible. One deals with academic credits while the other with monetary value. It does not make sense to put them together, but the 'union' query forces them into the same column.

Please place and execute your SQL statement below.

In [389...

```
#Since MySQL does not support FULL OUTER JOIN, I will use the union of a NATURAL LEFT OUTER JOIN and NATURAL RIGHT OUTER JOIN
```

In [415...

```
%%sql

select * from
    (select ID, name, dept_name, salary, tot_cred from student NATURAL LEFT OUTER JOIN instructor
        union
    select ID, name, dept_name, salary, tot_cred from student NATURAL RIGHT OUTER JOIN instructor) as instructor_and_student
where dept_name="Comp. Sci."
```

```
 * mysql+pymysql://root:***@localhost?local_infile=1
7 rows affected.
```

Out[415…

ID	name	dept_name	salary	tot_cred
00128	Zhang	Comp. Sci.	None	102
12345	Shankar	Comp. Sci.	None	32
54321	Williams	Comp. Sci.	None	54
76543	Brown	Comp. Sci.	None	58
10101	Srinivasan	Comp. Sci.	65000.00	None
45565	Katz	Comp. Sci.	75000.00	None
83821	Brandt	Comp. Sci.	92000.00	None

Set Operations in Relational Algebra

Question

The query below produces information about instructors that are not advisors. You must write an equivalent relational algebra expression that contains only set operators and project. Replace the query and screen capture below with you answer.

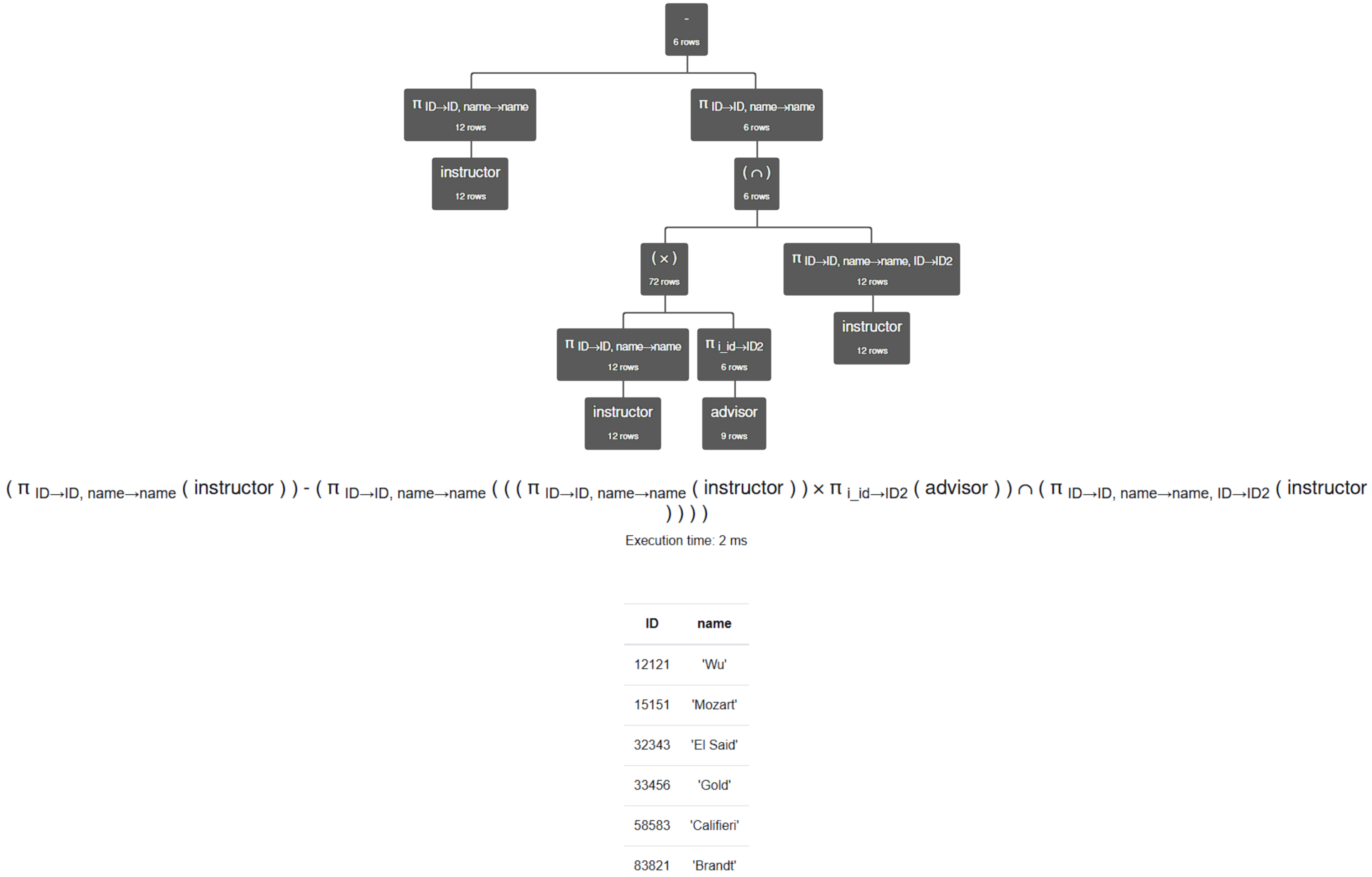
Answer

```
/*
    Replace the statement below with your answer.
    π ID←ID, name←name (σ i_id=null (instructor ⟕ ID=i_id advisor))

Answer:
*/


(π ID←ID, name←name (instructor)) - (π ID←ID, name←name (((π ID←ID, name←name (instructor)) × π ID2←i_id (advisor)) ∩
(π ID←ID, name←name, ID2←ID (instructor))))
```

Replace the images below with your screenshot.



$$(\pi_{ID \rightarrow ID,\ name \rightarrow name} (\text{instructor})) - (\pi_{ID \rightarrow ID,\ name \rightarrow name} (((\pi_{ID \rightarrow ID,\ name \rightarrow name} (\text{instructor})) \times \pi_{i\_id \rightarrow ID2} (\text{advisor})) \cap (\pi_{ID \rightarrow ID,\ name \rightarrow name,\ ID \rightarrow ID2} (\text{instructor}))))$$

Execution time: 2 ms

ID	name
12121	'Wu'
15151	'Mozart'
32343	'El Said'
33456	'Gold'
58583	'Califieri'
83821	'Brandt'

Spring Courses: Your Answer

ER-Modeling

Question

Consider the following scenario.

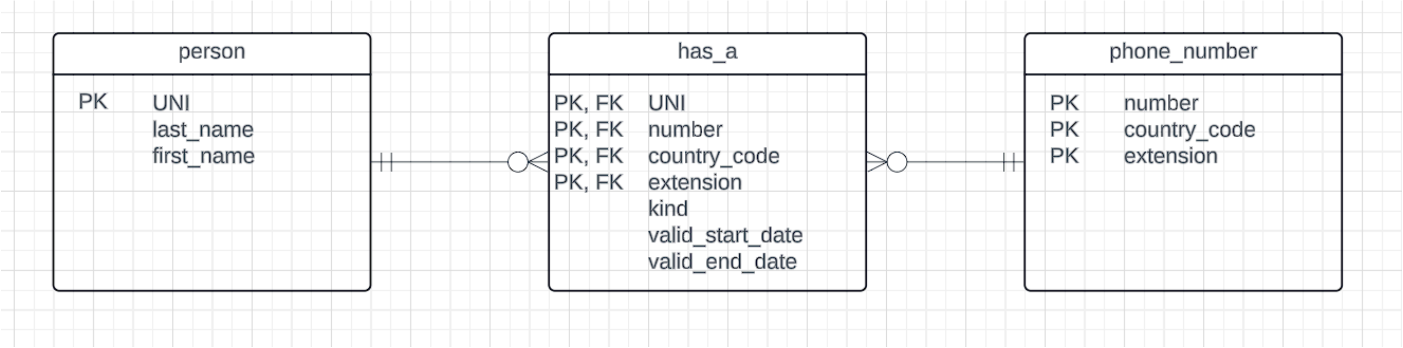
1. There are two entity types:
   A. `person` has attributes `last_name, first_name` and `UNI.` The primary key is `UNI.`
   B. `phone_number` has the attributes `country_code, number` and `extension.` The primary key is a composite of all 3 attributes.
2. There is one relationships -- `has_a` is a relationship between a `person` and `phone_number.`
   - A `person` may be related to 0, 1 or many `phone_numbers.`
   - A `phone_number` may be related to 0, 1 or many `persons.`
   - Each relationship has 3 properties:
     A. `kind` is in the set `{home, mobile, work, voicemail, supporting_admin}.` It is possible that the `kind` is not known.
     B. `valid_start_date` defines when the association started.
     C. `valid_end_date` defines when the association ended.

Using Crow's Foot Notation and a tool like Lucidchart, draw a logical ER diagram modeling the relationship. You may add notes/comments that explain decisions you make.

Answer

Since the has_a relationship between person and phone_number is both many to many and has additional attributes, I modeled it as an entity. Each has_a entity must refer to exactly one person and exactly one phone_number. Each person or phone_number entity can reference 0,1,or many has_a entities. Since we are drawing a logical ER diagram, I decided to omit details such as the constraints on 'kind'

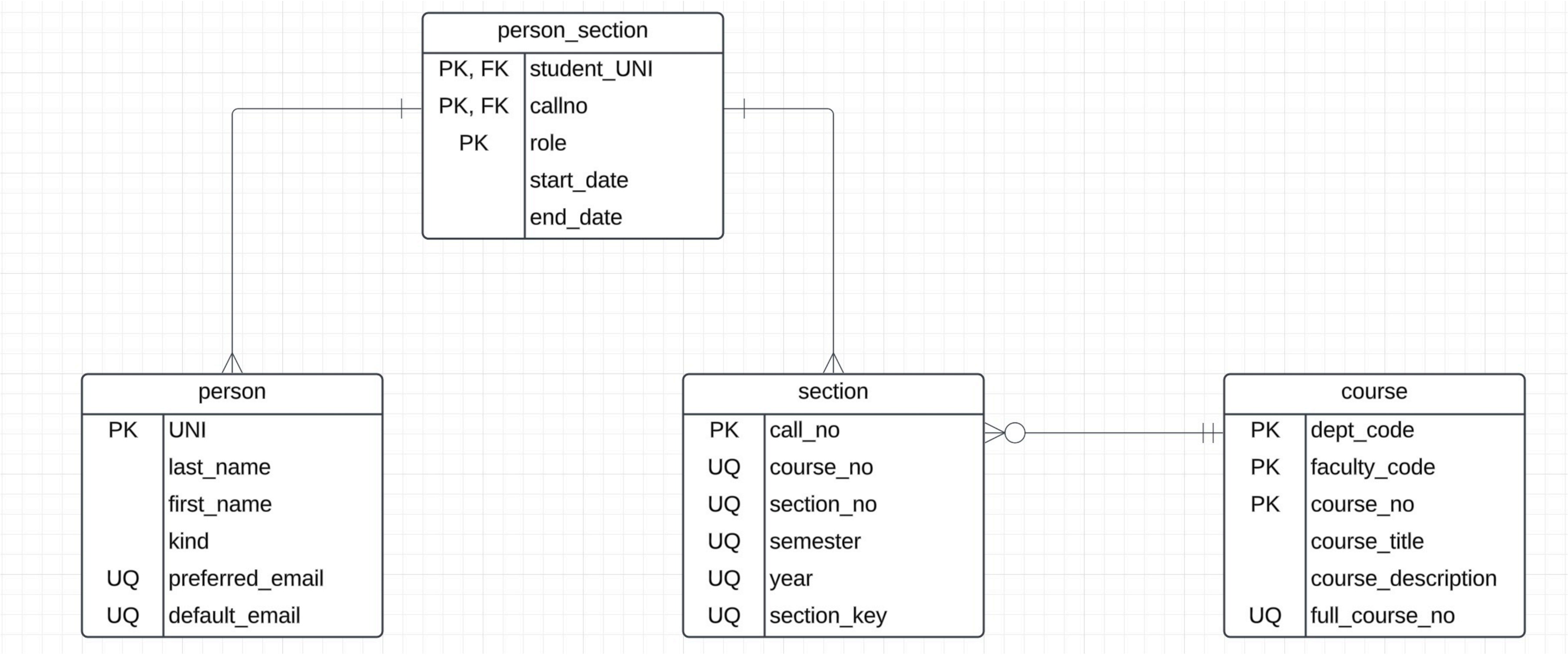
Replace the images below with your screenshot.



Spring Courses: Your Answer

ER Diagram to DDL

Question



ER Diagram to DDL

Consider the preceding, **approximate** ER logical model diagram. The diagram is approximate because the definition below of the model may require minor changes in the implemented DDL relative to the diagram. For example, you may have to add constraints, columns not shown, etc.

The semantics/requirements are below.

A sample `person` record for me in `person` would be in the form

```
{dff9, Ferguson, Donald, Faculty, donald.ferguson@cs.columbia.edu, dff9@columbia.edu}
```

- The default email is always of the form `uni@columbia.edu`.
- Preferred email is always `UNIQUE` but a person *may not have* a preferred email.
- The possible values for `kind` are one of `{Student, Faculty, Staff}`.

A sample `course` record for our *Intro. to Databases* course would be in the form

```
{COMS, W, 4111, Introduction to Databases, OMG! This class is terrifying., COMSW4111}
```

- `dept_code` is always 4 characters and will not contain a digit, space, -, or _
- `faculty_code` is one of `{W, C, E, B, G}`.
- `course_no` is always 4 digits and cannot begin with a 0.
- `full_course_no` is the concatenation of `dept_code, faculty_code, course_no`.

A sample `section` for our session of COMSW4111 would be

```
{11969, COMSW4111, 002, 1, 2025, COMSW4111_002_1_2025}
```

- `call_no` is always 5 digits and may begin with 0.
- `course_no` is the same as `full_course_no` in `course`.
- `section_no` is always 3 characters. It can be 3 digits and may start with 0. Or, it can be of the form `V02`, that is starts with `V` and has two digits.
- `year` has the obvious meaning and constraints.
- `section_key` is the concatenation of the fields with the _ delimiter.

A sample `person_section` for me would be

```
{dff9, 11969, instructor, 20250125, 20250502}
```

- The `role` is one of `{instructor, student, TA, auditor}`. A person may have nore than one `role` in a course.
- The `start_date` must be before the `end_date`.

Put, execute and test your DDL in the code cells below. You can explain assumptions and changes in the markdown cell that precedes the code cells.

Answer

Place you explanatory notes on design choices and assumption in this markdown cell.

having year as a UNIQUE attribute on section will not work since multiple sections of a course may occur in the same year. same situation with semester and course_no. Hence I removed the unique attribute here

Start date and end date are set as not null

renamed to course2 and section2 to avoid conflict with the book DBs renamed 'student_UNI' to 'person_UNI' since it may contain UNI of instructors as well

Please place and execute your SQL statement below.

In [454...

```
%%sql
drop table person_section;
drop table person;
drop table section2;


drop table course2;
```

```
 * mysql+pymysql://root:***@localhost?local_infile=1
0 rows affected.
0 rows affected.
0 rows affected.
0 rows affected.
```

Out[454...

```
[]
```

In [455...

```
%%sql
/*
    DDL statements.
*/


create table person (
    uni VARCHAR(10) PRIMARY KEY,
    last_name VARCHAR(100) NOT NULL,
    first_name VARCHAR(100) NOT NULL,
    kind ENUM('Student', 'Faculty', 'Staff') NOT NULL,
    preferred_email VARCHAR(100) UNIQUE,
    default_email VARCHAR(100) NOT NULL UNIQUE,
    CONSTRAINT valid_default_email CHECK (default_email = CONCAT(uni, '@columbia.edu'))
);

create table course2 (
    dept_code char(4) NOT NULL CHECK (dept_code REGEXP '^\[A-Za-z\]\{4\}$'),
    faculty_code ENUM('W', 'C', 'E', 'B', 'G') NOT NULL,
    course_no char(4) NOT NULL CHECK (course_no REGEXP '^\[1-9\]\[0-9\]\{3\}$'),
    course_title varchar(100) NOT NULL,
    description TEXT,
    PRIMARY KEY (dept_code, faculty_code, course_no),
    full_course_no VARCHAR(10) NOT NULL UNIQUE,
    CONSTRAINT valid_full_course_no CHECK (full_course_no = CONCAT(dept_code, faculty_code, course_no))
);
```

```
create table section2 (
    call_no CHAR(5) PRIMARY KEY CHECK (call_no REGEXP '^\[0-9\]\{5\}$'),
    course_no VARCHAR(10) NOT NULL REFERENCES course2(full_course_no),
    section_no CHAR(3) NOT NULL CHECK (section_no REGEXP '^(V\[0-9\]\{2\}|\[0-9\]\{3\})$'),
    semester INT NOT NULL CHECK (semester > 0),
    year INT NOT NULL,
    section_key VARCHAR(20) NOT NULL UNIQUE,
    CONSTRAINT valid_section_key CHECK (section_key = CONCAT(course_no, '_', section_no, '_', semester, '_', year))
);

create table person_section (
    person_UNI VARCHAR(10) NOT NULL,
    callno CHAR(5) NOT NULL,
    role ENUM('instructor', 'student', 'TA', 'auditor') NOT NULL,
    start_date DATE NOT NULL,
    end_date DATE NOT NULL,
    CONSTRAINT valid_dates CHECK (start_date < end_date),
    PRIMARY KEY (person_UNI, callno, role),
    FOREIGN KEY (person_UNI) REFERENCES person(uni),
    FOREIGN KEY (callno) REFERENCES section2(call_no)
);
```

```
 * mysql+pymysql://root:***@localhost?local_infile=1
0 rows affected.
0 rows affected.
0 rows affected.
0 rows affected.
```

Out[455… []

Place some `SELECT` and `INSERT` SQL statements below that demonstrate the correctness of your schema implementation. You will likely need more than 3 tests. When you ask me how many tests you should write, I am going to respond, "Really? You need to do enough tests to show that your DDL is correct."

In [456…

```
%%sql
/*
    Inserting a valid entry works
*/
insert into person values ('dff9', 'Ferguson', 'Donald', 'Faculty', 'donald.ferguson@cs.columbia.edu', 'dff9@columbia.edu');
insert into person values ('bm3027', 'Ma', 'Brian', 'Student', null, 'bm3027@columbia.edu');
```

```
 * mysql+pymysql://root:***@localhost?local_infile=1
1 rows affected.
1 rows affected.
```

Out[456… []

In [457…

```
%%sql
/*
    Inserting an invalid default email leads to violation
    using a default email with a different uni value
*/
insert into person values ('abcd', 'Ma2', 'Brian2', 'Student', null, 'bmaaa3027@columbia.edu');
```

```
 * mysql+pymysql://root:***@localhost?local_infile=1
(pymysql.err.OperationalError) (3819, "Check constraint 'valid_default_email' is violated.")
[SQL: /*
    Inserting an invalid default email leads to violation
    using a default email with a different uni value
*/
insert into person values ('abcd', 'Ma2', 'Brian2', 'Student', null, 'bmaaa3027@columbia.edu');]
(Background on this error at: https://sqlalche.me/e/20/e3q8)
```

In [458…

```
%%sql
/*
    Uniqueness constraint cannot be violated
*/
insert into person values ('bm3027', 'Ma', 'Brian', 'Student', null, 'bm3027@columbia.edu');
```

```
 * mysql+pymysql://root:***@localhost?local_infile=1
(pymysql.err.IntegrityError) (1062, "Duplicate entry 'bm3027' for key 'person.PRIMARY'")
[SQL: /*
    Uniqueness constraint cannot be violated
*/
insert into person values ('bm3027', 'Ma', 'Brian', 'Student', null, 'bm3027@columbia.edu');]
(Background on this error at: https://sqlalche.me/e/20/gkpj)
```

In [459…

```
%%sql
/*
    valid entry inserts into course2
*/
insert into course2 values ('COMS', 'W', '4111', 'Introduction to Databases', 'OMG! This class is terrifying.', 'COMSW4111');
```

```
 * mysql+pymysql://root:***@localhost?local_infile=1
1 rows affected.
```

Out[459…

```
[]
```

In [460…

```
%%sql
/*
    ENUM constraint of faculty_code of course2 cannot be violated
*/
insert into course2 values ('COMS', 'X', '4112', 'Introduction to Databases 2', 'OMG! This class is terrifying.', 'COMSW4112');
```

```
 * mysql+pymysql://root:***@localhost?local_infile=1
(pymysql.err.DataError) (1265, "Data truncated for column 'faculty_code' at row 1")
[SQL: /*
    ENUM constraint of faculty_code of course2 cannot be violated
*/
insert into course2 values ('COMS', 'X', '4112', 'Introduction to Databases 2', 'OMG! This class is terrifying.', 'COMSW4112');]
(Background on this error at: https://sqlalche.me/e/20/9h9h)
```

In [461…

```
%%sql
/*
    section code constraint of section_2 cannot be violated
    composite full_course_no must be generated from corresponding attributes
*/
insert into section2 values ('11969', 'COMSW4111', '001', 1, 2025, 'COMSW4111_001_1_2025');
insert into section2 values ('31151', 'COMSW4111', 'V02', 1, 2025, 'COMSW4111_V02_1_2025');
insert into section2 values ('31152', 'COMSW4111', '002', 1, 2025, 'COMSW4111_002_1_2025');
insert into section2 values ('31153', 'COMSW4112', 'V02', 1, 2025, 'XXXXXXXX');
```

```
 * mysql+pymysql://root:***@localhost?local_infile=1
1 rows affected.
1 rows affected.
1 rows affected.
(pymysql.err.OperationalError) (3819, "Check constraint 'valid_section_key' is violated.")
[SQL: insert into section2 values ('31153', 'COMSW4112', 'V02', 1, 2025, 'XXXXXXXX');]
(Background on this error at: https://sqlalche.me/e/20/e3q8)
```

In [462…

```
%%sql
/*
    person_section reference constraints work
    start date cannot be after end date
*/
insert into person_section values ('dff9', '11969', 'instructor', 20250125, 20250502);
insert into person_section values ('invalid', '11969', 'instructor', 20250125, 20250502);
```

```
 * mysql+pymysql://root:***@localhost?local_infile=1
1 rows affected.
(pymysql.err.IntegrityError) (1452, 'Cannot add or update a child row: a foreign key constraint fails (`db_book`.`person_section`, CONSTRAIN
T `person_section_ibfk_1` FOREIGN KEY (`person_UNI`) REFERENCES `person` (`uni`))')
[SQL: insert into person_section values ('invalid', '11969', 'instructor', 20250125, 20250502);]
(Background on this error at: https://sqlalche.me/e/20/gkpj)
```

In [463…

```
%%sql
/*
    person_section start date cannot be after end date
*/
insert into person_section values ('dff9', '11969', 'student', 20250502, 20250125);
```

```
 * mysql+pymysql://root:***@localhost?local_infile=1
(pymysql.err.OperationalError) (3819, "Check constraint 'valid_dates' is violated.")
[SQL: /*
    person_section start date cannot be after end date
*/
insert into person_section values ('dff9', '11969', 'student', 20250502, 20250125);]
(Background on this error at: https://sqlalche.me/e/20/e3q8)
```

SQL DML

Question

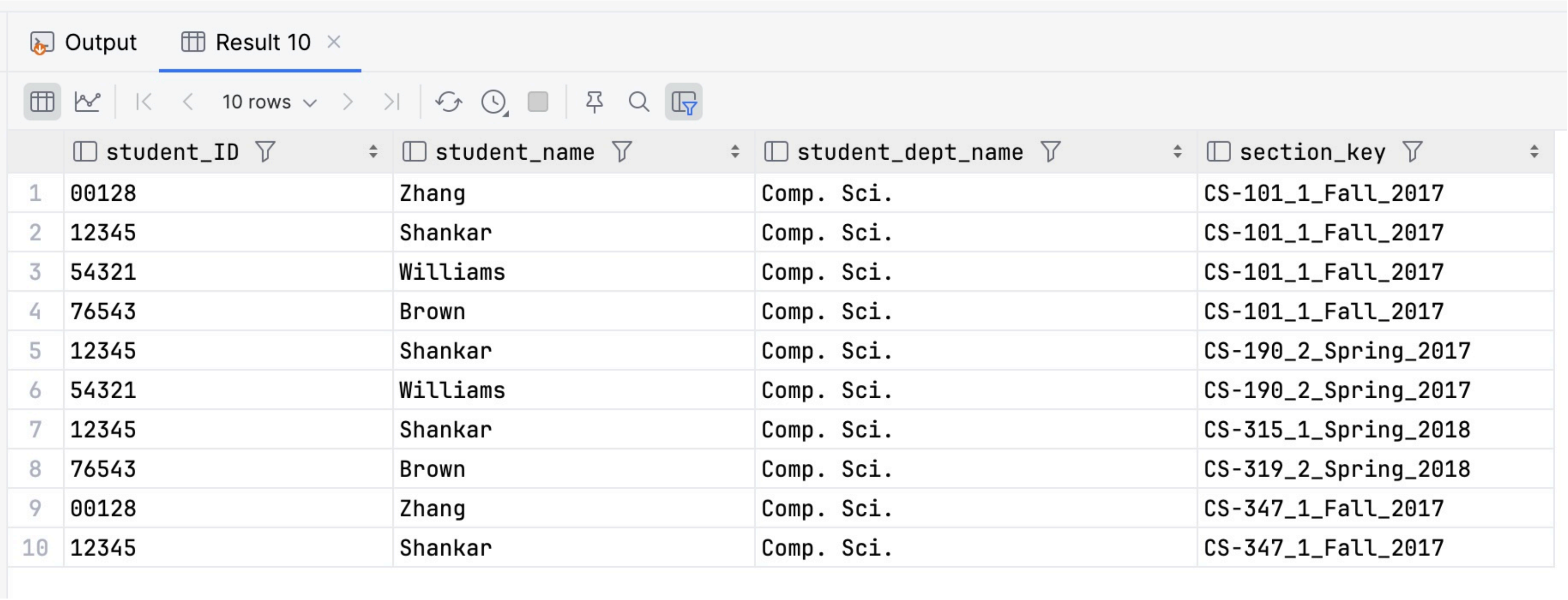
Write an SQL query that uses subqueries and does not use `JOIN` to produce a table of the form:

- `student_id`
- `student_name`
- `student_dept_name`
- `section_key`, which is a concatenation of `course_id, sec_id, semester, year` and uses `_` as the delimeter.

The result should only contain students in the `'Comp. Sci.'` department.

You should be able to figure this out from the description and examining the `db_book` data you installed. But, to simplify:

1. Use the tables `takes` and `student`.
2. The result of my implementation is below.



	student_ID	student_name	student_dept_name	section_key
1	00128	Zhang	Comp. Sci.	CS-101_1_Fall_2017
2	12345	Shankar	Comp. Sci.	CS-101_1_Fall_2017
3	54321	Williams	Comp. Sci.	CS-101_1_Fall_2017
4	76543	Brown	Comp. Sci.	CS-101_1_Fall_2017
5	12345	Shankar	Comp. Sci.	CS-190_2_Spring_2017
6	54321	Williams	Comp. Sci.	CS-190_2_Spring_2017
7	12345	Shankar	Comp. Sci.	CS-315_1_Spring_2018
8	76543	Brown	Comp. Sci.	CS-319_2_Spring_2018
9	00128	Zhang	Comp. Sci.	CS-347_1_Fall_2017
10	12345	Shankar	Comp. Sci.	CS-347_1_Fall_2017

Query Result

Answer

In [119…

```
%%sql
/*
    Write and execute your answer.
*/
select * from
(select
    (select student.ID from student where student.ID=takes.ID) as student_ID,
    (select student.name from student where student.ID=takes.ID) as student_name,
    (select student.dept_name from student where student.ID=takes.ID) as student_dept_name,
    CONCAT(course_id, '_', sec_id, '_', semester, '_', year) as section_key
from takes) as newTable where student_dept_name='Comp. Sci.'
```

```
 * mysql+pymysql://root:***@localhost?local_infile=1
10 rows affected.
```

Out[119…

student_ID	student_name	student_dept_name	section_key
00128	Zhang	Comp. Sci.	CS-101_1_Fall_2017
12345	Shankar	Comp. Sci.	CS-101_1_Fall_2017
54321	Williams	Comp. Sci.	CS-101_1_Fall_2017
76543	Brown	Comp. Sci.	CS-101_1_Fall_2017
12345	Shankar	Comp. Sci.	CS-190_2_Spring_2017
54321	Williams	Comp. Sci.	CS-190_2_Spring_2017
12345	Shankar	Comp. Sci.	CS-315_1_Spring_2018
76543	Brown	Comp. Sci.	CS-319_2_Spring_2018
00128	Zhang	Comp. Sci.	CS-347_1_Fall_2017
12345	Shankar	Comp. Sci.	CS-347_1_Fall_2017

In [ ]: