F24-W4111-03: Introduction to Databases: Homework 1, Part B

Submission Instructions

Note to TAs: Please complete this information, create GradeScope entries, etc.

Environment Setup

This section tests your environment for HW1B.

If you successfuly completed HW0, you should not have any problems.

Please make sure you set your MySQL user id and password correctly.

In [1]:
```
#%pip install pandas
import pandas
```

In [4]:
```
#%pip install sqlalchemy
import sqlalchemy
```

In [5]:
```
#%pip install pymysql
import pymysql
```

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

In [6]:
```
#%pip install ipython-sql
%load_ext sql
```

In [11]:
```
%sql mysql+pymysql://root:gjc20021013@localhost
```

In [13]:
```
engine = sqlalchemy.create_engine("mysql+pymysql://root:gjc20021013@localhos
```

Entity Relationship Modeling

Top-Down Modeling

The ability to prduce an ER diagram from a "human" description of the data model is an import skill. In this process, you may have to make and document assumptions or explain decisions. There is no single, correct answer. As long as your assumptions and decisions are reasonable, and your model accurately reflects requirements and decisions, your model answer is "correct."

In this scenario, there are four entity types/entity sets:

1. `Person(id, last_name, first_name, middle_name, created_timestamp, last_modified_timestamp)` : Basic information about a person. The type has properties/attributes:
   - `id` uniquely indentifies the `Person`
   - `last_name`
   - `first_name`
   - `middle_name`
   - `created_timestamp` : When the the entity was created for the first time.
   - `last_modified_timestamp` : The last time the entity's information changed.
2. `Contact_Information(contact_type, contact_value)` : Represents a mechanism for contacting a person.
   - `id` : A unique ID for the `Contact_Information` .
   - `contact_type` : Indicates the type of contact, e.g. "primary phone," "email," etc.
   - `contact_value:` The value for the contact. This is simply a text string for both types of contact. For example, "bilbo.baggins@shire.org" or "+1 212-555-1212."
3. `Order(id, product_name, order_date, description)` : Represents someone having placed an order to purchase something. Order has the properties:
   - `id` : Uniquely identifies the `Order`
   - `product_name` : The name of the product, e.g. "Strawbery Poptarts," "Cross Pen."
   - `order_date` : The date the order was placed
   - `description` : A text description of the order
4. `Comment(id, comment, comment_timestamp)` : Represent a user's comment on an order. Comment has three properties:
   - `id` : Uniquely identifies the `Comment`
   - `comment` : Text of the comment
   - `comment_timestamp` : Timestamp when the comment was made.

The model has the following relationships/entity sets:

- `Person-Comment` is a relationship the represents the fact that the `Person` made the `Comment` . A `Person` may make many `Comments` , but a `Comment` is made by exactly one user.

- `Order-Comment` associates the `Comment` with the `Order.` There may be many `Comments` on an `Order` but a `Comment` has one `Order.`
- `Person-Contact-Info` is between `Person` and `Contact-Info.` A `Person` may have multiple `Contact-Info` entries. A `Contact-Info` relates to exactly one `Person`.

The model must represent the fact that `Contact-Info` is valid between a start timestamp and end timestamp.

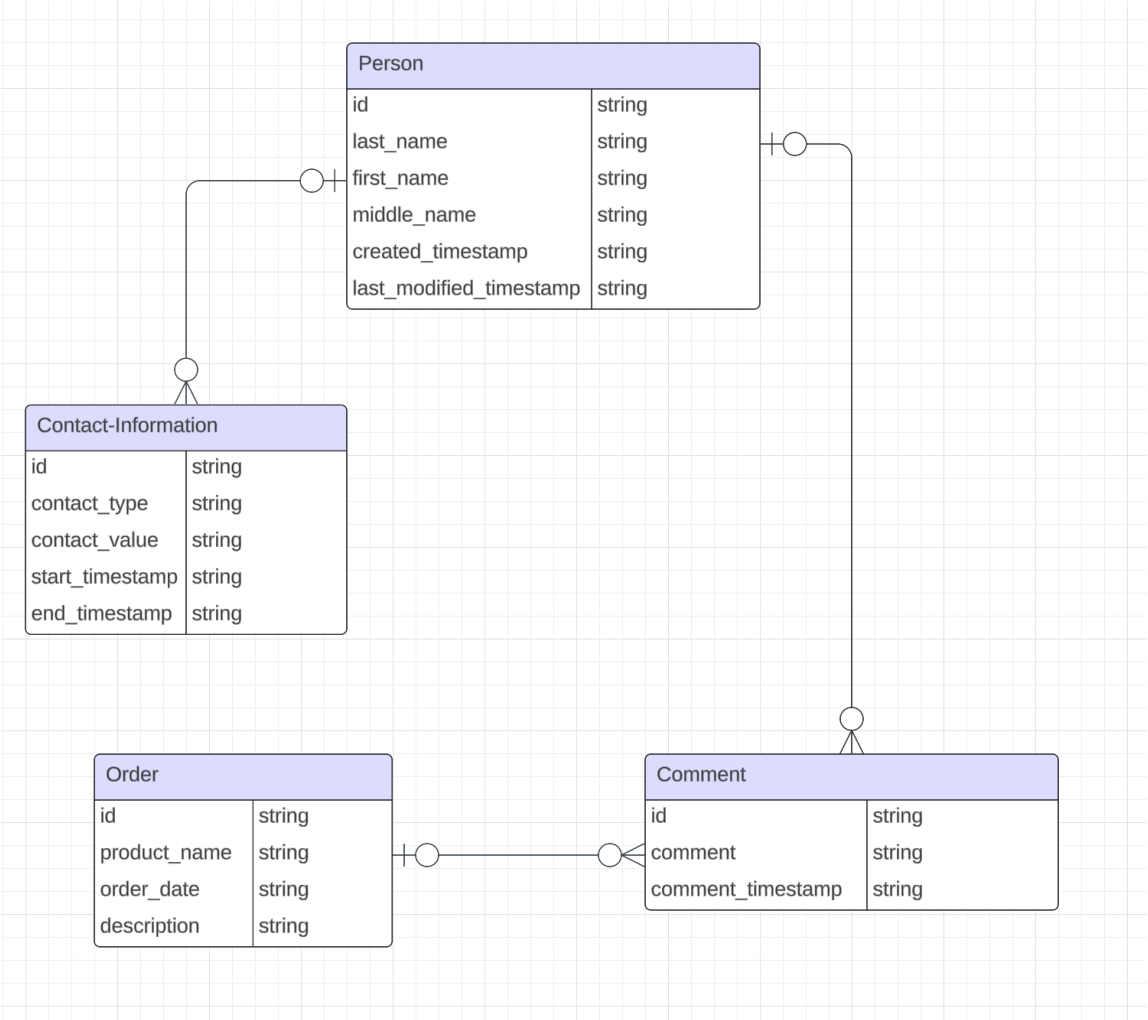
The system never deletes any information.

You must create a Crow's Foot Notation *logical model* that is your model that satisfies the requirements. You may have to add unspecified attributes to entity types. You can add comments and notes.

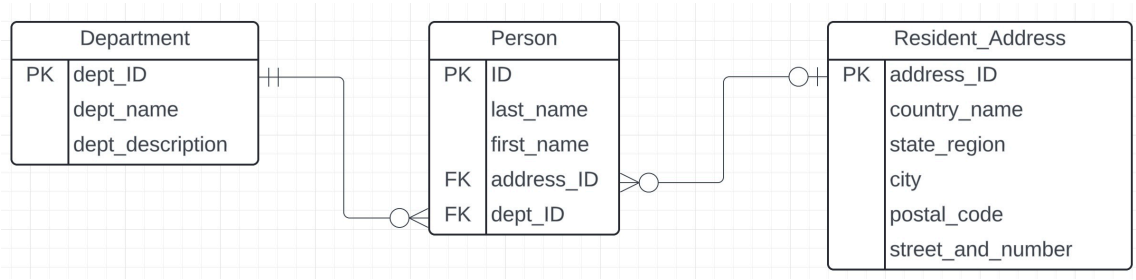
Show your diagram below. You can add notes to your diagram or add explanatory text. You can take a screenshot of your diagram and include below. The "Implement ER Diagram" question has an example of embedding an image in the notebook.

There is no single correct answer.

Digram:



Implement ER Diagram



Write SQL DDL that creates the tables and relationships in the preceding diagram

You can pick `VARCHAR(32)` for the type of each column.

You must specify keys and foreign keys.

Create a new database that you name `hw1b_<uni>` and replace `<uni>` with your UNI. For example, mine would be `hw1b_dff9.`

You must enter and successfully execute your SQL in the code cell below.

In [12]:
```
%%sql

/* Your create and alter table statements. */
DROP DATABASE IF EXISTS hw1b_jg4874;
CREATE DATABASE hw1b_jg4874;

USE hw1b_jg4874;

CREATE TABLE department (
    dept_id VARCHAR(32) PRIMARY KEY,
    dept_name VARCHAR(32),
    dept_description VARCHAR(32)
);

CREATE TABLE Resident_Address (
    address_id VARCHAR(32) PRIMARY KEY,
    country_name VARCHAR(32),
    state_region VARCHAR(32),
    city VARCHAR(32),
    postal_code VARCHAR(32),
    street_and_number VARCHAR(32)
);

CREATE TABLE Person (
    ID VARCHAR(32) PRIMARY KEY,
    last_name VARCHAR(32),
    first_name VARCHAR(32),
    address_id VARCHAR(32),
    dept_id VARCHAR(32),
    FOREIGN KEY (address_id) REFERENCES Resident_Address(address_id),
    FOREIGN KEY (dept_id) REFERENCES department(dept_id)
);
```

```
 * mysql+pymysql://root:***@localhost
3 rows affected.
1 rows affected.
0 rows affected.
0 rows affected.
0 rows affected.
0 rows affected.
```

Out[12]: []

Relational Algebra

You will use the Relax calulator and the schema associated with the text book for this question.

https://dbis-uibk.github.io/relax/calc/gist/4f7866c17624ca9dfa85ed2482078be8/relax-silberschatz-english.txt/0

Problem 1

Write a relational algebra expression that produces a result table with the following format:

`(student_id, student_name, course_title, course_id, sec_id, semester, year, instructor_id, instructor_name)`

- `student_id` is a student's ID (`student.ID`)
- `student_name` is a student's name (`student.name`)
- `course_title` (`course.title`)
- The following columns are common to `section, takes, teaches:`
  - `course_id`
  - `sec_id`
  - `semester`
  - `year`
- `instructor_id` is an instructor's ID (`instructor.ID`)
- `instructor_name` is an instructor's name (`instructor.name`)

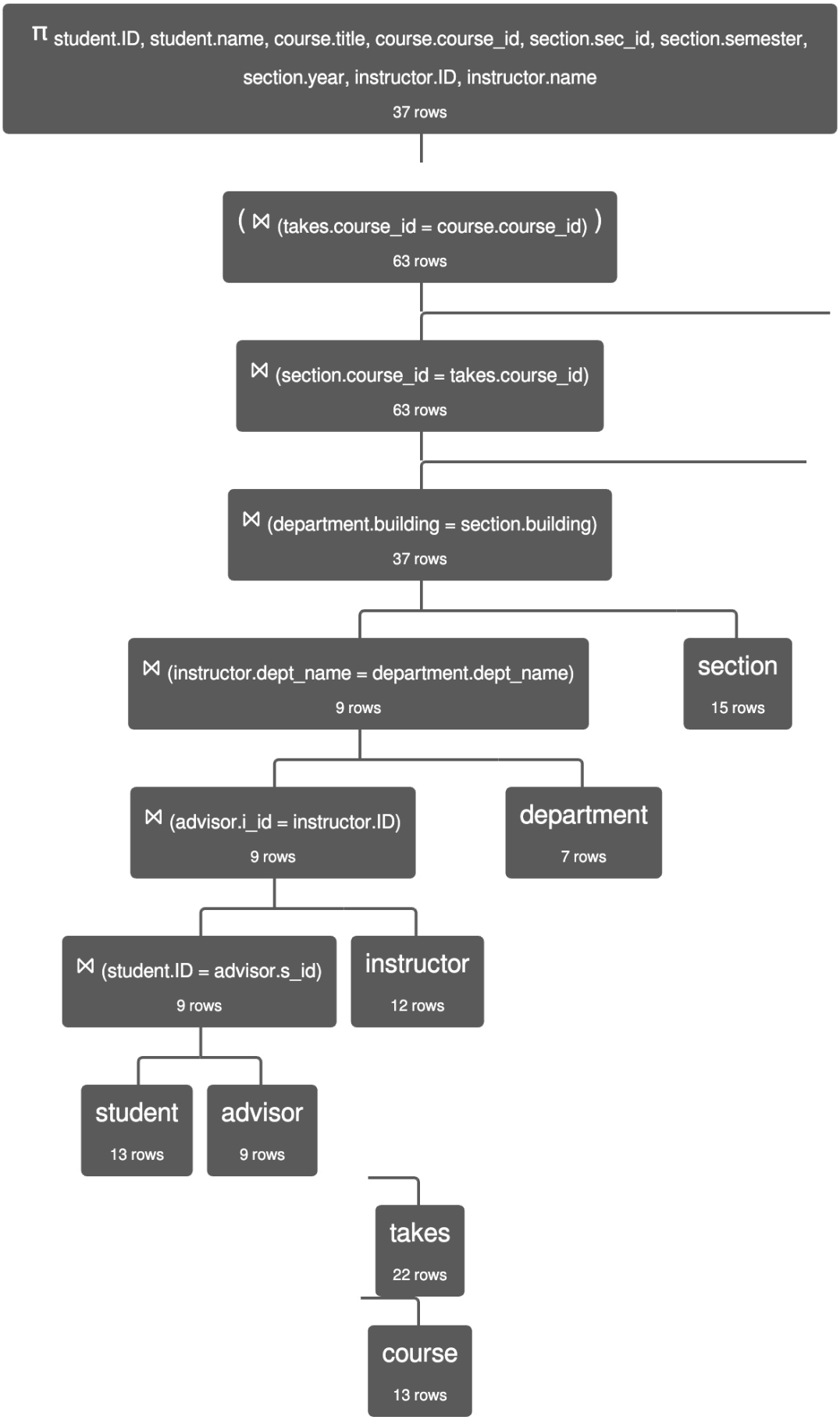
This derived relation represents student that took a section and the instructor taught the section.

Cut and paste your query in the markdown cell below.

Past relational algebra here. The following is an example of pasting a relational algebra expression. Replace the following with your expression.

```
 π student.ID, student.name, course.title, course.course_id,
section.sec_id, section.semester, section.year,
instructor.ID, instructor.name
(
    student ⋈ (student.ID = advisor.s_id) advisor ⋈
(advisor.i_id = instructor.ID) instructor ⋈
                (instructor.dept_name = department.dept_name)
department ⋈
    (department.building = section.building) section ⋈
    (section.course_id = takes.course_id) takes ⋈
    (takes.course_id = course.course_id) course
 )
```

Execute your query on the Relax calculator and show an image of the first page of your result below. The following shows an example of the format of the answer applied to the above example.


π student.ID, student.name, course.title, course.course_id, section.sec_id, section.semester, section.year, instructor.ID, instructor.name
37 rows
(⋈ (takes.course_id = course.course_id))
63 rows
⋈ (section.course_id = takes.course_id)
63 rows
⋈ (department.building = section.building)
37 rows
⋈ (instructor.dept_name = department.dept_name)
9 rows
section
15 rows
⋈ (advisor.i_id = instructor.ID)
9 rows
department
7 rows
⋈ (student.ID = advisor.s_id)
9 rows
instructor
12 rows
student
13 rows
advisor
9 rows
takes
22 rows
course
13 rows

$$\pi \text{ student.ID, student.name, course.title, course.course\_id, section.sec\_id, section.semester, section.year, instructor.ID, instructor.name } (((((\text{ student } \bowtie \text{ (student.ID = advisor.s\_id) advisor }) \bowtie \text{ (advisor.i\_id = instructor.ID) instructor }) \bowtie \text{ (instructor.dept\_name = department.dept\_name) department }) \bowtie \text{ (department.building = section.building) section }) \bowtie \text{ (section.course\_id = takes.course\_id) takes }) \bowtie \text{ (takes.course\_id = course.course\_id) course })$$

Execution time: 4 ms

result

student.ID	student.name	course.title	course.course_id	section.sec_id	section.semester	section.year	instructor.ID	instructor.name
128	Zhang	Game Design	CS-190	1	Spring	2009	45565	Katz
128	Zhang	Game Design	CS-190	2	Spring	2009	45565	Katz
128	Zhang	Image Processing	CS-319	2	Spring	2010	45565	Katz
128	Zhang	Database System Concepts	CS-347	1	Fall	2009	45565	Katz
128	Zhang	Intro. to Digital Systems	EE-181	1	Spring	2009	45565	Katz
12345	Shankar	Game Design	CS-190	1	Spring	2009	10101	Srinivasan
12345	Shankar	Game Design	CS-190	2	Spring	2009	10101	Srinivasan
12345	Shankar	Image Processing	CS-319	2	Spring	2010	10101	Srinivasan
12345	Shankar	Database System Concepts	CS-347	1	Fall	2009	10101	Srinivasan
12345	Shankar	Intro. to Digital Systems	EE-181	1	Spring	2009	10101	Srinivasan
23121	Chavez	Intro. to Biology	BIO-101	1	Summer	2009	76543	Singh
23121	Chavez	Genetics	BIO-301	1	Summer	2010	76543	Singh
23121	Chavez	World History	HIS-351	1	Spring	2010	76543	Singh
44553	Peltier	Robotics	CS-315	1	Spring	2010	22222	Einstein
44553	Peltier	Image Processing	CS-319	1	Spring	2010	22222	Einstein
44553	Peltier	Physical Principles	PHY-101	1	Fall	2009	22222	Einstein
45678	Levy	Robotics	CS-315	1	Spring	2010	22222	Einstein
45678	Levy	Image Processing	CS-319	1	Spring	2010	22222	Einstein
45678	Levy	Physical Principles	PHY-101	1	Fall	2009	22222	Einstein
76543	Brown	Game Design	CS-190	1	Spring	2009	45565	Katz
76543	Brown	Game Design	CS-190	2	Spring	2009	45565	Katz
76543	Brown	Image Processing	CS-319	2	Spring	2010	45565	Katz
76543	Brown	Database System Concepts	CS-347	1	Fall	2009	45565	Katz
76543	Brown	Intro. to Digital Systems	EE-181	1	Spring	2009	45565	Katz
76653	Aoi	Game Design	CS-190	1	Spring	2009	98345	Kim
76653	Aoi	Game Design	CS-190	2	Spring	2009	98345	Kim
76653	Aoi	Image Processing	CS-319	2	Spring	2010	98345	Kim
76653	Aoi	Database System Concepts	CS-347	1	Fall	2009	98345	Kim
76653	Aoi	Intro. to Digital Systems	EE-181	1	Spring	2009	98345	Kim
98765	Bourikas	Game Design	CS-190	1	Spring	2009	98345	Kim
98765	Bourikas	Game Design	CS-190	2	Spring	2009	98345	Kim
98765	Bourikas	Image Processing	CS-319	2	Spring	2010	98345	Kim
98765	Bourikas	Database System Concepts	CS-347	1	Fall	2009	98345	Kim
98765	Bourikas	Intro. to Digital Systems	EE-181	1	Spring	2009	98345	Kim
98988	Tanaka	Robotics	CS-315	1	Spring	2010	76766	Crick
98988	Tanaka	Image Processing	CS-319	1	Spring	2010	76766	Crick
98988	Tanaka	Physical Principles	PHY-101	1	Fall	2009	76766	Crick

---

Problem 2

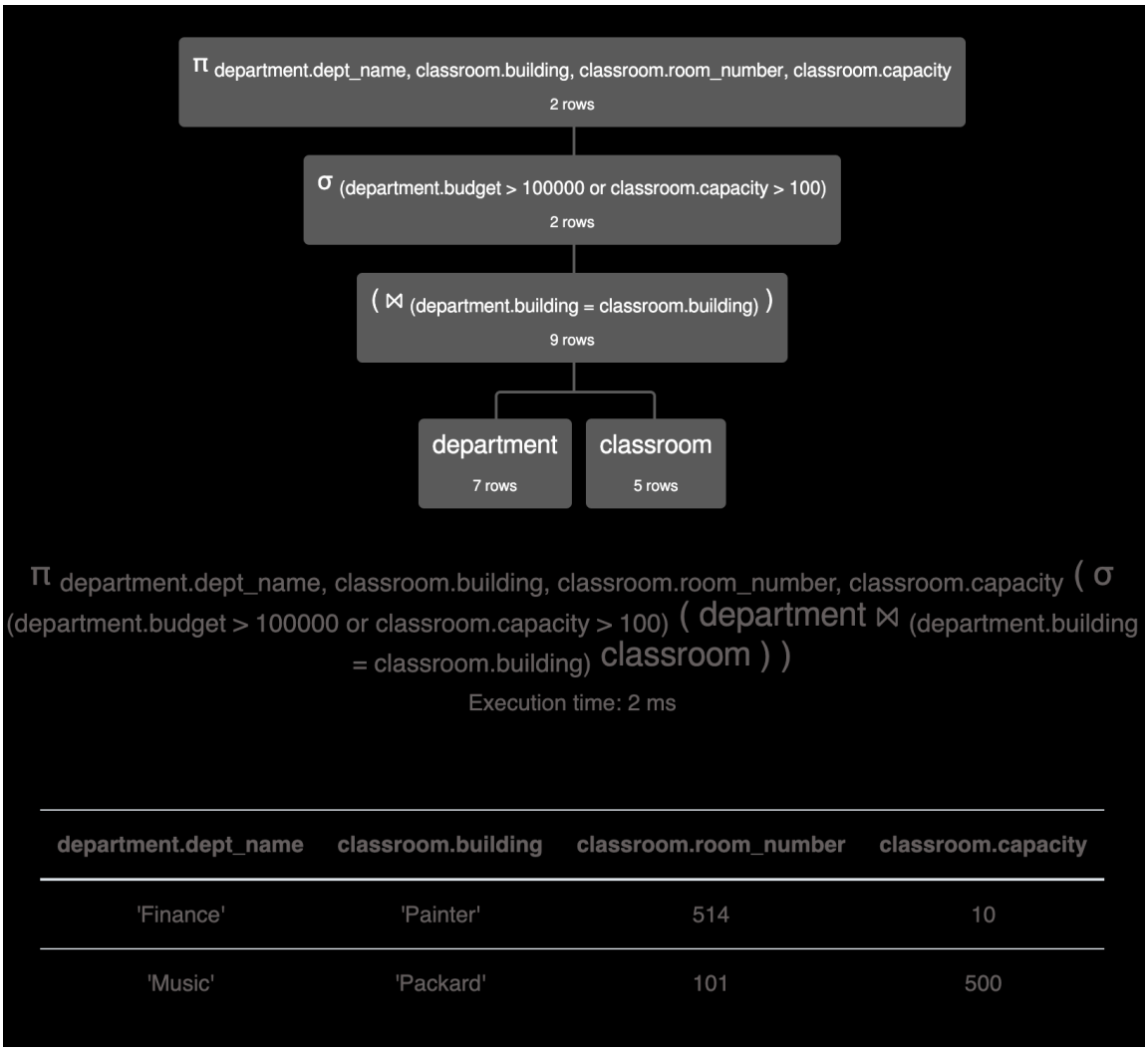
Write a relational algebra expression that produces a result table with the following format:

```
(dept_name, building, classroom, capacity)
```

This contains tuples where:

- The department is in the building, e.g. there is a tuple in `department` that has the `dept_name` and `building`.
- The `classroom` is in the `building`.
- The result ONLY contains entries for which the department's `budget` is greather than 100,000 or the classroom's `capacity` is greater than 100.

```
π department.dept_name, classroom.building,
classroom.room_number, classroom.capacity
(       σ(department.budget > 100000 v classroom.capacity >
100)
        (department ⋈ (department.building =
classroom.building) classroom)
)
```



department.dept_name	classroom.building	classroom.room_number	classroom.capacity
'Finance'	'Painter'	514	10
'Music'	'Packard'	101	500

SQL

Use the database that is associated with the recommended textbook for these questions. You loaded this in HW0.

Problem 1

Write a SQL query that produces a table of the form `(student_id, student_name, advisor_id, advisor_name)` that shows the ID and name of a student combined with their advisor. Only include rows where both the student and the advisor are in the `Comp. Sci.` and the student has at least 50 total credits.

Execute your SQL below.

In [16]:
```
%%sql
use db_book;
select
  student.ID as student_id,
  student.name as student_name,
  instructor.ID as advisor_id,
  instructor.name as advisor_name
from
  student
join
  advisor on student.ID = advisor.s_ID
join
  instructor on advisor.i_ID = instructor.ID
where
  student.dept_name = 'Comp. Sci.'
  and instructor.dept_name = 'Comp. Sci.'
  and student.tot_cred >= 50;
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
2 rows affected.
```

Out[16]:

student_id	student_name	advisor_id	advisor_name
00128	Zhang	45565	Katz
76543	Brown	45565	Katz

Problem 2

Consider the following query.

In [17]:
```
%%sql
use db_book;
select * from db_book.student where dept_name='Comp. Sci.'
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
4 rows affected.
```

Out[17]:

ID	name	dept_name	tot_cred
00128	Zhang	Comp. Sci.	102
12345	Shankar	Comp. Sci.	32
54321	Williams	Comp. Sci.	54
76543	Brown	Comp. Sci.	58

The following table makes a copy of the student table.

In [18]:
```
%%sql
use db_book;
create table student_hw1b as select * from student
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
13 rows affected.
```

Out[18]: []

In [19]:
```
%%sql
use db_book;
select * from student_hw1b where dept_name='Comp. Sci.'
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
4 rows affected.
```

Out[19]:

ID	name	dept_name	tot_cred
00128	Zhang	Comp. Sci.	102
12345	Shankar	Comp. Sci.	32
54321	Williams	Comp. Sci.	54
76543	Brown	Comp. Sci.	58

We are now going to make some changes to `student_hw1b`

Write and execute a SQL statement that changes Willliams tot_cred to 75.

In [20]:
```
%%sql
use db_book;
update student_hw1b set tot_cred='75' where name='Williams'
```

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

Out[20]: []

Show the result.

In [21]:
```
%sql select * from student_hw1b where dept_name='Comp. Sci.'
```

```
 * mysql+pymysql://root:***@localhost
4 rows affected.
```

Out[21]:

ID	name	dept_name	tot_cred
00128	Zhang	Comp. Sci.	102
12345	Shankar	Comp. Sci.	32
54321	Williams	Comp. Sci.	75
76543	Brown	Comp. Sci.	58

Write a SQL statement that deletes Williams from the `student_hw1b` table and execute in the cell below.

In [22]:
```
%%sql
use db_book;
delete from student_hw1b where name='Williams'
```

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

Out[22]: []

Show the resulting table.

In [23]:
```
%sql select * from student_hw1b where dept_name='Comp. Sci.'
```

```
 * mysql+pymysql://root:***@localhost
3 rows affected.
```

Out[23]:

ID	name	dept_name	tot_cred
00128	Zhang	Comp. Sci.	102
12345	Shankar	Comp. Sci.	32
76543	Brown	Comp. Sci.	58

Write and execute SQL statement that puts the original data for Williams back in the table.

In [24]:
```
%%sql
insert into student_hw1b (ID, name, dept_name, tot_cred)
  values ('54321', 'Williams', 'Comp. Sci.', 54)
```

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

Out[24]: []

Show the table.

In [25]:
```
%sql select * from student_hw1b where dept_name='Comp. Sci.'
```

```
 * mysql+pymysql://root:***@localhost
4 rows affected.
```

Out[25]:

ID	name	dept_name	tot_cred
00128	Zhang	Comp. Sci.	102
12345	Shankar	Comp. Sci.	32
76543	Brown	Comp. Sci.	58
54321	Williams	Comp. Sci.	54