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

Homework 1
Introduction to Core Concepts, ER Modeling, Relational Algebra, SQL

Introduction

This notebook contains Homework 1. **Both Programming and Nonprogramming tracks should complete this homework.**

Submission Instructions

- You will submit **PDF and ZIP files** for this assignment. Gradescope will have two separate assignments for these.
- For the PDF:
  - 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.
  - **MAKE SURE YOU DON'T SUBMIT A SINGLE PAGE PDF.** Your PDF should have multiple pages.
- For the ZIP:
  - Zip a folder containing this notebook and any screenshots.
  - You may delete any unnecessary files, such as caches.

Add Student Information

In [ ]:

```python
# Print your name, uni, and track below

name = "Ava Hajratwala"
uni = "ash2261"
```

```
track = "Programming Track"

print(name)
print(uni)
print(track)
```

```
Ava Hajratwala
ash2261
Programming Track
```

Setup

SQL Magic

The `sql` extension was installed in HW0. Double check that if this cell doesn't work.

```
In [ ]: %load_ext sql
```

You may need to change the password below.

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

```
In [ ]: %sql SELECT * FROM db_book.student WHERE ID = 12345
```

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

Out[ ]:

ID	name	dept_name	tot_cred
12345	Shankar	Comp. Sci.	32

Python Libraries

```
In [ ]: from IPython.display import Image
        import pandas
```

Written Questions

Chapter 1 from the recommended textbook Database System Concepts, Seventh Edition covers general information and concepts about databases and database management systems. Lecturing on the general and background information is not a good use of precious class time. To be more efficient with class time, the chapter 1 information is a reading assignment.

Answering the written questions in HW 1, Part 1 does not require purchasing the textbook and reading the chapter. The chapter 1 slides provided by the textbook authors provide the necessary information. In some cases, students may also have to search the web or other sources to “read” the necessary information.

When answering the written questions, do not “bloviate”. The quantity of words does not correlate with the quality of the answer. We will deduct points if you are not succinct. The answers to the questions require less than five sentences or bullet points.

“If you can't explain something in a few words, try fewer.”

You may use external resources, but you should cite your sources.

W1

What is a database management system and how do relational databases organize data?

A database management system is a computer tool that creates and manages databases. A relational database organizes data in a table, where the columns are attributes and the rows are instances of the entity the table represents.

W2

Columbia University uses several applications that use databases to run the university. Examples are SSOL and CourseWorks. An alternate approach could be letting students, faculty, administrators, etc. use shared Google Sheets to create, retrieve, update, and delete information. What are some problems with the shared spread sheet approach and what functions do DMBS implement to solve the problems?

In the spreadsheet system, everyone would have equal administrative access, so students could modify the meeting time of a class, for instance. Also, even if people weren't trying to purposely sabotage the data, the spreadsheet is prone to user error: people might enter data inconsistently, so there could be misunderstandings (i.e., listing a class in Milstein as MIL, which stands for Milbank). Database management systems have authentication measures in place to make sure that only people who have the power to change things like class time can, which keeps the data consistent and safe.

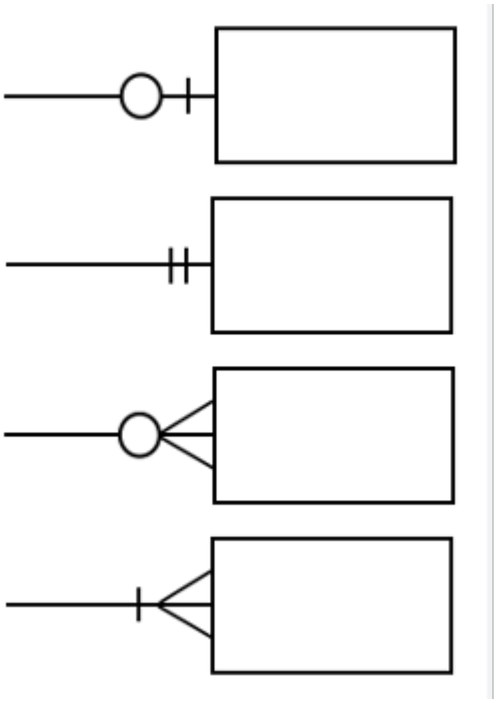
W3

Explain the differences between SQL, MySQL Server and DataGrip.

SQL is the programming language that we are using to query databases. MySQL is a database management program using a client-server model. It can make databases and queries (using the SQL language) and receives the server's response. DataGrip provides a GUI for interacting with the databases MySQL creates.

W4

Crow's Foot Notation has four endings for relationship lines. Briefly explain the meaning of each ending.



1. Zero or one
2. One and only one
3. Zero or many
4. One or many

W5

What is a primary key and why is it important?

A primary key is a unique attribute (column) that can be used to identify an entry (row) in a data table (aka relational database). We can use them in lookups if we are trying to single out a specific entry.

W6

The relational algebra is closed under the operators. Explain what this means and give an example.

Relational algebra is closed because algebraic operations acting on relational databases themselves produce new relational databases. An example of this is the select operation, which returns the database with a subset of its rows.Even if rows are missing, the result of a select operation is still a relational database, which means it is closed.

W7

Some of the Columbia University databases/applications represent the year/semester attribute of a section in the form "2023_2". The first four characters are the academic year, and the last character is the semester (1, 2, or 3). The data type for this attribute might be CHAR(6). Using this example, explain the concepts of domain and atomic domain. How is domain different from type?

Domain is the set of allowable values that an element may contain. Generally we want atomic domains, where the attributes values are indivisible. For this example, domain is {integers representing a year}_{integers 1 or 2}. This domain is not atomic because it describes two things: the semester and the year. The domain of an attribute is different than its data type because

W8

Briefly explain the difference between a database schema and database instance.

A database schema is like a class in an object oriented language, defining what attributes instances in the entity set should have. It represents the high-level structure of a database.

A database instance is a "snapshot" of the database's data at a moment in time.

W9

Briefly explain the concepts of data definition language and data manipulation language.

The data definition language of SQL lets us specify relations and their properties, like datatypes of attributes, schema, integrity constraints, and so on. The DDL has commands for defining views as well as the beginning and ends of transactions.

SQL's data manipulation language lets us query information from an existing database as well as add or remove from it.

W10

What is physical data independence?

Physical data independence is the idea that any change in the physical structure of the database should not interfere with an application that is trying to access the data.

Entity-Relationship Modeling

Overview

The ability to understand a general description of a requested data model and to transform into a more precise, specified *logical model* is one of the most important skills for using databases. SW and data engineers build applications and data models for end-users. The end-users, product managers and business managers are not SW or data modeling experts. They will express their *intent* in imprecise, text and words.

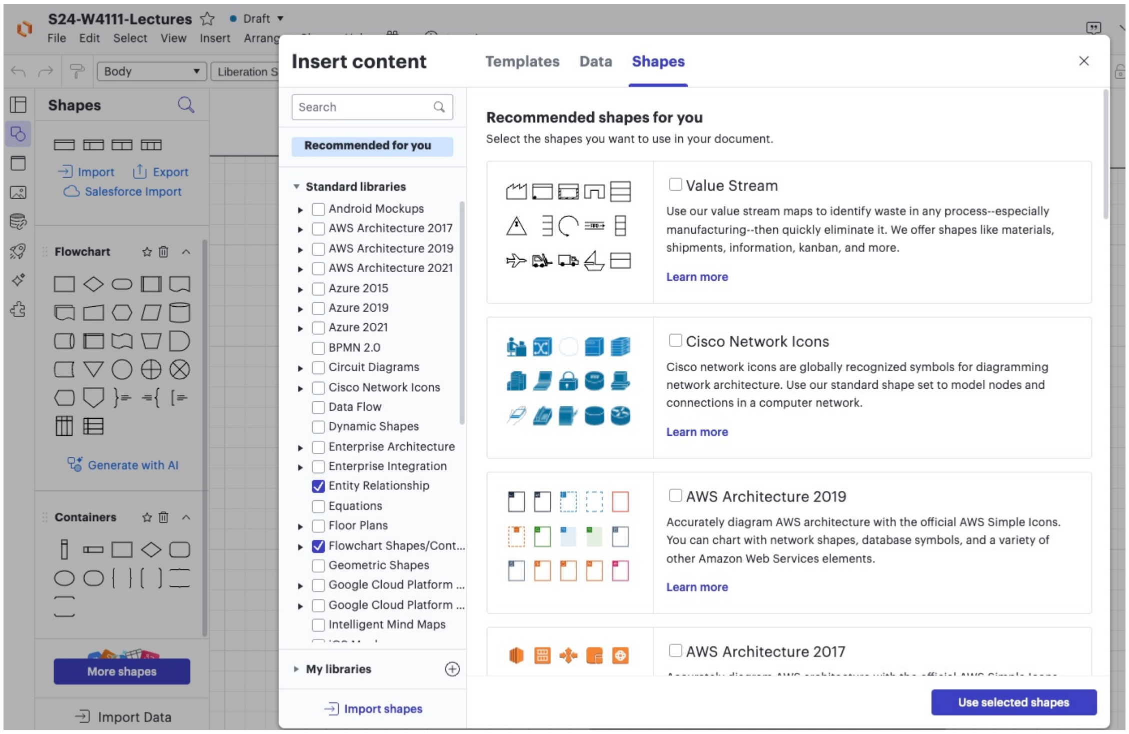
The users and business stakeholder often can understand and interact using a *conceptual model* but details like keys, foreign keys, ... are outside their scope.

In this problem, you will:

- Understand a short written description of a requested data model.
- Produce a *conceptual data model diagram* using Lucidchart.
- Produce a *logical data model diagram* using Lucidchart.

You can sign up for a free Lucidchart account. The free account provides the capabilities you will need for this course.

To draw the diagrams, you need to add the *entity relationship* shapes. Lecture 2 demonstrated how to add the shapes.



Recommended shapes for you

Select the shapes you want to use in your document.

□ Value Stream

Use our value stream maps to identify waste in any process--especially manufacturing--then quickly eliminate it. We offer shapes like materials, shipments, information, kanban, and more.

Learn more

□ Cisco Network Icons

Cisco network icons are globally recognized symbols for diagramming network architecture. Use our standard shape set to model nodes and connections in a computer network.

Learn more

□ AWS Architecture 2019

Accurately diagram AWS architecture with the official AWS Simple Icons. You can chart with network shapes, database symbols, and a variety of other Amazon Web Services elements.

Learn more

□ AWS Architecture 2017

Adding Entity Relationship Shapes

We provide a simple Lucidchart document from Lecture 2 that helps you get started. You need a Lucidchart account to access the document and diagrams.

Data Model Description

The data model represents banks, customers. employees and accouts. The model has the following entity types/sets:

1. *Customer*
2. *Employee* of the banking company
3. *Branch,* which is a location of one of the banks offices
4. *Savings Account*
5. *Checking Account*
6. *Loan*
7. *Portfolio*

Customer has the following properties:

- *customerID*
- *lastName*
- *firstName*
- *email*
- *dateOfBirth*

Employee has the following properties:

- *employeeID*
- *lastName*
- *firstName*
- *jobTitle*

Branch has the following properties:

- *branchID*
- *zipCode*

Savings Account has the following properties:

- *accountID*
- *balance*
- *interestRate*

Checking Account has the following properties:

- *accountID*
- *balance*

Loan has the following properties.

- *loanID*
- *balance*
- *interestRate*

Portfolio has the following properties:

- *portfolioID*
- *createdDate*

The data model has the following relationships:

- *Customer Branch* connects a customer and a branch. A *Customer* is connected to exactly one *Branch*. A *Branch* may have 0, 1 or many customers.
- *Employee Branch* connects an employee and a branch. An *Employee* is connected to exactly one *Branch*. A *Branch* may have 0, 1 or many associated employees.
- *Savings Account Branch*, *Checking Account Branch*, and *Loan Branch* all have the same pattern.
  - An account/loan has exactly one branch.
  - A *Branch* many have 0, 1 or many accounts/loans.
- *Savings Customer*, *Checking Customer*, *Loan Customer*, and *Portfolio Customer* follow the same pattern.
  - The account/loan has exactly one customer.
  - The customer may have 0 or 1 of each type of account.

- A *Portfolio* is related to exactly one *Customer*, exactly one *Savings Account*, exactly one *Checking Account*, and exactly one *Loan*.
- *Portfolio Advisor* relates a *Portfolio* and *Employee*. An *Employee* may be the advisor for 0, 1 or many *Portfolios*. A *Portfolio* may have at most one *Employee* advisor.

Answer

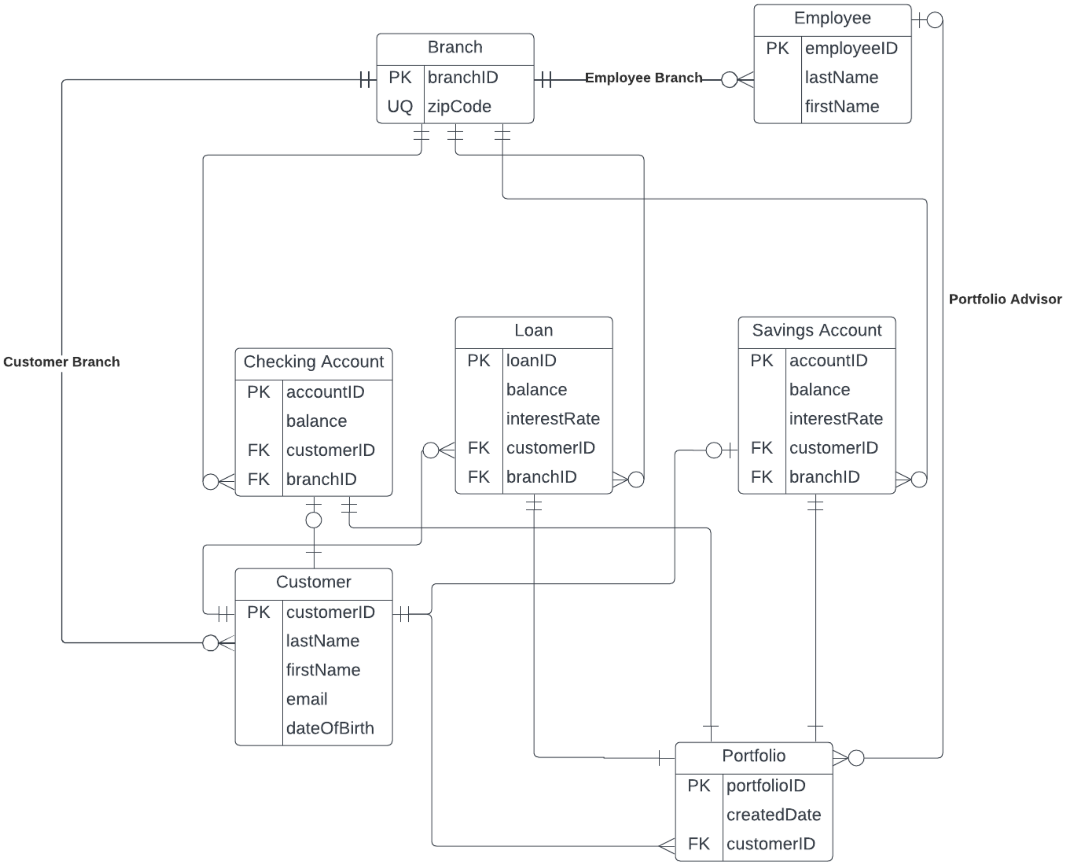
1. Place your Logical Model diagram below.
2. You *may* have to add attributes to entities to implement the model.
3. You *may* make reasonable assumptions. Please document your assumptions below. You may add comments/notes to your diagram for clarity.

Assumptions:

1. Assumption 1
2. Assumption 2
3. ... ...

ER Diagram:

Save your diagram to an image, place in the same directory as your notebook and change the file name in the HTML `img` tag in this Markdown cell.



Logical ER Diagram

Relational Algebra

R-1

The following is the SQL DDL for the `db_book.classroom` table.

```
CREATE TABLE IF NOT EXISTS db_book.classroom
(
    building    VARCHAR(15) NOT NULL,
    room_number VARCHAR(7)  NOT NULL,
    capacity    DECIMAL(4)  NULL,
    PRIMARY KEY (building, room_number)
);
```

Using the notation from the lecture slides, provide the corresponding relation schema definition.

classroom(building, room_number, capacity)

Answer Format

For the answers to the relational algebra questions, you will use the RelaX calculator with the schema associated with the book. You answer should include the algebra statement in as text and a screenshot of the execution result. Question **R0** below shows a sample of that the answer will look like.

R0

Write a relational algebra statement that produces a table of the following form:

- ID is the instructor ID
- name is the instructor name
- course_id, sec_id, semester, year of a section
- building, room_number

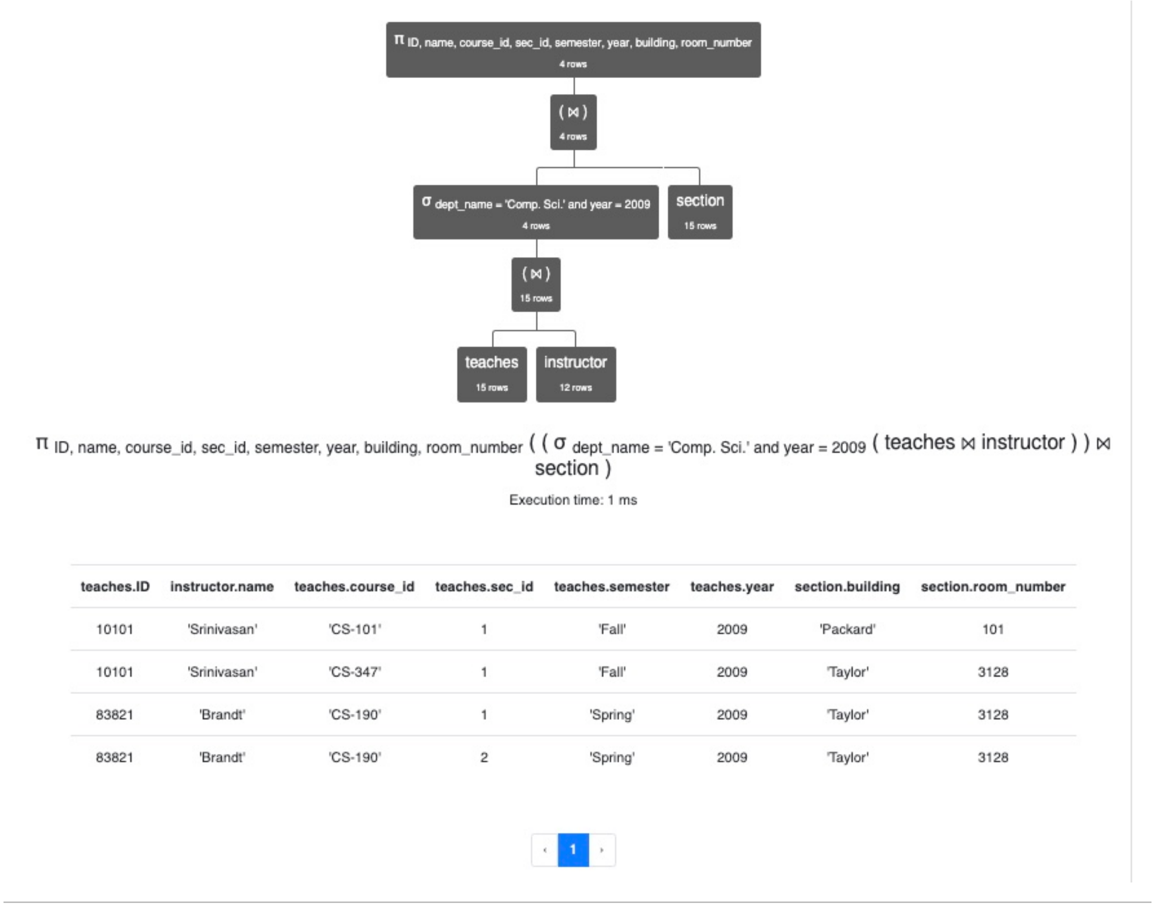
Note:

1. You will have to use the instructor, teaches and section relations
2. Your answer should only include sections taught in `Comp. Sci.` in `2009`

Algebra statement:

```
π ID, name, course_id, sec_id, semester, year, building,
room_number(
    (σ dept_name='Comp. Sci.' ∧ year=2009
        (teaches ⨝ instructor)
        )
⨝ section)
```

Execution:


π ID, name, course_id, sec_id, semester, year, building, room_number
4 rows
(⋈)
4 rows
σ dept_name = 'Comp. Sci.' and year = 2009
4 rows
section
15 rows
(⋈)
15 rows
teaches
15 rows
instructor
12 rows
‹ 1 ›

$\pi_{\text{ID, name, course\_id, sec\_id, semester, year, building, room\_number}}\ ((\sigma_{\text{dept\_name = 'Comp. Sci.' and year = 2009}}\ (\text{teaches} \bowtie \text{instructor})) \bowtie \text{section})$

Execution time: 1 ms

teaches.ID	instructor.name	teaches.course_id	teaches.sec_id	teaches.semester	teaches.year	section.building	section.room_number
10101	'Srinivasan'	'CS-101'	1	'Fall'	2009	'Packard'	101
10101	'Srinivasan'	'CS-347'	1	'Fall'	2009	'Taylor'	3128
83821	'Brandt'	'CS-190'	1	'Spring'	2009	'Taylor'	3128
83821	'Brandt'	'CS-190'	2	'Spring'	2009	'Taylor'	3128

RO Execution Result

R1

Write a relational algebra statement that produces a relation with the columns:

- `student.name`
- `student.dept_name`
- `student.tot_cred`
- `instructor.name` (the instructor that advises the student)
- `instructor.dept_name`

Only keep students who have earned more than 90 credits.

Note:

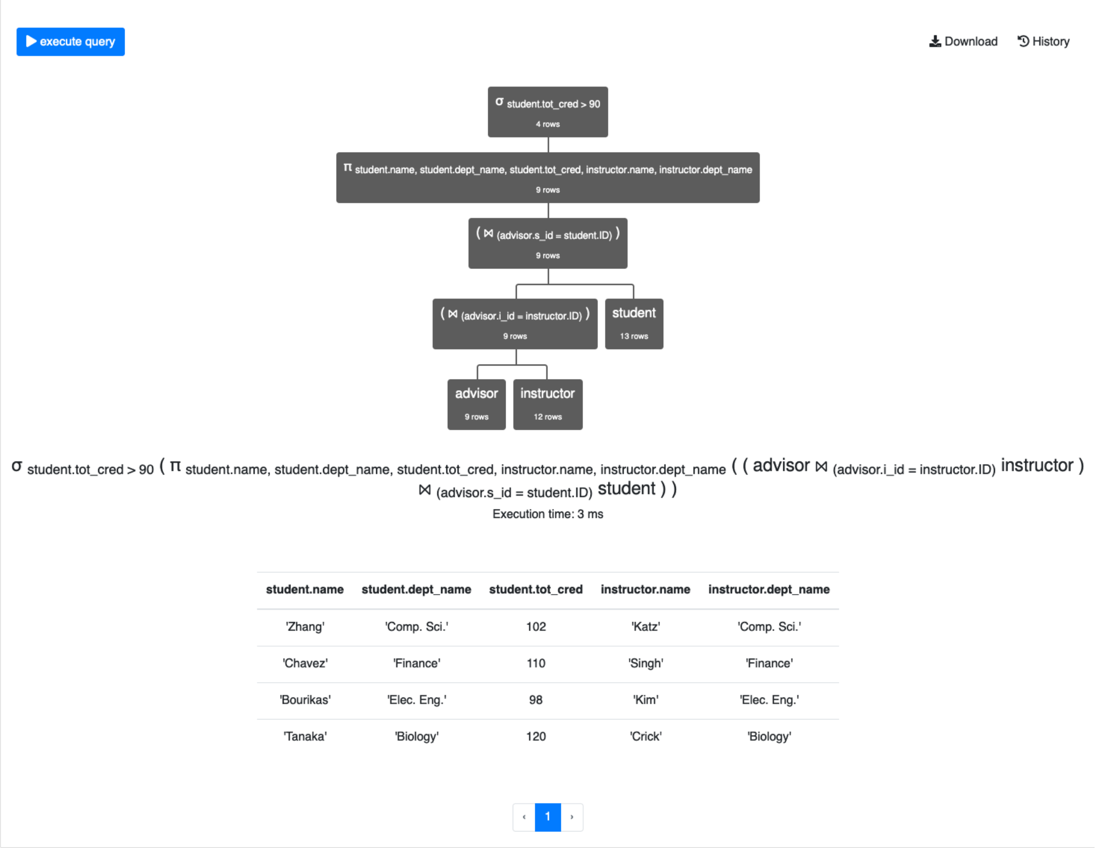
1. You will have to use the `student`, `instructor`, and `advisor` relations.
2. You should only include students that have an advisor, i.e., `instructor.name` and `instructor.dept_name` should be non-null for all rows.

Algebra statement:

```
σ student.tot_cred > 90
```

```
(
    π student.name, student.dept_name, student.tot_cred,
instructor.name, instructor.dept_name (
        (advisor ⨝ (advisor.i_id = instructor.ID)
instructor) ⨝ (advisor.s_id = student.ID) student
    )
)
```

Execution:


▶ execute query
Download
History
σ student.tot_cred > 90
4 rows
π student.name, student.dept_name, student.tot_cred, instructor.name, instructor.dept_name
9 rows
(⋈ (advisor.s_id = student.ID))
9 rows
(⋈ (advisor.i_id = instructor.ID))
9 rows
student
13 rows
advisor
9 rows
instructor
12 rows
‹ 1 ›

σ student.tot_cred > 90 (π student.name, student.dept_name, student.tot_cred, instructor.name, instructor.dept_name ((advisor ⋈ (advisor.i_id = instructor.ID) instructor) ⋈ (advisor.s_id = student.ID) student))

Execution time: 3 ms

student.name	student.dept_name	student.tot_cred	instructor.name	instructor.dept_name
'Zhang'	'Comp. Sci.'	102	'Katz'	'Comp. Sci.'
'Chavez'	'Finance'	110	'Singh'	'Finance'
'Bourikas'	'Elec. Eng.'	98	'Kim'	'Elec. Eng.'
'Tanaka'	'Biology'	120	'Crick'	'Biology'

R1 Execution Result

R2

Write a relational algebra statement that produces a relation with the columns:

- `course_id`
- `title`
- `prereq_course_id`
- `prereq_course_title`

This relation represents courses and their prereqs.

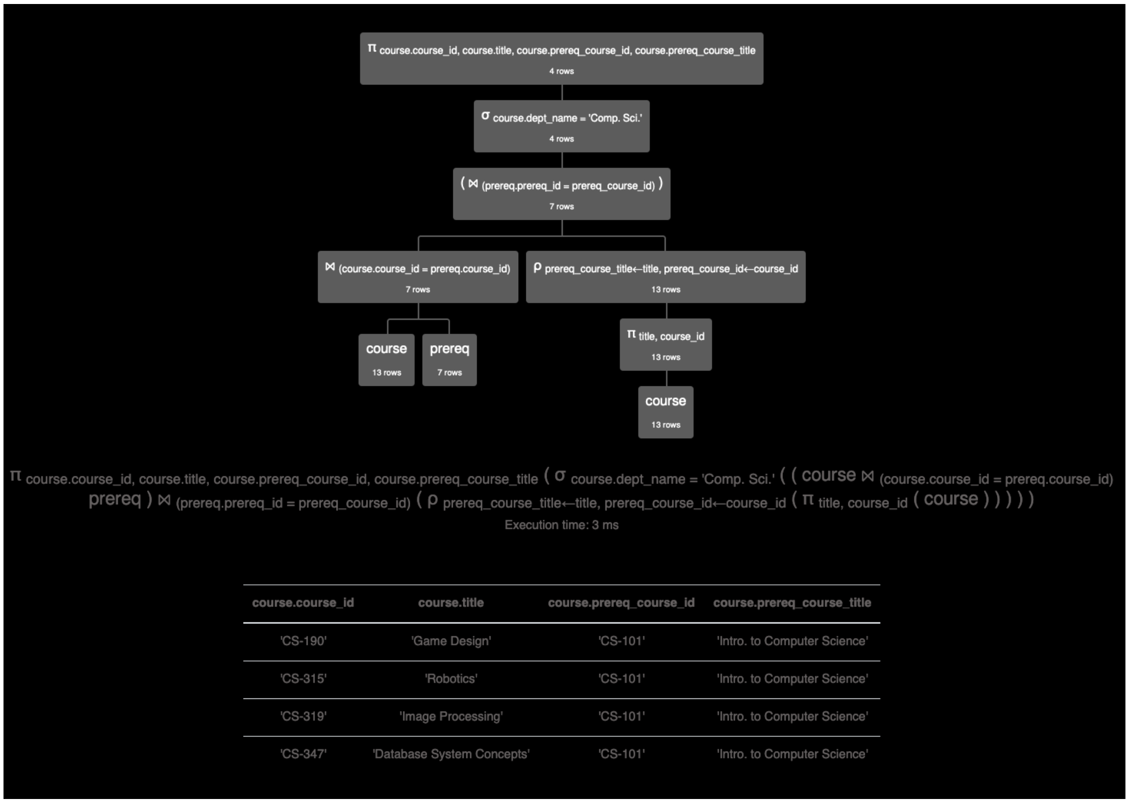
Note:

1. This query requires the `course` and `prereq` tables.
2. Your answer should only include courses in the `Comp. Sci.` department.
3. If a course has no prereqs, `prereq_course_id` and `prereq_course_title` should both be *null*.
4. You *may* have to use table and column renaming.

Algebra statement:

```
π course.course_id, course.title, course.prereq_course_id,
course.prereq_course_title (
    σ course.dept_name = 'Comp. Sci.' (
            course ⨝ (course.course_id = prereq.course_id)
prereq ⨝ (prereq.prereq_id = prereq_course_id) (ρ
prereq_course_title ← title, prereq_course_id ← course_id (π
title, course_id (course)))
    )
)
```

Execution:


π course.course_id, course.title, course.prereq_course_id, course.prereq_course_title
4 rows
σ course.dept_name = 'Comp. Sci.'
4 rows
(⋈ (prereq.prereq_id = prereq_course_id))
7 rows
⋈ (course.course_id = prereq.course_id)
7 rows
ρ prereq_course_title←title, prereq_course_id←course_id
13 rows
course
13 rows
prereq
7 rows
π title, course_id
13 rows
course
13 rows

π course.course_id, course.title, course.prereq_course_id, course.prereq_course_title (σ course.dept_name = 'Comp. Sci.' ((course ⋈ (course.course_id = prereq.course_id) prereq) ⋈ (prereq.prereq_id = prereq_course_id) (ρ prereq_course_title←title, prereq_course_id←course_id (π title, course_id (course)))))

Execution time: 3 ms

course.course_id	course.title	course.prereq_course_id	course.prereq_course_title
'CS-190'	'Game Design'	'CS-101'	'Intro. to Computer Science'
'CS-315'	'Robotics'	'CS-101'	'Intro. to Computer Science'
'CS-319'	'Image Processing'	'CS-101'	'Intro. to Computer Science'
'CS-347'	'Database System Concepts'	'CS-101'	'Intro. to Computer Science'

R2 Execution Result

SQL

New Database

MySQL Tutorial is a good site with information that complements and extends the core material in our course. Much of the material the site covers is applicable to other SQL products. MySQL Tutorial uses an interesting dataset that is more complex than the simple "db_book" database. This is the Classic Models Dataset. The complexity allows us to better appreciate more complex SQL concepts.

You learned how to run a SQL script/file as part of HW0. **Use the same approach to load and create the `Classic Models Database`.** The file is `classic-models-database.sql` and is in the HW1 folder.

To test loading the data, you can use the cell below.

```
In [ ]: %sql mysql+pymysql://root:dbuserdbuser@localhost
        %sql show databases;
```

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

Out[ ]:

Database
classicmodels
db_book
information_schema
mysql
performance_schema
sys

```
In [ ]: %sql USE classicmodels;
```

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

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

SQL 1

This query uses `customers` and `employees`.

Write and execute a SQL query that produces a table with the following columns:

- `customerContactName`
- `customerPhone`
- `salesRepName`

Only keep customers from France. Order your output by `customerContactName`.

Notes:

- The names of your columns must match exactly with what is specified.
- `customerContactName` can be formed by combining `customers.contactFirstName` and `customers.contactLastName` .
- `salesRepName` can be formed by combining `employees.firstName` and `employees.lastName` .

In [ ]:

```
%%sql

SELECT
    CONCAT(customers.contactFirstName, ' ', customers.contactLastName) AS cu
    customers.phone AS customerPhone,
    CONCAT(employees.firstName, ' ', employees.lastName) AS salesRepName
FROM
    customers
JOIN
    employees ON customers.salesRepEmployeeNumber = employees.employeeNumber
WHERE
    customers.country = 'France'
ORDER BY
    customerContactName;
```

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

Out[ ]:

customerContactName	customerPhone	salesRepName
Annette Roulet	61.77.6555	Gerard Hernandez
Carine Schmitt	40.32.2555	Gerard Hernandez
Daniel Tonini	30.59.8555	Gerard Hernandez
Daniel Da Silva	+33 1 46 62 7555	Loui Bondur
Dominique Perrier	(1) 47.55.6555	Loui Bondur
Frédérique Citeaux	88.60.1555	Gerard Hernandez
Janine Labrune	40.67.8555	Gerard Hernandez
Laurence Lebihan	91.24.4555	Loui Bondur
Marie Bertrand	(1) 42.34.2555	Loui Bondur
Martine Rancé	20.16.1555	Gerard Hernandez
Mary Saveley	78.32.5555	Loui Bondur
Paul Henriot	26.47.1555	Loui Bondur

SQL 2

This query uses `employees` , `customers` , `orders` , `orderdetails` .

Write and execute a SQL query that produces a table showing the amount of money each sales rep has generated.

Your table should have the following columns:

- `salesRepName`
- `moneyGenerated`

Order your output from greatest to least `moneyGenerated` .

Notes:

- The names of your columns must match exactly with what is specified.
- `salesRepName` can be formed by combining `employees.firstName` and `employees.lastName` .
- To calculate `moneyGenerated` :
  - Every order in `orders` is associated with multiple rows in `orderdetails` . The total amount of money spent on an order is the sum of `quantityOrdered * priceEach` for all the associated rows in `orderdetails` . **Only consider orders that are `Shipped` .**
  - A customer can have multiple orders. The total amount of money a customer has spent is the sum of the money spent on all that customer's orders.
  - A sales rep can have multiple customers. `moneyGenerated` is the sum of the money spent by all that sales rep's customers.
- You may find the WITH keyword to be useful for cleaner code.

In [ ]:

```
%%sql

WITH TotalSales AS (
    SELECT
        CONCAT(employees.firstName, ' ', employees.lastName) AS salesRepName
        SUM(od.quantityOrdered * od.priceEach) AS moneyGenerated
    FROM
        employees
    JOIN customers ON employees.employeeNumber = customers.salesRepEmployeeN
    JOIN orders ON customers.customerNumber = orders.customerNumber
    JOIN orderdetails od ON orders.orderNumber = od.orderNumber
    WHERE
        orders.status = 'Shipped'
    GROUP BY
        salesRepName
)

SELECT
    salesRepName,
    moneyGenerated
FROM
    TotalSales
ORDER BY
    moneyGenerated DESC;
```

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

Out[ ]:

salesRepName	moneyGenerated
Gerard Hernandez	1065035.29
Leslie Jennings	1021661.89
Pamela Castillo	790297.44
Larry Bott	686653.25
Barry Jones	637672.65
George Vanauf	584406.80
Loui Bondur	569485.75
Peter Marsh	523860.78
Andy Fixter	509385.82
Foon Yue Tseng	488212.67
Mami Nishi	457110.07
Steve Patterson	449219.13
Martin Gerard	387477.47
Julie Firrelli	386663.20
Leslie Thompson	307952.43