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 [2]:

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

name = "Kin Hang Godric Li"
uni = "kgl2128"
track = "Non-programming Track"

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

```
Kin Hang Godric Li
kgl2128
Non-programming Track
```

Setup

SQL Magic

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

In [66]:

```
%load_ext sql
```

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

You may need to change the password below.

In [67]:

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

In [68]:

```
%sql SELECT * FROM db_book.student WHERE ID = 12345
```

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

Out[68]:

ID	name	dept_name	tot_cred
12345	Shankar	Comp. Sci.	32

Python Libraries

In [69]:

```
from IPython.display import Image
import pandas
```

Written Questions

Chapter 1 from the recommended textbook Database System Concepts, Seventh Edition (https://codex.cs.yale.edu/avi/db-book/) 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 (https://codex.cs.yale.edu/avi/db-book/slides-dir/index.html) 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 allows storage, retrieval, running queries on data. Relational databases organise data with tables where each row represents a relationship among entities and each column is an attribute. Tables can then be linked through a primary key or foreign key.

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?

One of the problem is data searching. Shared spread sheet requires a different program to be written for every search operation while DMBS provides builtin search operations through SQL.
(https://www.geeksforgeeks.org/advantages-of-dbms-over-file-system/ (https://www.geeksforgeeks.org/advantages-of-dbms-over-file-system/))
Another problem is data volume. Shared spread sheets are not good at handling millions of rows of data while DMBS stores data more efficiently and is designed to handle millions of rows of data.
(https://earthsoft.com/2018/06/07/databases-versus-spreadsheets/ (https://earthsoft.com/2018/06/07/databases-versus-spreadsheets/))

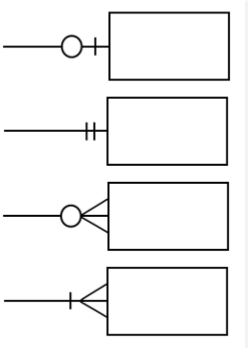
W3

Explain the differences between SQL, MySQL Server and DataGrip.

SQL is the language you use to manipulate a database system;
MySQL Server is a DBMS that stores the data and uses SQL as its query language;
DataGrip is a platform that allows developer to use SQL to communicate with MySQL to manipulate the database.

W4

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



First crow's foot: zero or one entity;
Second crow's foot: one and only one entity;
Third crow's foot: zero or many entities;
Fourth crow's foot: one or many entities.

W5

What is a primary key and why is it important?

A primary key should be unique and is used to identify a certain row in a table. It is important because it prevents duplicates in the data

W6

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

It means output from one operation can become input of another operation which allows expression to be nested.(https://home.adelphi.edu/~siegfried/cs443/443l9.pdf (https://home.adelphi.edu/~siegfried/cs443/443l9.pdf)). For example, "π Title, Profit (σ Budget > 1000 (Movies))" first selects movies with budget > 1000 and display their title and profit.

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?

A domain is the set of all possible values of an attribute(year/semester is a domain);
An atomic domain is a domain where all possible values cannot be split into more specific attributes(year is an atomic domain, semester is an atomic domain);
a domain refers to the values of the data and type refers to the datatype (year/semester is a domain and its data type is CHAR(6))

W8

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

A database schema indicates the logical structure of a databse to show what kind of data is in a table; a database instance is a snapshot of a the data in the database at a given instant in time

W9

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

Data definiton language (DDL) is reponsible for dealing with the structure of the database such as creating/deleting tables and modifying structures of tables through adding/removing columns, similar to defining a schema;
Data manipulation language (DML) is responsible for dealing with the content of the data in the database such as retrieving data, adding/removing entities from tables, and modifying entities in a table.
(https://www.almabetter.com/bytes/tutorials/sql/dml-ddl-commands-in-sql (https://www.almabetter.com/bytes/tutorials/sql/dml-ddl-commands-in-sql))

W10

What is physical data independence?

Physical data independence is the ability to freely modify the implementation details of the physical storage of data without affecting users' and applications' way of interaction with the data at a conceptual and logical level
(https://unstop.com/blog/data-independence-in-dbms (https://unstop.com/blog/data-independence-in-dbms))

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.

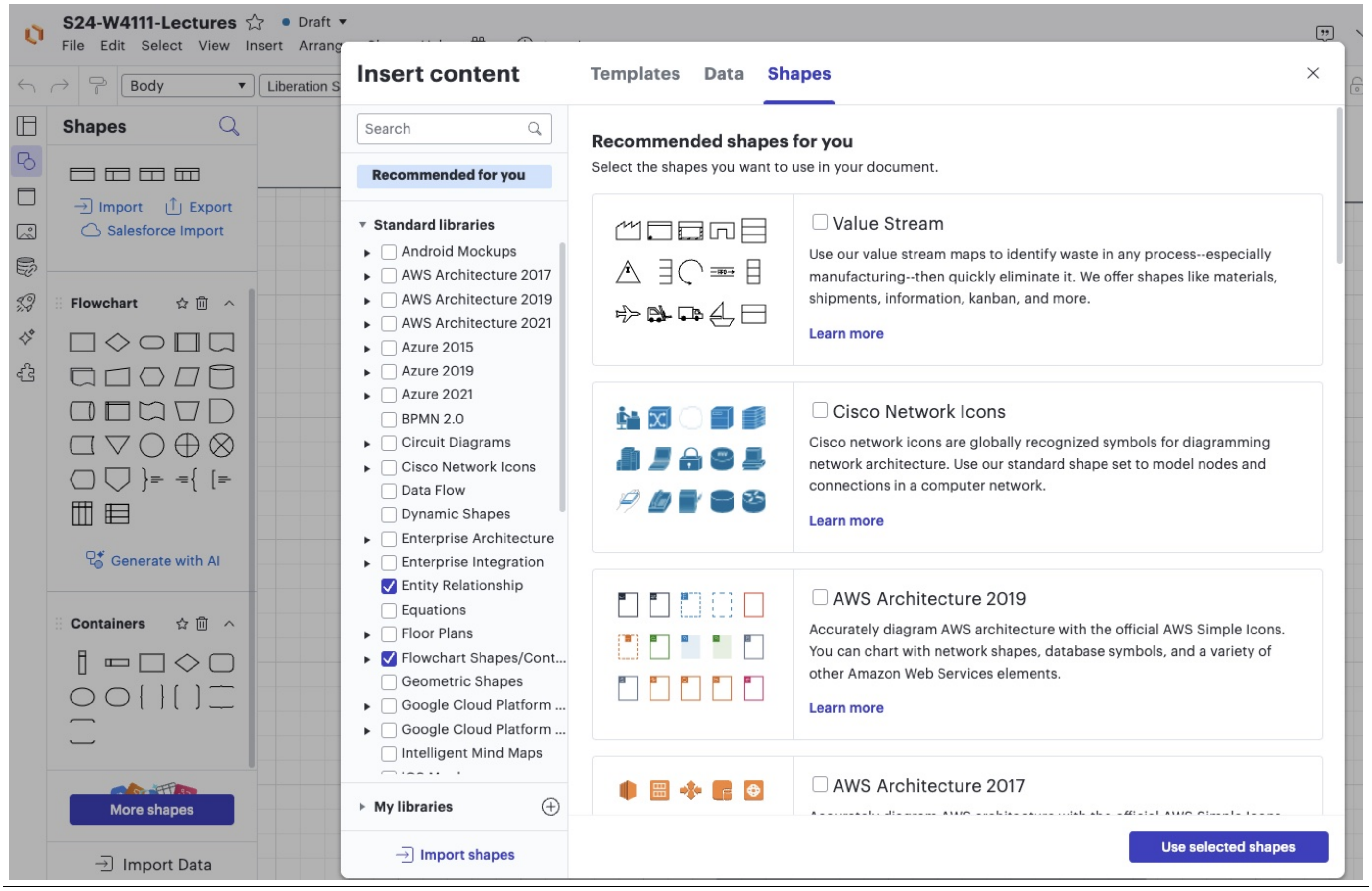
In this problem, you will:

- 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. (https://www.lucidchart.com/pages/landing)](https://www.lucidchart.com/pages/landing) 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 (https://lucid.app/lucidchart/828777b1-7b2d-4828-bedb-37b6d456c33e/edit?invitationId=inv_a142899a-7e60-44e9-b18e-335d7c9767fc)](https://lucid.app/lucidchart/828777b1-7b2d-4828-bedb-37b6d456c33e/edit?invitationId=inv_a142899a-7e60-44e9-b18e-335d7c9767fc) 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

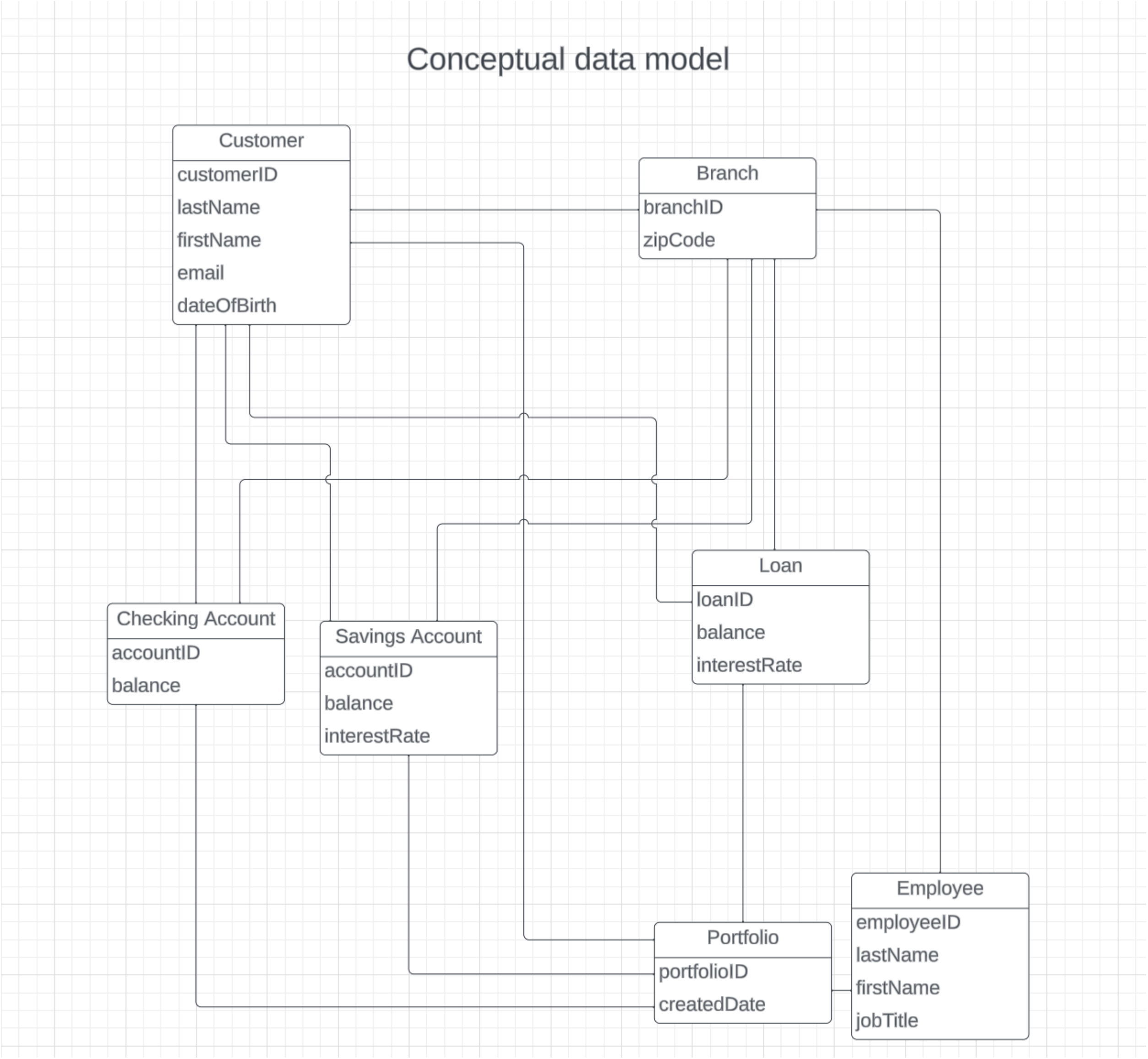
1. Place your Logical Model diagram below.
2. You *may* have to add attributes to entities to implement the model.
3. You *may* make reasonable assumptions. Please document your assumptions below. You may add comments/notes to your diagram for clarity.

Assumptions:

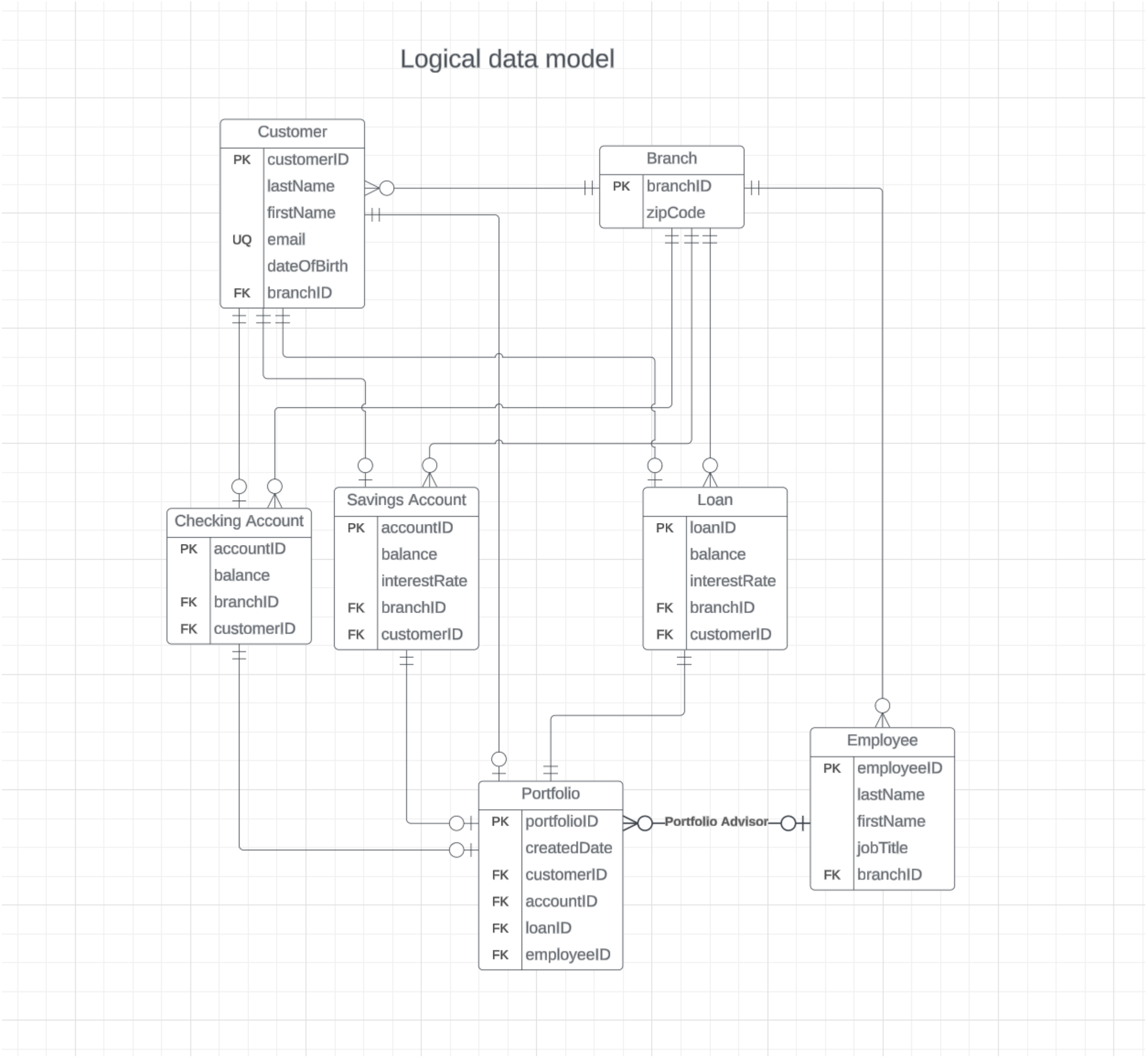
1. Each customer may have 0 or 1 portfolio
2. Each customer may have 0 or 1 loan
3. A checking account and savings account can have 0 or 1 portfolio
4. A loan has exactly one portfolio

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.



Conceptual ER Diagram



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.

$classroom(\underline{building}, \underline{room\_number}, capacity)$

Answer Format

For the answers to the relational algebra questions, you will use the RelaX calculator (https://dbis-uibk.github.io/relax/calc/gist/4f7866c17624ca9dfa85ed2482078be8/relax-silberschatz-english.txt/0) 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

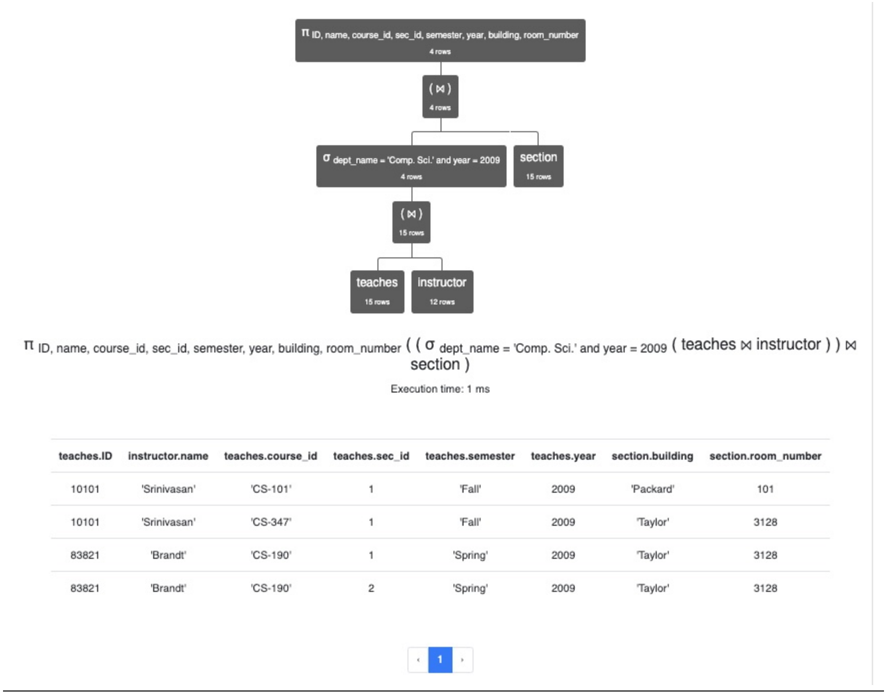
Note:

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

π ID, name, course_id, sec_id, semester, year, building, room_number ((σ dept_name = 'Comp. Sci.' and year = 2009 (teaches ⋈ instructor)) ⋈ 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.

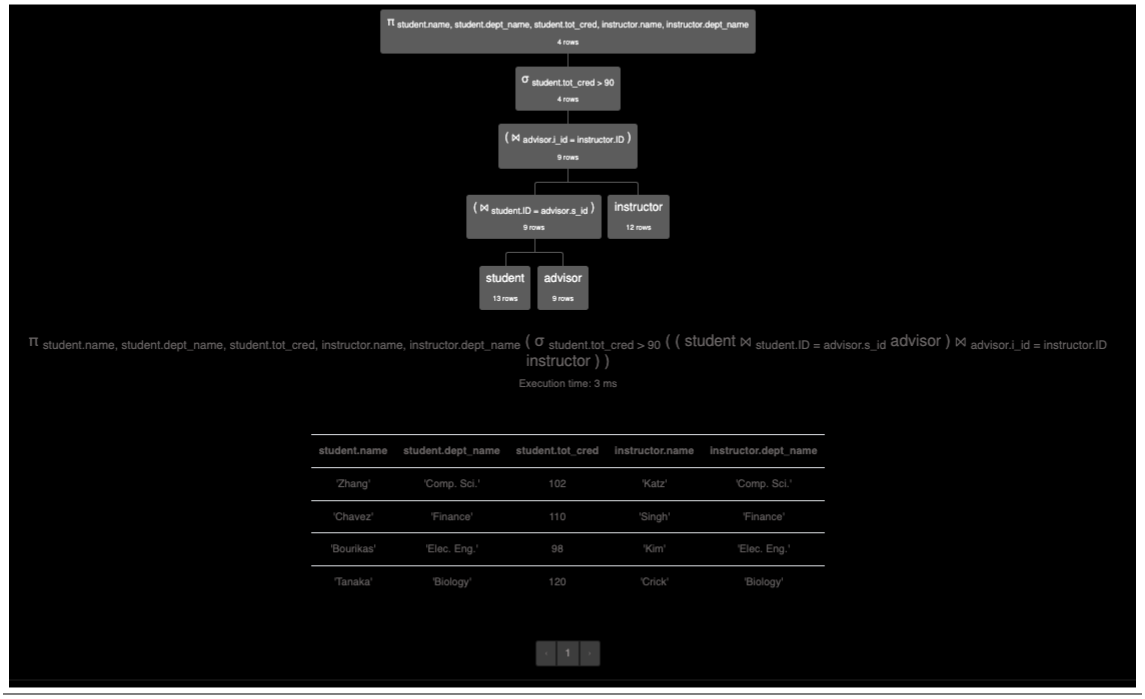
Note:

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:

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

Execution:


π student.name, student.dept_name, student.tot_cred, instructor.name, instructor.dept_name
4 rows
σ student.tot_cred > 90
4 rows
(⋈ advisor.i_id = instructor.ID)
9 rows
(⋈ student.ID = advisor.s_id)
9 rows
instructor
12 rows
student
13 rows
advisor
9 rows

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

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:

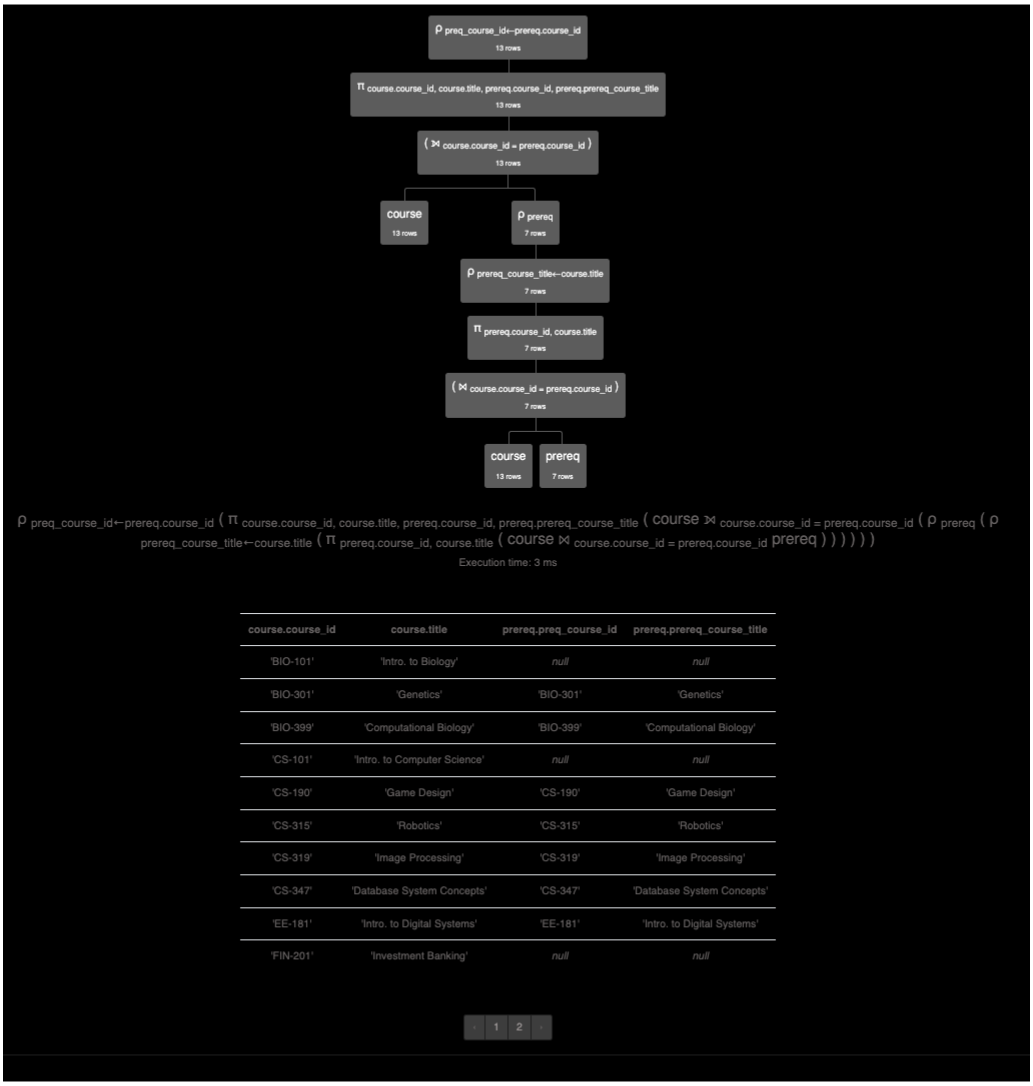
1. This query requires the `course` and `prereq` tables.
2. Your answer should only include courses in the `Comp. Sci.` department.
3. If a course has no prereqs, `prereq_course_id` and `prereq_course_title` should both be *null*.
4. You *may* have to use table and column renaming.

Algebra statement:

```
ρ preq_course_id←prereq.course_id (
    π course.course_id, course.title, prereq.course_id, prereq.prereq_course_title (
        course ⟕ course.course_id = prereq.course_id (
            ρ prereq (
                ρ prereq_course_title←course.title (
                    π prereq.course_id, course.title (
                        course ⋈ course.course_id = prereq.course_id prereq
                    )
                )
            )
        )
    )
)
```

Execution:

|

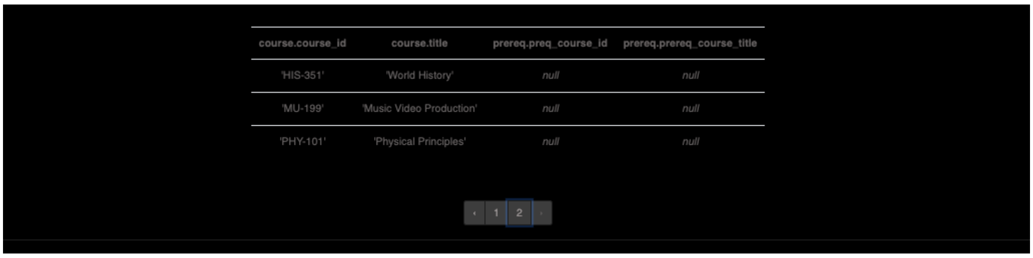

ρ preq_course_id←prereq.course_id
13 rows
π course.course_id, course.title, prereq.course_id, prereq.prereq_course_title
13 rows
(⟕ course.course_id = prereq.course_id)
13 rows
course
13 rows
ρ prereq
7 rows
ρ prereq_course_title←course.title
7 rows
π prereq.course_id, course.title
7 rows
(⋈ course.course_id = prereq.course_id)
7 rows
course
13 rows
prereq
7 rows
‹
1
2
›

ρ preq_course_id←prereq.course_id (π course.course_id, course.title, prereq.course_id, prereq.prereq_course_title (course ⟕ course.course_id = prereq.course_id (ρ prereq (ρ prereq_course_title←course.title (π prereq.course_id, course.title (course ⋈ course.course_id = prereq.course_id prereq))))))

Execution time: 3 ms

course.course_id	course.title	prereq.preq_course_id	prereq.prereq_course_title
'BIO-101'	'Intro. to Biology'	*null*	*null*
'BIO-301'	'Genetics'	'BIO-301'	'Genetics'
'BIO-399'	'Computational Biology'	'BIO-399'	'Computational Biology'
'CS-101'	'Intro. to Computer Science'	*null*	*null*
'CS-190'	'Game Design'	'CS-190'	'Game Design'
'CS-315'	'Robotics'	'CS-315'	'Robotics'
'CS-319'	'Image Processing'	'CS-319'	'Image Processing'
'CS-347'	'Database System Concepts'	'CS-347'	'Database System Concepts'
'EE-181'	'Intro. to Digital Systems'	'EE-181'	'Intro. to Digital Systems'
'FIN-201'	'Investment Banking'	*null*	*null*

|



course.course_id	course.title	prereq.preq_course_id	prereq.prereq_course_title
'HIS-351'	'World History'	*null*	*null*
'MU-199'	'Music Video Production'	*null*	*null*
'PHY-101'	'Physical Principles'	*null*	*null*

R2 Execution Result

SQL

New Database

[MySQL Tutorial (https://www.mysqltutorial.org/)](https://www.mysqltutorial.org/) 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 (https://www.mysqltutorial.org/getting-started-with-mysql/mysql-sample-database/)](https://www.mysqltutorial.org/getting-started-with-mysql/mysql-sample-database/). 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 [8]:
```
%sql show tables;
```

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

Out[8]:

Tables_in_classicmodels
customers
employees
offices
orderdetails
orders
payments
productlines
products

In [73]:
```
%sql USE classicmodels;
```

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

Out[73]:
```
[]
```

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 [50]:
```
%%sql
SELECT
    CONCAT_WS(' ', customers.contactFirstName, customers.contactLastName) AS customerContactName,
    customers.phone AS customerPhone,
    CONCAT_WS(' ', employees.firstName, employees.lastName) AS salesRepName
FROM
    customers
    LEFT JOIN
    employees
        ON customers.salesRepEmployeeNumber = employees.employeeNumber
WHERE
    customers.country = "France"
ORDER BY
    customerContactName
```

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

Out[50]:

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 (https://www.tutorialspoint.com/mysql/mysql_with.htm)](https://www.tutorialspoint.com/mysql/mysql_with.htm) to be useful for cleaner code.

In [123]:

```
%%sql
WITH TotalMoneyOrderDetail AS (
    SELECT
        orderNumber,
        SUM(quantityOrdered * priceEach) AS TotalMoneyOrder
    FROM
        orderdetails
    GROUP BY
        orderdetails.orderNumber
),
TotalMoneyCustomer AS (
    SELECT
        SUM(TotalMoneyOrderDetail.TotalMoneyOrder) AS TotalMoney,
        customerNumber
    FROM
        TotalMoneyOrderDetail
    JOIN
        orders ON TotalMoneyOrderDetail.orderNumber = orders.orderNumber
    WHERE
        orders.status = "Shipped"
    GROUP BY
        customerNumber
)
SELECT
    CONCAT_WS(' ', employees.firstName, employees.lastName) AS salesRepName,
    SUM(TotalMoneyCustomer.TotalMoney) AS moneyGenerated
FROM
    TotalMoneyCustomer
JOIN
    customers ON customers.customerNumber = TotalMoneyCustomer.customerNumber
JOIN
    employees ON customers.salesRepEmployeeNumber = employees.employeeNumber
GROUP BY
    customers.salesRepEmployeeNumber
ORDER BY
    moneyGenerated DESC
```

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

Out[123]:

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

In [ ]: