COMS W4111-002/V02, Spring 22: Take Home Midterm

Overview

NOTE:

- This is a draft of the midterm exam. We are releasing the draft to enable students to start thinking about the exam and potentially identifying material that needs clarification.
- Despite being a draft, the only changes will be clarifications. So, students can begin thinking about and working on answers.
- The TAs will provide guidance for submission formats and annotations.

Instructions

- **TA TODO:** TAs to finalize instructions, especially submission instructions.

Due Date, Completing the Exam and Rules

1. The midterm exam is due at 11:59 PM on Monday, 28-MAR-2022. **You are not allowed to use late days.**

2. You may use on-line information and sources to answer questions. But,
    A. You cannot simply cut and paste answers or code. You answer must demonstrate that you understood the material and are capable of producing an answer from your understanding.
    B. You must cite any only sources of information that you used. This can simply be a comment in a text/markdown cell in your answer. For example, `(Note: I used https://www.w3schools.com/sql/sql_check.asp to help me with the syntax for adding a check constraint).`
    C. You do NOT need to cite lecture notes, recordings, slides, ... You do not need to cite information from the recommended textbook or textbook slides.

3. You MUST NOT collaborate with ANYONE, including other students. You MAY speak with the professor or a TA to discuss the exam.

4. If you have questions, post them as PRIVATE question on Ed dicussion and use the Category `Exams->Midterm.`

5. There is a pinned Ed discussion topic Midterm Clarifications (https://edstem.org/us/courses/18760/discussion/1302989) that the professor and TA will use to communicate updates and clarifications. **Students are responsible for checking this post.**

Submission Format and Instructions

- **TA TODO:** TAs to finalize instructions, especially submission instructions.

Environment Setup

Notes:

1. This section tests your environment.

2. You will need to change the MySQL userID and password in some of the cells below to match your configuration.

3. You may need to load data and copy databases. The relevant questions provide information.

In [1]:
```
%load_ext sql
```

In [2]:
```
%sql mysql+pymysql://root:Edy990127@localhost
```

Out[2]:
```
'Connected: root@None'
```

In [3]:
```
from sqlalchemy import create_engine
```

In [4]:
```
sql_engine = create_engine("mysql+pymysql://root:Edy990127@localhost")
```

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

In [6]:
```
sql = """
    select customerName, customerNumber, city, country from classicmodels.custome
        where country = 'France'
"""

res = pd.read_sql(sql, con=sql_engine)
```

In [7]:
```
res
```

Out[7]:

	customerName	customerNumber	city	country
0	Atelier graphique	103	Nantes	France
1	La Rochelle Gifts	119	Nantes	France
2	Saveley & Henriot, Co.	146	Lyon	France
3	Daedalus Designs Imports	171	Lille	France
4	La Corne D'abondance, Co.	172	Paris	France
5	Mini Caravy	209	Strasbourg	France
6	Alpha Cognac	242	Toulouse	France
7	Lyon Souveniers	250	Paris	France
8	Auto Associés & Cie.	256	Versailles	France
9	Marseille Mini Autos	350	Marseille	France
10	Reims Collectables	353	Reims	France
11	Auto Canal+ Petit	406	Paris	France

In [8]:
```
import pymysql
```

In [9]:
```
sql_conn = pymysql.connect(
    user="root",
    password='Edy990127',
    host="localhost",
    port=3306,
    cursorclass=pymysql.cursors.DictCursor,
    autocommit=True)
```

In [10]:
```
cur = sql_conn.cursor()

res = cur.execute(sql)
res = cur.fetchall()
res
```

Out[10]:
```
[{'customerName': 'Atelier graphique',
  'customerNumber': 103,
  'city': 'Nantes',
  'country': 'France'},
 {'customerName': 'La Rochelle Gifts',
  'customerNumber': 119,
  'city': 'Nantes',
  'country': 'France'},
 {'customerName': 'Saveley & Henriot, Co.',
  'customerNumber': 146,
  'city': 'Lyon',
  'country': 'France'},
 {'customerName': 'Daedalus Designs Imports',
  'customerNumber': 171,
  'city': 'Lille',
  'country': 'France'},
 {'customerName': "La Corne D'abondance, Co.",
  'customerNumber': 172,
  'city': 'Paris',
  'country': 'France'},
 {'customerName': 'Mini Caravy',
  'customerNumber': 209,
  'city': 'Strasbourg',
  'country': 'France'},
 {'customerName': 'Alpha Cognac',
  'customerNumber': 242,
  'city': 'Toulouse',
  'country': 'France'},
 {'customerName': 'Lyon Souveniers',
  'customerNumber': 250,
  'city': 'Paris',
  'country': 'France'},
 {'customerName': 'Auto Associés & Cie.',
  'customerNumber': 256,
  'city': 'Versailles',
  'country': 'France'},
 {'customerName': 'Marseille Mini Autos',
  'customerNumber': 350,
  'city': 'Marseille',
  'country': 'France'},
 {'customerName': 'Reims Collectables',
  'customerNumber': 353,
  'city': 'Reims',
  'country': 'France'},
 {'customerName': 'Auto Canal+ Petit',
  'customerNumber': 406,
  'city': 'Paris',
  'country': 'France'}]
```

In [11]:
```
cur.close()
```

Written Questions

Note:

"If you can't explain something in a few words, try fewer." – Robert Brault

"Professor Ferguson has the patience of a ferret that just drank a double espresso. If your answer is long, he gets bored and cranky, and deducts points." - Anonymous TA advising students in a previous semester.

- We expect brief, succinct answers.
- We deduct points for bloviating.

W1

Briefly explain the differences between:

1. *Candidate Key* and *Super Key.*
2. *Primary Key* and *Unique_Key.*
3. *Natural Key* and *Surrogate Key.*

Answer

1. Super Key is a set of attributes or columns that uniquely identifies each row in a table while Candidate Key recognizes the tuples in relation or table.

The count of super keys is generally more as compared to the candidate key.

All candidate keys are super keys while not every super key is a candidate key.

Super keys make guideline to choose Candidate keys, Candidate keys make guidelines to choose primary keys.

(Note: I use information from https://byjus.com/gate/difference-between-super-key-and-candidate-key/#:~:text=Difference%20between%20Super%20Key%20and%20Candidate%20Key%20,make%2msclkid=19e7abcaadf911ec9cc549eef052a556 (https://byjus.com/gate/difference-between-super-key-and-candidate-key/#:~:text=Difference%20between%20Super%20Key%20and%20Candidate%20Key%20,make%2msclkid=19e7abcaadf911ec9cc549eef052a556))

2. There can be multiple unique keys but only one primary key.

Primary key can not have null values, unique key can have null values.

In primary key, duplicate keys are not allowed, while in unique key, if one or more key parts are null, then duplicate keys are allowed.

The purpose of the primary key is to enforce entity integrity while the purpose of unique key is to enforce unique data.

3. Natural key is an attribute that can uniquely identify a row, and exists in the real world, whiel surrogate key is an attribute that can uniquely identify a row, and does not exist in the real world.

W2

1. Define the concept of *immutable* columns (data).
2. What is a benefit of using immutable columns to define a primary key.

Answer

1. Immutable columns(data) are the data that is not allowed to be updated once it is instantiated.
2. In general, if the primary key is linked to a foreign key, or if it's used as an identifier outside the database (e.g. an URL), then the primary key should be immutable, unique and not null. Then, in this case, using immutable columns provide great convenience since they are immutable

W3

Codd's Third Rule states, "Null values (distinct from the empty character string or a string of blank characters and distinct from zero or any other number) are supported in fully relational DBMS for representing missing information and inapplicable information in a systematic way, independent of data type."

Briefly explain the value of this rule, and provide two examples.

Answer

This third rule means that we can use NULL to represent values like be 1. Data is missing. 2. Data is unknown. 3. Data is not applicable.

We use NULL because it is simple, clear, and a universally understood expression. While other indicators will cause confusion since the specific choice people choose to represent NULL is varying. Use other indicators will generally need more work and cause problems much more easily.

Example of cases that is applicable for "Null" usage: String: “ ”, “NA”.

W4

The relational model and SQL are *closed* under their operations. Briefly explain what this concept means.

Answer

For relational model and SQL, because they take an instance of relations and do operations that work on one or more relations to describe another relation without altering the original relation, they could leave both the operands and the outputs as relations. This is called " the relational model and SQL are closed under their operations"

W5

Codd's 6th rule states, "All views that are theoretically updatable are also updatable by the system."

Using the following table definition, use SQL (`CREATE VIEW`) to define:

1. Two views of the table that are theoretically not possible to update.
2. One view that is theoretically possible to update.

- You do not need to execute the create statement. We are focusing on your understanding.

```
create table S22_W4111_Midterm.midterm_students
(
    social_security_no char(9) not null
        primary key,
    last_name varchar(64) null,
    first_name varchar(64) null,
    enrollment_year year null,
    total_credits int null
);
```

Answer

1. If a view does not contain primary key then it cannot be updated.

CREATE VIEW student_name AS SELECT last_name, first_name FROM S22_W4111_Midterm.midterm_students

if a view has dinstinct clause in its statement, then it cannot be updated

CREATE VIEW student_name AS SELECT social_security_no, distinct total_credits FROM S22_W4111_Midterm.midterm_students

2. generally if I select everything from a table, it's updatable.

CREATE VIEW midterm_students AS SELECT * FROM S22_W4111_Midterm.midterm_students

W6

In the Columbia University directory of classes, (http://www.columbia.edu/cu/bulletin/uwb/#/cu/bulletin/uwb/subj/COMS/W4111-20213-002) the "Section Key" for this course is `20213COMS4111W002`.

- Explain why having a column `section_key varchar(17)` that holds section key values is not-atomic.
- Give two explanations for why using the section key (not atomic data) for a column causes problems.

Answer

1.Section key is this case is made up of several parts, 20213, COMS, W4111, 002, which means that it is reducible and divisible, able to be divided to other keys.

2.Using not atomic data for a column might make updates to the database occurring only partially.

A partially updated column would cause greater problems than just reject the data

W7

Briefly explain the differences between:

- Database stored procedure
- Database function
- Database trigger

Answer

Database Stored procedure can be executed when we need it. Database functions can be called but not executed. Database trigger can only be executed automatically on specified action on a table such as update and delete.

Database function must return a value. Data Base Stored Procedure return is optional, it can return zero or n values. Database trigger never return value.

Database function only have input parameters, Database stored procedures can have input or output parameters. Database trigger does not have input or output parameter.

Database functions can be called from database stored procedure or trigger. Database trigger can’t be called from store procedure or function. Database stored procedures can't be called from a function but can be called from a trigger.

W8

Briefly explain:

- Natural join
- Equi-join
- Theta join
- Self-join

Answer

EQUI JOIN performs a JOIN based on equality or matching column values of the associated tables. An equal sign is used as comparison operator in the where clause to refer equality between associated columns from different tables.

NATURAL JOIN is a type of EQUI JOIN, but for natural join, columns with the same name of associated tables will appear only once.

Theta Join is a type of join where the condition is not equality.

SELF JOIN is a regular join that join the table with itself.

W9

Consider the schema for Classic Models, (https://www.mysqltutorial.org/mysql-sample-database.aspx) which we have used in the class.

1. Is any entity type in the schema a *weak entity?* If yes, list one of the weak entity types.
2. In database design, using `ON DELETE CASCADE` may not be a desired behavior/design. Why is it more likely that `ON DELETE CASCADE` is the correct behavior for weak entities when the referencing row is deleted?

Answer

The payments entity is a weak entity. Since its primary key set consists of a foreign key from customers entity.

ON DELETE CASCADE is used to delete the rows from the child table automatically, when the rows from the parent table are deleted. In the case of weak entities, if the referencing row is deleted, it might be the case that the data has some problems, in this case, it would be wise to also delete the row in weak entities.

W10

1. Briefly explain the concept of a *database cursor.*
2. Why is using a cursor sometimes helpful for applications processing database information.

Answer

Database cursor is a temporary work station. It is allocated by database server for performing data manipulation language operations by user. Cursors are used to store database tables.

By using a cursor, we are generally iterating or moving from one row to the next and updating rows at the same time. If we encounter an error, we can try something else, or just skip the operation. In this sense, it would be very convenient to process database information as we could process the data row by row and could even move back to the previous row.

Relational Algebra

R1

- You can assume that the type for the columns in this question are `TEXT`.
- Translate the following relational schema definition into an equivalent SQL `CREATE TABLE` statement.
- You do not need to execute the statement. We are focusing on understanding.

$$(\underline{branch\_id}, \underline{account\_id}, balance)$$

Answer

create table branch_account (branch_id TEXT not null primary key, account_id TEXT not null primary key, balance TEXT null);

R2

- Use the RelaX online calculator with the Silberschatz - UniversityDB (https://dbis-uibk.github.io/relax/calc/gist/4f7866c17624ca9dfa85ed2482078be8/relax-silberschatz-english.txt/0) for this query.
- You may use **only** the `ρ (pi), σ, ⋈, ⟕, ⋈` for this question. You can use predicates/conditions for the query, and column list for the project.
- Write a relational algebra statement that produces a table containing the `ID` and `name` of instructors who do not advise any students. Your result should be of the form:

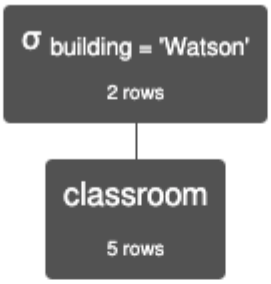
```
(ID, name)
```

- Your answer must contain the query statement (in text, not in an image) and a screen capture of the query execution and result. An example of the structure of the answer is:

Query:

```
σ building='Watson' (classroom)
```

Screen capture:



$$\sigma_{\text{building = 'Watson'}}(\text{classroom})$$

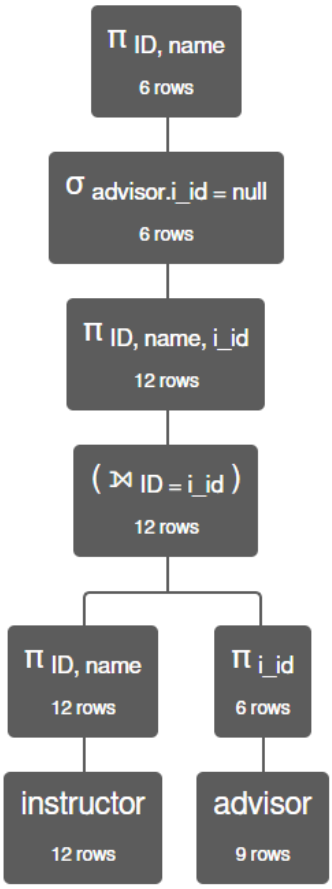
classroom.building	classroom.room_number	classroom.capacity
'Watson'	100	30
'Watson'	120	50

Answer

Query:

πID,name (σadvisor.i_id = null (πID,name,i_id (πID,name (instructor) ⟗ID = i_id (πi_id (advisor)))))

In [12]:
```
#Screen Capture:
er_model_file_name1 = 'R2-1.png'
er_model_file_name2 = 'R2-2.png'
print("\n")
from IPython.display import Image
from IPython.display import display
x = Image(filename=er_model_file_name1)
y = Image(filename=er_model_file_name2)
display(x, y)
```



$$\pi_{ID,\ name}\left(\sigma_{advisor.i\_id\ =\ null}\left(\pi_{ID,\ name,\ i\_id}\left(\pi_{ID,\ name}\left(instructor\right)\bowtie_{ID\ =\ i\_id}\left(\pi_{i\_id}\left(advisor\right)\right)\right)\right)\right)$$

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

R3

- Use the RelaX online calculator with the Silberschatz - UniversityDB (https://dbis-uibk.github.io/relax/calc/gist/4f7866c17624ca9dfa85ed2482078be8/relax-silberschatz-english.txt/0) for this query.
- You may only use the relational operators π, σ, $\bowtie$ for the solution. You may use column lists and conditions/predicates.
- Write a relational algebra expression equivalent to the following SQL statement.

```
select * from takes
    where course_id in
          (select course_id from course where dept_name not in
              (select dept_name from department where building='Taylo
r'))
    and
    year='2009';
```

- **Note:**
  - The book's sample data you loaded into MySQL is different from the data in the RelaX calculator. RelaX has data from a prior version of the book.

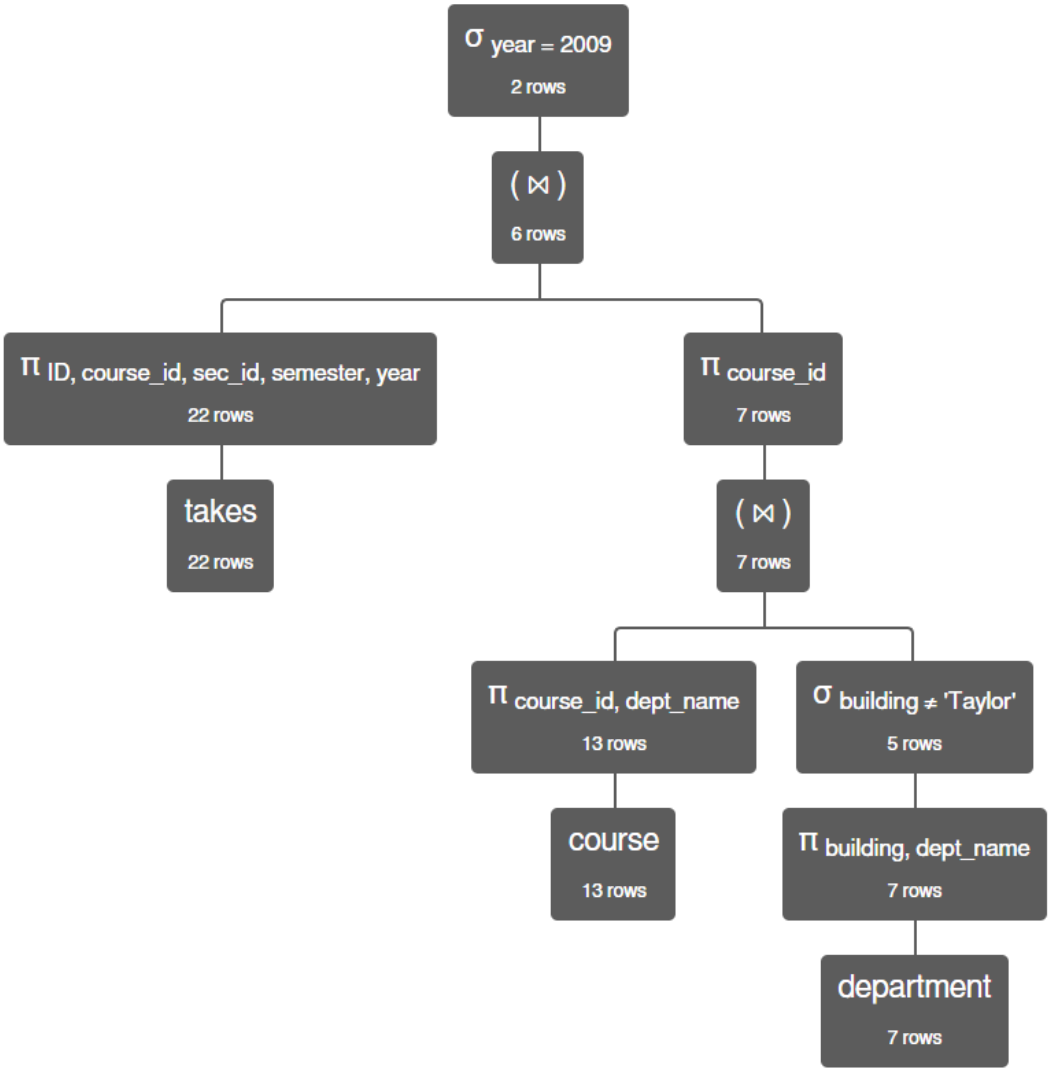
- If you want to test the SQL statement in MySQL, you can load the data from version 6 (https://www.db-book.com/db6/lab-dir/sample_tables-dir/index.html) of the book into a separate schema in MySQL.

Answer

Query:

σyear = 2009 (πID,course_id,sec_id,semester,year (takes) ⋈ (π course_id ((πcourse_id,dept_name (course)) ⋈ (σbuilding ≠ 'Taylor' (πbuilding,dept_name (department))))))

In [13]:
```
#Screen Capture:
er_model_file_name1 = 'R3-1.png'
er_model_file_name2 = 'R3-2.png'
print("\n")
from IPython.display import Image
from IPython.display import display
x = Image(filename=er_model_file_name1)
y = Image(filename=er_model_file_name2)
display(x, y)
```



$$\sigma_{year = 2009}(\pi_{ID, course\_id, sec\_id, semester, year}(takes) \bowtie (\pi_{course\_id}((\pi_{course\_id, dept\_name}(course)) \bowtie (\sigma_{building \neq 'Taylor'}(\pi_{building, dept\_name}(department))))))$$

takes.ID	takes.course_id	takes.sec_id	takes.semester	takes.year
44553	'PHY-101'	1	'Fall'	2009
98988	'BIO-101'	1	'Summer'	2009

Entity Relationship Model

- Setup:
  - Being able to translate a written description of a schema into an ER diagram is an important skill.
  - This question tests that skill by describing a data model that you have to define using Crow's Foot Notation.
  - You ER model must be implementable in SQL DDL.
  - You do not need to choose data types for columns.
  - You should add notes to your diagram for clarification if you think necessary.

ER1

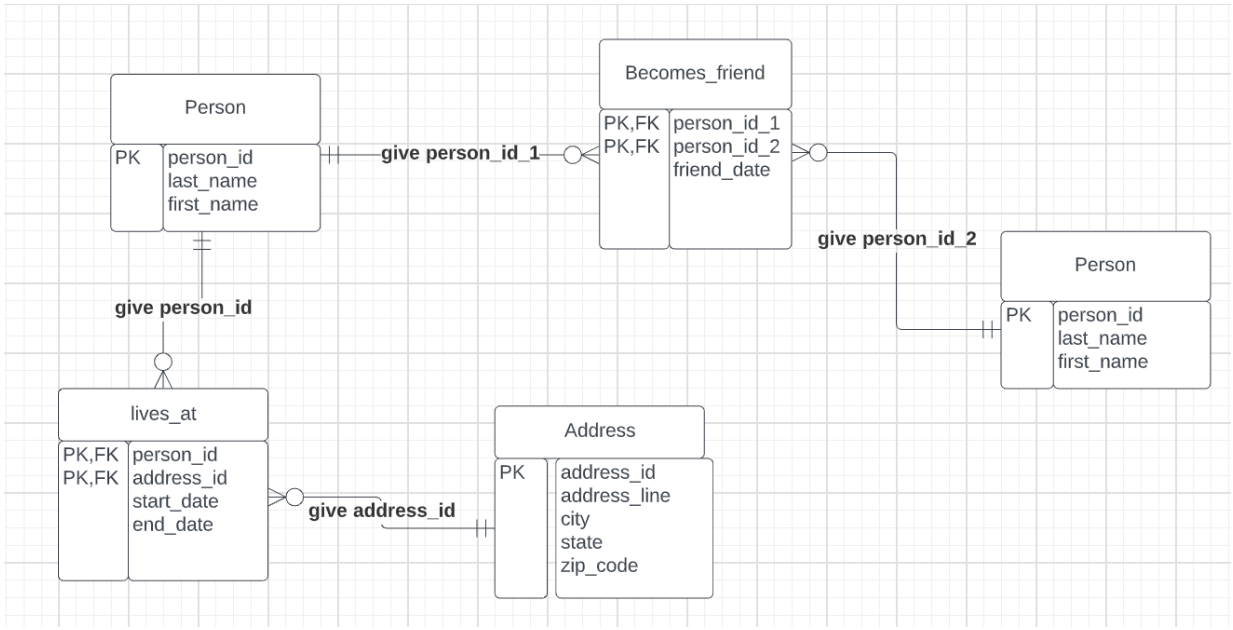
- Description:
  - There are two entity types:
    1. $Person(\underline{person\_id}, last\_name, first\_name)$
    2. $Address(\underline{address\_id}, address\_line, city, state, zip\_code)$
  - There are two many-to-many relationships:
    1. $Person - Address$: A $Person$ $lives\_at$ an $Address$ from a $start\_date$ to an end_date.
    2. $Person - Person$: A $Person$ $became\_friends\_with$ a $Person$ on a $friend\_date$
  - You do not need to worry about semantics constraints, e.g. it is OK if $lives\_at$ terms overlap.

- **Notes:**
  - Use LucidChart to draw your ER diagram.
  - You do not need to worry about data types.
  - You must show primary and foreign keys.

- You do not need to worry about semantics constraints, e.g. it is OK if $lives\_at$ terms overlap.
- Put your diagram in the directory that contains your notebook and include following instructions provided for previous homework assignments.

Answer

In [14]:
```
er_model_file_name1 = 'ER1.png'
print("\n")
from IPython.display import Image
from IPython.display import display
x = Image(filename=er_model_file_name1)
display(x)
```



ER2

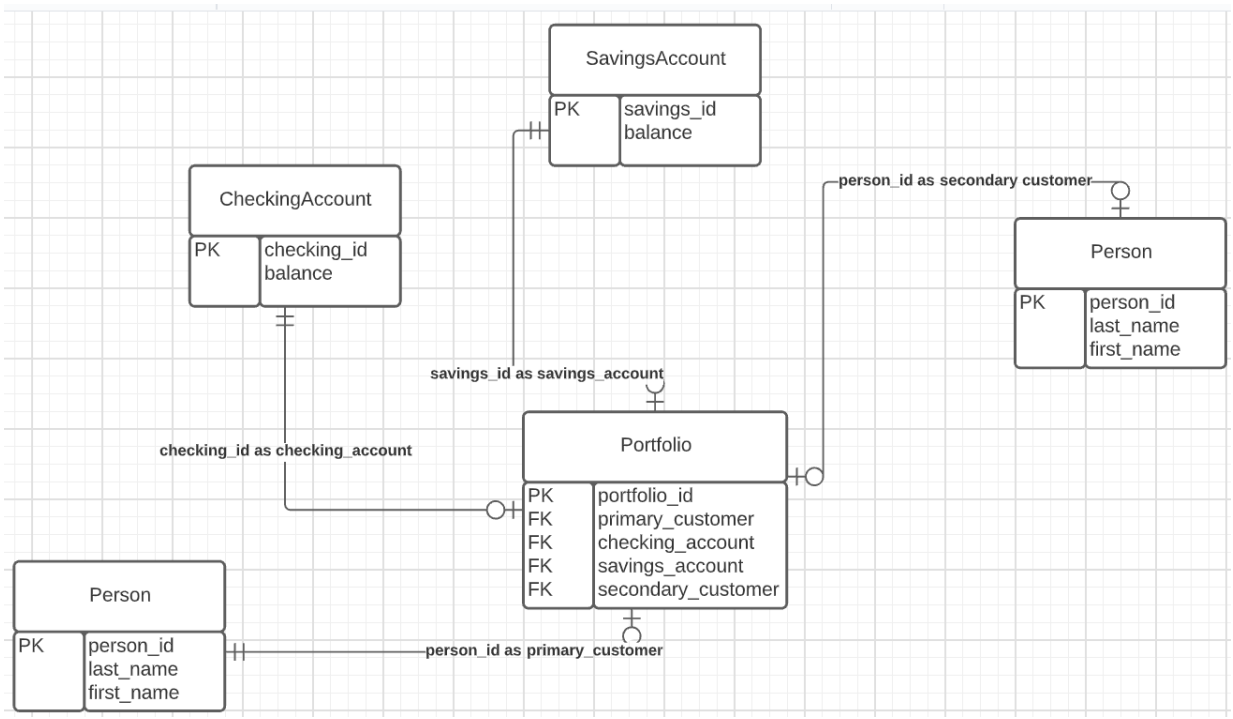
- Description:
  - There are three entity types:
    1. $CheckingAccount(\underline{checking\_id}, balance)$
    2. $SavingsAccount(\underline{savings\_id}, balance)$
    3. $Person(\underline{person\_id}, last\_name, first\_name)$
  - A $Portfolio$ is an entity type that:

    1. Has a primary key $portfolio\_id$.
    2. *Aggregates* other entity types:
       - Exactly one $primary\_customer$.
       - At most one $secondary\_customer$.
       - Exactly one $checking\_account$.
       - Exactly one $savings\_account$.

- **Notes:**
  - Use LucidChart to draw your ER diagram.
  - You do not need to worry about data types.
  - You must show primary and foreign keys.
  - You do not need to worry about semantics constraints, e.g. it is OK of the same person is the primary and secondary customer.

Answer

In [15]:
```
er_model_file_name1 = 'ER2.png'
print("\n")
from IPython.display import Image
from IPython.display import display
x = Image(filename=er_model_file_name1)
display(x)
```



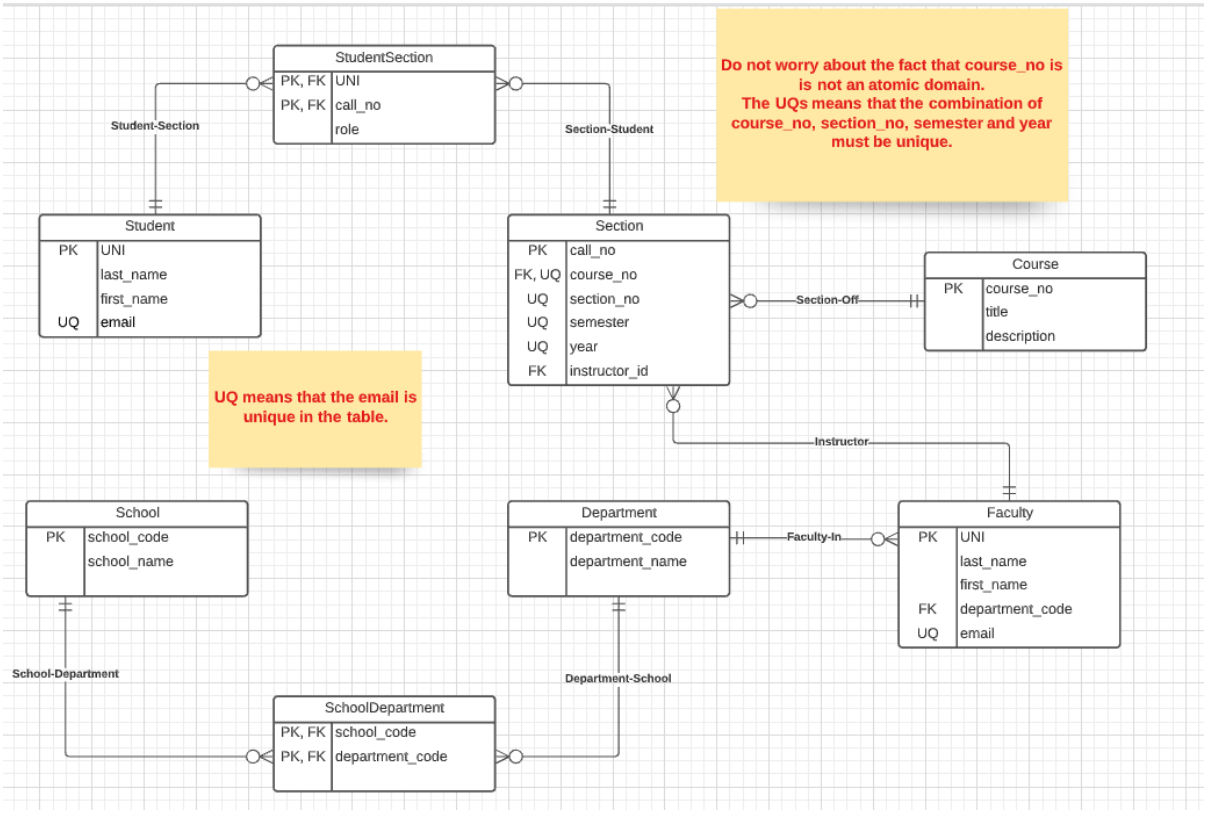
SQL Schema and DDL

DDL1

- You have a logical datamodel ER-diagram (see below).
- You need to use DDL to define a schema that realizes the model.
- Logical models are not specific enough for direct implementation. This means that:

- You will have to assign concrete types to columns, and choose things like `GENERATED`, `DEFAULT`, etc.
- You have to make assumptions about things like `NOT NULL`.
- You may have to decompose a table into two tables, or extract common attributes from multiple tables into a single, referenced table.
- Implementing the relationships may require adding columns and foreign keys, associative entities, etc.
- You may have to make other design and implementation choices. **This means that there is no single correct answer.**

- In addition to the key constraints, you must also implement the following constraints:
  - `email` values must contain a `@` and end in `.edu`
  - `semester` must be `fall`, `spring` or `summer`
  - `year` must be greater than or equal to 2021 and less than or equal to 2023.



ER Diagram

Answer

Design Decisions, Notes, etc. Document any assumption or decisions.

DDL

- Execute your DDL in the cell below. You may use DataGrip or other tools to help build the schema and statements.

- You can copy and paste the `SQL CREAT TABLE` below, but you MUST execute the statements.

In [141]:
```
# DDL in cells below.
# Use you UNI for the schema.

%sql create schema if not exists de2418_s22_midterm;
%sql select 1;
```

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

Out[141]:

1
1

In [142]:
```
%%sql
create table if not exists de2418_s22_midterm.course
(
    course_no   varchar(20) default '0' not null
        primary key,
    title       varchar(20)             null,
    description varchar(20)             null
);
```

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

Out[142]: []

In [143]:
```
%%sql
create table if not exists de2418_s22_midterm.department
(
    department_code varchar(20) default '0' not null
        primary key,
    department_name varchar(20)             null
);
```

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

Out[143]: []

In [144]:
```
%%sql
create table if not exists de2418_s22_midterm.faculty
(
    UNI             varchar(20) default '0' not null
        primary key,
    last_name       varchar(20)             null,
    first_name      varchar(20)             null,
    department_code varchar(20)             null,
    email           varchar(20)             null,
    constraint Faculty_email_uindex
        unique (email),
    constraint faculty_department_department_code_fk
        foreign key (department_code) references department (department_code),
    constraint chk_email check (email like '%_@__%.__%')
);
```

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

Out[144]:
```
[]
```

In [145]:
```
%%sql
create table if not exists de2418_s22_midterm.school
(
    school_code varchar(20) default '0' not null
        primary key,
    school_name varchar(20)             null
);
```

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

Out[145]:
```
[]
```

In [146]:
```
%%sql
create table if not exists de2418_s22_midterm.schooldepartment
(
    school_code     varchar(20) default '0' not null,
    department_code varchar(20) default '0' not null
        primary key,
    constraint schooldepartment_department_department_code_fk
        foreign key (department_code) references department (department_code),
    constraint schooldepartment_school_school_code_fk
        foreign key (school_code) references school (school_code)
);
```

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

Out[146]:
```
[]
```

In [147]:
```
%%sql
create table if not exists de2418_s22_midterm.section
(
    call_no       varchar(20) default '0' not null
        primary key,
    course_no     varchar(20)             null,
    section_no    varchar(20)             null,
    semester      varchar(20)             null,
    year          varchar(20)             null,
    instructor_id varchar(20)             null,
    constraint section_course_course_no_fk
        foreign key (course_no) references course (course_no),
    constraint section_faculty_UNI_fk
        foreign key (instructor_id) references faculty (UNI),
    constraint semester_check check (semester in ('fall', 'spring', 'summer')),
    constraint year_check check (year >= 2021 AND year <= 2023)
);
```

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

Out[147]: []

In [148]:
```
%%sql
create table if not exists de2418_s22_midterm.student
(
    UNI        varchar(20) default '0' not null
        primary key,
    last_name  varchar(20)             null,
    first_name varchar(20)             null,
    email      varchar(20)             null,
    constraint check_email check (email like '%_@__%.edu%')
);
```

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

Out[148]: []

In [149]:
```
%%sql
create table if not exists de2418_s22_midterm.studentsection
(
    UNI     varchar(20) default '0' not null,
    call_no varchar(20) default '0' not null,
    role    varchar(20)             null,
    primary key (call_no, UNI),
    constraint studentsection_section_call_no_fk
        foreign key (call_no) references section (call_no),
    constraint studentsection_student_UNI_fk
        foreign key (UNI) references student (UNI)
);
```

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

Out[149]:
```
[]
```

In [150]:
```
# SQL tests.
# Test your DDL with sample inserts/updates showing that you correctly implemente
#
%sql insert into de2418_s22_midterm.student(UNI, last_name, first_name, email) va
```

```
 * mysql+pymysql://root:***@localhost
(pymysql.err.OperationalError) (3819, "Check constraint 'check_email' is violat
ed.")
[SQL: insert into de2418_s22_midterm.student(UNI, last_name, first_name, email)
values("A","B","C","de2418columbia.edu")]
(Background on this error at: https://sqlalche.me/e/14/e3q8) (https://sqlalche.
me/e/14/e3q8))
```

In [151]:
```
%%sql
insert into
de2418_s22_midterm.section(call_no, section_no, semester,year)
values("B","B","spring","2020")
```

```
 * mysql+pymysql://root:***@localhost
(pymysql.err.OperationalError) (3819, "Check constraint 'year_check' is violate
d.")
[SQL: insert into
de2418_s22_midterm.section(call_no, section_no, semester,year)
values("B","B","spring","2020")]
(Background on this error at: https://sqlalche.me/e/14/e3q8) (https://sqlalche.
me/e/14/e3q8))
```

In [152]:
```
%%sql
insert into
de2418_s22_midterm.section(call_no, section_no, semester,year)
values("C","B","s","2021")
```

```
 * mysql+pymysql://root:***@localhost
(pymysql.err.OperationalError) (3819, "Check constraint 'semester_check' is vio
lated.")
[SQL: insert into
de2418_s22_midterm.section(call_no, section_no, semester,year)
values("C","B","s","2021")]
(Background on this error at: https://sqlalche.me/e/14/e3q8) (https://sqlalche.
me/e/14/e3q8))
```

In [153]:
```
%sql insert into de2418_s22_midterm.student(UNI, last_name, first_name, email) va
```

```
 * mysql+pymysql://root:***@localhost
(pymysql.err.OperationalError) (3819, "Check constraint 'check_email' is violat
ed.")
[SQL: insert into de2418_s22_midterm.student(UNI, last_name, first_name, email)
values("D","B","C","de2418@columbia.ed")]
(Background on this error at: https://sqlalche.me/e/14/e3q8) (https://sqlalche.
me/e/14/e3q8))
```

In [157]:
```
%sql insert into de2418_s22_midterm.student(UNI, last_name, first_name, email) va
```

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

Out[157]:
```
[]
```

In [158]:
```
%%sql
UPDATE de2418_s22_midterm.student
SET email = 'a'
WHERE UNI = "B";
```

```
 * mysql+pymysql://root:***@localhost
(pymysql.err.OperationalError) (3819, "Check constraint 'check_email' is violat
ed.")
[SQL: UPDATE de2418_s22_midterm.student
SET email = 'a'
WHERE UNI = "B";]
(Background on this error at: https://sqlalche.me/e/14/e3q8) (https://sqlalche.
me/e/14/e3q8))
```

SQL Queries

- You will use the Classic Models (https://www.mysqltutorial.org/mysql-sample-database.aspx) data for these questions.
- You loaded this database in a previous HW and tested that you have the database in the setup section.

S1

- Produce a table of the form:
  $(productLine, productName, productVendor, total\_revenue)$
- The contribution to total_value for an $orderLine$ is $quantityOrdered * priceEach$
- Total value is the sum of the $orderLine$ contributions for all $productCodes$ in a $productLine$.
- Only include results with $total\_value$ greater or equal to `$150000` and sorted by $total\_value$ descending.
- **NOTE:** You should be able to produce the answer without my providing the correct query output. I was giggling diabolically like the Riddler from Batman when writing the question. Then something like the following happened.

- So the output is below.

In [30]:
```
%%sql
drop table if exists classicmodels.midterm_s1;
create table if not exists classicmodels.midterm_s1
(
    productLine   varchar(50) null,
    productName   varchar(70) null,
    productVendor varchar(50) null,
    total_revenue char(20)    null
);
```

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

Out[30]:
```
[]
```

In [31]:
```
%%sql
drop table if exists classicmodels.temp;
create table if not exists classicmodels.temp
(
    orderLineNumber   varchar(50) null,
    productCode   varchar(70) null,
    perprocode varchar(50) null
);
```

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

Out[31]:
```
[]
```

In [32]:
```
%%sql
insert into classicmodels.temp(orderLineNumber, productCode, perprocode)
select orderLineNumber, productCode,sum(priceEach * QuantityOrdered) as perprocod
from classicmodels.orderdetails
group by productCode
```

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

Out[32]:
```
[]
```

In [33]:
```
%%sql
drop table if exists classicmodels.temp2;
create table if not exists classicmodels.temp2
(
    productLine   varchar(50) null,
    total_reve varchar(50) null
);
```

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

Out[33]: []

In [34]:
```
%%sql
insert into classicmodels.temp2 (productLine,total_reve)
select productLine, sum(perprocode) as total_revenue
from classicmodels.temp
join classicmodels.products
on classicmodels.temp.productCode = classicmodels.products.productCode
group by productLine
```

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

Out[34]: []

In [35]:
```
%%sql
drop table if exists classicmodels.temp3;
create table if not exists classicmodels.temp3
(
    productName   varchar(50) null,
    total_reve varchar(50) null
);
```

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

Out[35]: []

In [36]:
```
%%sql
insert into classicmodels.temp3 (productName,total_reve)
(select productName,total_reve from
classicmodels.products
left join
classicmodels.temp2
on classicmodels.products.productLine = classicmodels.temp2.productLine
where total_reve > 150000)
```

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

Out[36]: []

In [37]:
```
%%sql
insert into classicmodels.midterm_s1(productLine, productName, productVendor,tota
select productLine, p.productName, productVendor, total_reve
from classicmodels.products p
inner join classicmodels.temp3 t on p.productName = t.productName
```

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

Out[37]: []

In [38]:
```
%sql select * from classicmodels.midterm_s1 ORDER BY total_revenue DESC;
```

Classic Cars	1992 Ferrari 360 Spider red	Unimax Art Galleries	3853922.49
Classic Cars	1985 Toyota Supra	Highway 66 Mini Classics	3853922.49
Classic Cars	1969 Dodge Super Bee	Min Lin Diecast	3853922.49
Classic Cars	1976 Ford Gran Torino	Gearbox Collectibles	3853922.49
Classic Cars	1948 Porsche Type 356 Roadster	Gearbox Collectibles	3853922.49
Classic Cars	1970 Triumph Spitfire	Min Lin Diecast	3853922.49
Classic Cars	1957 Corvette Convertible	Classic Metal Creations	3853922.49
Classic Cars	1957 Ford Thunderbird	Studio M Art Models	3853922.49
Classic Cars	1970 Chevy Chevelle SS 454	Unimax Art Galleries	3853922.49
Classic Cars	1970 Dodge Coronet	Highway 66 Mini Classics	3853922.49
Classic Cars	1966 Shelby Cobra 427 S/C	Carousel DieCast Legends	3853922.49
Classic Cars	1949 Jaguar XK 120	Classic Metal Creations	3853922.49

- Put your query below. You do not need to create a view.

In [39]:

```
%%sql
drop table if exists classicmodels.midterm_s1;
create table if not exists classicmodels.midterm_s1
(
    productLine   varchar(50) null,
    productName   varchar(70) null,
    productVendor varchar(50) null,
    total_revenue char(20)    null
);

drop table if exists classicmodels.temp;
create table if not exists classicmodels.temp
(
    orderLineNumber   varchar(50) null,
    productCode   varchar(70) null,
    perprocode varchar(50) null
);

insert into classicmodels.temp(orderLineNumber, productCode, perprocode)
select orderLineNumber, productCode,sum(priceEach * QuantityOrdered) as perprococ
from classicmodels.orderdetails
group by productCode;

drop table if exists classicmodels.temp2;
create table if not exists classicmodels.temp2
(
    productLine   varchar(50) null,
    total_reve varchar(50) null
);

insert into classicmodels.temp2 (productLine,total_reve)
select productLine, sum(perprocode) as total_revenue
from classicmodels.temp
join classicmodels.products
on classicmodels.temp.productCode = classicmodels.products.productCode
group by productLine;

drop table if exists classicmodels.temp3;
create table if not exists classicmodels.temp3
(
    productName   varchar(50) null,
    total_reve varchar(50) null
);

insert into classicmodels.temp3 (productName,total_reve)
(select productName,total_reve from
classicmodels.products
left join
classicmodels.temp2
on classicmodels.products.productLine = classicmodels.temp2.productLine
where total_reve > 150000);

insert into classicmodels.midterm_s1(productLine, productName, productVendor,tota
select productLine, p.productName, productVendor, total_reve
from classicmodels.products p
inner join classicmodels.temp3 t on p.productName = t.productName;
```

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

Out[39]: []

S2

- Produce a table containing the rows from `products` for any products not in any `orderDetails`.

In [40]:
```
%%sql
select * from classicmodels.products
where not productCode in
(select productCode from classicmodels.orderDetails)
```

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

Out[40]:

productCode	productName	productLine	productScale	productVendor	productDescription	quantity
S18_3233	1985 Toyota Supra	Classic Cars	1:18	Highway 66 Mini Classics	This model features soft rubber tires, working steering, rubber mud guards, authentic Ford logos, detailed undercarriage, opening doors and hood, removable split rear gate, full size spare mounted in bed, detailed interior with opening glove box	

S3

- Define `order_time` to be the number of days between `orders.orderDate` and `orders.shippedDate`.

- The following tables has the form `(customerNumber, customerName, orderNumber, order_time)` for
  - Customers with `customers.country` in USA or Canada.
  - And the `orders.status` is not `Cancelled`.
- Your answer must match the sample below, including row order.

In [95]:
```
%%sql
drop table if exists classicmodels.midterm_s3;
create table if not exists classicmodels.midterm_s3
(
    customerNumber   TEXT null,
    customerName   TEXT null,
    orderNumber TEXT null,
    order_time TEXT    null
);
```

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

Out[95]:
```
[]
```

In [96]:
```
%%sql

use classicmodels;

insert into midterm_s3 (customerNumber, customerName,orderNumber,order_time)
select c.customerNumber, customerName, o.orderNumber, datediff(o.shippedDate,o.or
customers c
join orders o
on c.customerNumber = o.customerNumber
where(c.country in ('USA', 'Canada') and o.status != "Cancelled")
having order_time >=5 or order_time is Null
order by -order_time asc, orderNumber
```

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

Out[96]:
```
[]
```

In [97]:
```
%sql select * from classicmodels.midterm_s3;
```

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

Out[97]:

customerNumber	customerName	orderNumber	order_time
328	Tekni Collectables Inc.	10401	None
450	The Sharp Gifts Warehouse	10407	None
362	Gifts4AllAges.com	10414	None
124	Mini Gifts Distributors Ltd.	10421	None
157	Diecast Classics Inc.	10422	None
161	Technics Stores Inc.	10140	6
205	Toys4GrownUps.com	10145	6
219	Boards & Toys Co.	10154	6
321	Corporate Gift Ideas Co.	10159	6
347	Men 'R' US Retailers, Ltd.	10160	6
462	FunGiftIdeas.com	10166	6
175	Gift Depot Inc.	10172	6
124	Mini Gifts Distributors Ltd.	10182	6
320	Mini Creations Ltd.	10185	6
205	Toys4GrownUps.com	10189	6
328	Tekni Collectables Inc.	10251	6
204	Online Mini Collectables	10276	6
198	Auto-Moto Classics Inc.	10290	6
339	Classic Gift Ideas, Inc	10307	6
161	Technics Stores Inc.	10317	6
363	Online Diecast Creations Co.	10322	6
486	Motor Mint Distributors Inc.	10331	6
198	Auto-Moto Classics Inc.	10352	6
462	FunGiftIdeas.com	10388	6
129	Mini Wheels Co.	10111	5
198	Auto-Moto Classics Inc.	10130	5
447	Gift Ideas Corp.	10131	5
124	Mini Gifts Distributors Ltd.	10142	5
487	Signal Collectibles Ltd.	10149	5
363	Online Diecast Creations Co.	10192	5
455	Super Scale Inc.	10196	5
475	West Coast Collectables Co.	10199	5
239	Collectable Mini Designs Co.	10226	5

customerNumber	customerName	orderNumber	order_time
486	Motor Mint Distributors Inc.	10236	5
181	Vitachrome Inc.	10237	5
456	Microscale Inc.	10242	5
455	Super Scale Inc.	10245	5
233	Québec Home Shopping Network	10261	5
319	Mini Classics	10308	5
157	Diecast Classics Inc.	10318	5
424	Classic Legends Inc.	10337	5
161	Technics Stores Inc.	10362	5
124	Mini Gifts Distributors Ltd.	10368	5
219	Boards & Toys Co.	10376	5
124	Mini Gifts Distributors Ltd.	10396	5
233	Québec Home Shopping Network	10411	5

- Put your query below. You do not need to create a view.

In [98]:
```
%%sql

use classicmodels;

drop table if exists midterm_s3;
create table if not exists midterm_s3
(
    customerNumber   TEXT null,
    customerName   TEXT null,
    orderNumber TEXT null,
    order_time TEXT    null
);

insert into midterm_s3 (customerNumber, customerName,orderNumber,order_time)
select c.customerNumber, customerName, o.orderNumber, datediff(o.shippedDate,o.or
customers c
join orders o
on c.customerNumber = o.customerNumber
where(c.country in ('USA', 'Canada') and o.status != "Cancelled")
having order_time >=5 or order_time is Null
order by -order_time asc, orderNumber
```

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

Out[98]: []

S4

- In almost all cases, the `scale` of a `products` is encoded in the `productCode`.
- For example,

In [130]:
```
%sql select productCode, productScale from classicmodels.products limit 5;
```

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

Out[130]:

productCode	productScale
S10_1678	1:10
S10_1949	1:10
S10_2016	1:10
S10_4698	1:10
S10_4757	1:10

- There are, however, some cases where this is not true. Produce the following table.

In [131]:
```
%%sql
drop table if exists classicmodels.midterm_s4;
create table if not exists classicmodels.midterm_s4
(
    scale_in_product_scale   TEXT null,
    scale_in_product_Number   TEXT null,
    productCode TEXT null,
    productScale TEXT    null
);
```

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

Out[131]: []

In [132]:
```
%%sql
insert into classicmodels.midterm_s4 (scale_in_product_scale, scale_in_product_Nu
SELECT substring_index(productScale, ":",-1) as scale, substring_index(SUBSTRING_
productCode, productScale
FROM classicmodels.products;
```

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

Out[132]: []

In [133]:
```
%%sql
delete from classicmodels.midterm_s4
where scale_in_product_scale = scale_in_product_Number
```

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

Out[133]: []

In [134]:
```
%%sql
select * from classicmodels.midterm_s4;
```

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

Out[134]:

scale_in_product_scale	scale_in_product_Number	productCode	productScale
18	12	S12_3148	1:18
72	18	S18_2581	1:72
18	24	S24_3856	1:18
18	24	S24_4620	1:18
18	700	S700_2824	1:18
72	700	S700_3167	1:72

Data Loading and Cleanup

DL1

- There is a file `course_info.csv` in the directory/folder for the midterm.
- Do not ask how I got this information, but it looked something like this ... ...

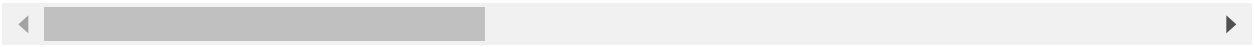
- The following code reads the JSON file and displays some of the data.

In [136]:
```
df = pd.read_csv("course_info.csv")
df.head()
```

Out[136]:

	Unnamed: 0	ExamDate	Term	ChargeAmt1	Meets4	TypeCode	SubtermCode	CampusCode	
0	0	NaN	20212	NaN	NaN	LC	NaN	MORN	AC(
1	1	NaN	20212	NaN	NaN	LC	NaN	MORN	AC
2	2	NaN	20212	NaN	NaN	LC	NaN	MORN	AC
3	3	NaN	20212	NaN	NaN	LC	NaN	MORN	AC
4	4	NaN	20212	NaN	NaN	LC	NaN	MORN	AC

5 rows × 45 columns



- The list of columns in the data is:

In [137]:
```
for c in df.columns:
    print(c)
```

```
Unnamed: 0
ExamDate
Term
ChargeAmt1
Meets4
TypeCode
SubtermCode
CampusCode
Course
Instructor3Name
ChargeAmt2
CallNumber
CourseTitle
NumFixedUnits
Instructor1Name
DivisionName
MaxSize
Instructor4Name
Meets2
SchoolCode
ChargeMsg2
SubtermName
Approval
BulletinFlags
PrefixName
PrefixLongname
EnrollmentStatus
CampusName
Meets5
ClassNotes
DivisionCode
ExamMeet
DepartmentName
NumEnrolled
Meets3
MaxUnits
SchoolName
Instructor2Name
Meets6
MinUnits
DepartmentCode
ChargeMsg1
TypeName
CourseSubtitle
Meets1
```

- And you can find out the meaning of the columns on [CU's Open Data site. (https://opendataservice.columbia.edu/doc)](https://opendataservice.columbia.edu/doc)

"Term": Five digit term code, year then semester 1=Spring, 2=Summer, 3=Fall. (e.g. 20131 = Spring 2013)
"Course": Subject, Course and Section number
"PrefixName":
"DivisionCode": Code of the division providing the course
"DivisionName": Name of the division providing the course
"CampusCode": Code of the campus the course is on
"CampusName": Name of the campus the course is on
"SchoolCode": Code of the school providing the course
"SchoolName": Name of the school providing the course
"DepartmentCode": Code of the department providing the course
"DepartmentName": Name of the department providing the course
"SubtermCode":
"SubtermName":
"CallNumber": Registration call number of the couse
"NumEnrolled": number currently enrolled
"MaxSize": max enrollment size
"EnrollmentStatus": O=Open C=Closed
"NumFixedUnits": default credits for the course
"MinUnits":
"MaxUnits":
"CourseTitle": Title of the couse
"CourseSubtitle": Subtitle of the course
"TypeCode": "LC",
"TypeName": "LECTURE",
"Approval": if approval is needed
"BulletinFlags": "B",
"ClassNotes": "",
"Meets1": "",
"Meets2": "",
"Meets3": "",
"Meets4": "",
"Meets5": "",
"Meets6": "",
"Instructor1Name": "NISSIM, DORON",
"Instructor2Name": "",
"Instructor3Name": "",
"Instructor4Name": "",
"PrefixLongname": "ACCOUNTING",
"ExamMeet": "",
"ExamDate": "",
"ChargeMsg1": "",
"ChargeAmt1": "",
"ChargeMsg2": "",
"ChargeAmt2": ""

- Now, while laughing diabolically, I had started to write a question asking you to create a schema, factor the data into multiple tables, clean up the data and set types, set keys and

schema, factor the data into multiple tables, clean up the data and set types, set keys and constraints, etc.

- Performing the work to answer this question took me two hours. I find that students typical need 10X as much time as I need. The diabolical Riddler laughter really took off when I was done.
- But, something a lot like this happened. ... ...

- So, I deleted the question and decided to make it one of the extra-credit assignments later in the semester.
- Do not worry about this question now. I will publish as extra-credit later in the semester.

Putting it All Together

A1

- Sadly for you, Batman seems to think I learned my lesson and can be trusted to define reasonable questions. We all know that is not true. So, when you think about it, this horrible question is at least partly Batman's fault.
- The midterm directory contains a file `people_info.csv`. The columns are:
  - `first_name`
  - `middle_name`
  - `last_name`
  - `email`
  - `employee_type`, which can be one of `Professor, Lecturer, Staff.` The value is empty of the person is a student.
  - `enrollment_year` which must be in the range `2016 - 2022.` The value is empty of the person is an employee.

- If `enrollment_year` is NULL, the person is a `Student` and `employee_type` must be NULL. If `employee_type` is not NULL, then the person is NOT a student and `enrollment_year` must be NULL.

- You must implement a [two-table (https://vertabelo.com/blog/inheritance-in-a-relational-database/)](https://vertabelo.com/blog/inheritance-in-a-relational-database/) solution to the inheritance pattern. This means that your solution will have tables `students` and `employees,` and will have a view `people.`

- Your solution must implement the following tasks and meet the following criteria.
    1. You must implement the two tables and views, including reasonable data types and constraints. This must include an additional column `uni` that we explain below. You may have additional columns if that helps.
    2. You must implement four stored procedures. The implementations of the four procedures are almost identical.
        A. The procedures are:
            a. create_student(first_name, middle_name, last_name, email, enrollment_year, uni, guid)
            b. create_employee(first_name, middle_name, last_name, email, employee_type, uni, guid)
            c. update_student(uni, first_name, middle_name, last_name, email, enrollment_year)
            d. update_employee(uni, first_name, middle_name, last_name, email, employee_type)
        B. For updates, the procedures ignore input parameters that are NULL and only apply the non-NULL values. The procedure does not update the `uni.`
        C. The create procedures must generate the value for the `GUID` and `uni` and return them as out values.
    3. `employee_type` must be one of `Professor, Lecturer, Staff.`
    4. `enrollment_year` must be in the range `2016 - 2022` inclusive.
    5. Only `middle_name` can be null.
    6. uni:
        A. Must be of the form "FMLnnnn" where "F" is the first letter of the first name, "M" is the first letter of the middle_name (if not NULL), "L" is the first letter of the last name.
        B. For any combination of letters, the numbers following the letters must start at 1 and increase. That is "DFF1", "AB1", "DFF2", "CD1", "CAD1", "CD2", ...
        C. You must implement a function to generate the `uni` and a trigger on the relevant tables to automatically set the `uni` and `GUID` on INSERT. You must implement a trigger that prevents changing the `uni` or `GUID.`
    7. `email` must be unique over both `student` and `employee.`
    8. You must use security to disable calling `INSERT, DELETE` and `UPDATE` on the underlying tables for any user other than `root` or `dbuser.` You must test this by creating an additional user `general_user` and connecting as that user.

- You must demonstrate correct implementation by loading the data, using select statements, attempting insert/update/delete, ...

- The following code will load the tables for you if you have properly implemented the tables and functions.

- The execution also helps you understand how to test procedures, etc.

In [49]:
```
%%sql

create schema if not exists de2418_s22_midterm;
```

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

Out[49]: []

In [166]:
```
%%sql
drop table if exists de2418_s22_midterm.students;
create table if not exists de2418_s22_midterm.students
(
    uni             text null,
    guid            text null,
    first_name      text null,
    middle_name     text null,
    last_name       text null,
    email           text null,
    employee_type   text null,
    enrollment_year text null,
    constraint employ_type_check check (employee_type is Null),
    constraint enrollment_year_check check (enrollment_year >= 2016 AND enrollmer
);
```

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

Out[166]: []

In [167]:
```
%%sql
drop table if exists de2418_s22_midterm.employees;
create table if not exists de2418_s22_midterm.employees
(
    uni             text null,
    guid            text null,
    first_name      text null,
    middle_name     text null,
    last_name       text null,
    email           text null,
    employee_type   text null,
    enrollment_year text null,
    constraint employee_type_check check (employee_type in ('Professor', 'Lecture
    constraint enroll_year_check check (enrollment_year is Null)
);
```

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

Out[167]: []

In [169]:
```
%%sql
create function generatehalfuni(in first_name TEXT,
                                in middle_name TEXT,
                                in last_name TEXT,
                                in email TEXT)
    returns TEXT
    begin
    declare huni TEXT;
        select concat(substring(first_name, 0, 1), substring(middle_name, 0, 1),
    return huni
end
```

```
 * mysql+pymysql://root:***@localhost
(pymysql.err.ProgrammingError) (1064, "You have an error in your SQL syntax; ch
eck the manual that corresponds to your MySQL server version for the right synt
ax to use near 'in first_name TEXT, \n                                  in middle
_name TEXT,\n      ' at line 1")
[SQL: create function generatehalfuni(in first_name TEXT,
                                in middle_name TEXT,
                                in last_name TEXT,
                                in email TEXT)
    returns TEXT
    begin
    declare huni TEXT;
        select concat(substring(first_name, 0, 1), substring(middle_name, 0,
1), substring(last_name, 0, 1)) into huni
    return huni
end]
(Background on this error at: https://sqlalche.me/e/14/f405) (https://sqlalche.
me/e/14/f405))
```

In [53]:
```
# The following only works if you have already created the tables, procedures, et
# So, you mus have answered the questions to run the load.
#
%sql use dff9_s22_midterm;
%sql delete from student;
%sql delete from employee;
%sql select 1;
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
 * mysql+pymysql://root:***@localhost
(pymysql.err.ProgrammingError) (1146, "Table 'dff9_s22_midterm.student' doesn't
exist")
[SQL: delete from student;]
(Background on this error at: https://sqlalche.me/e/14/f405) (https://sqlalche.
me/e/14/f405))
 * mysql+pymysql://root:***@localhost
(pymysql.err.ProgrammingError) (1146, "Table 'dff9_s22_midterm.employee' does
n't exist")
[SQL: delete from employee;]
(Background on this error at: https://sqlalche.me/e/14/f405) (https://sqlalche.
me/e/14/f405))
 * mysql+pymysql://root:***@localhost
1 rows affected.
```

Out[53]:

1
1

In [54]:
```
# Load the information from the file and show sample values.
import csv

people_info = []
with open("people_info.csv", "r") as in_file:
    d_rdr = csv.DictReader(in_file)
    for r in d_rdr:
        people_info.append(dict(r))

people_info_df = pd.DataFrame(people_info)
people_info_df
```

Out[54]:

	first_name	middle_name	last_name	email	employee_type	enrollment_
0	Sanders	Arline	Breckell	abreckell1x@fotki.com	Professor	
1	Zared		Fenelon	afenelona@themeforest.net		
2	Ethelin		Fidele	afidele12@google.ru	Lecturer	
3	Bibbye	Annabal	Guesford	aguesfordb@tumblr.com		
4	Xenia	Ardella	Kief	akieft@free.fr	Staff	
...	...	...	...	...	...	
95	Norry		Rubinchik	trubinchik16@howstuffworks.com		
96	Doug		Medforth	vmedforth1o@homestead.com	Staff	
97	Gerty		O'Donegan	vodoneganf@clickbank.net		
98	Anabelle	Wallas	Quimby	wquimby1c@nba.com		
99	Sasha	Win	Ruffli	wruffli2q@wordpress.com	Lecturer	

100 rows × 6 columns

In [55]:
```
people_info[-5:]
```

Out[55]:
```
[{'first_name': 'Norry',
  'middle_name': '',
  'last_name': 'Rubinchik',
  'email': 'trubinchik16@howstuffworks.com',
  'employee_type': '',
  'enrollment_year': '2016'},
 {'first_name': 'Doug',
  'middle_name': '',
  'last_name': 'Medforth',
  'email': 'vmedforth1o@homestead.com',
  'employee_type': 'Staff',
  'enrollment_year': ''},
 {'first_name': 'Gerty',
  'middle_name': '',
  'last_name': "O'Donegan",
  'email': 'vodoneganf@clickbank.net',
  'employee_type': '',
  'enrollment_year': '2020'},
 {'first_name': 'Anabelle',
  'middle_name': 'Wallas',
  'last_name': 'Quimby',
  'email': 'wquimby1c@nba.com',
  'employee_type': '',
  'enrollment_year': '2022'},
 {'first_name': 'Sasha',
  'middle_name': 'Win',
  'last_name': 'Ruffli',
  'email': 'wruffli2q@wordpress.com',
  'employee_type': 'Lecturer',
  'enrollment_year': ''}]
```

In [ ]:

In [56]:
```
# If you have defined your procedures, functions, tables and constraints correctl
# to load your data.
#
import copy

def add_person(p):
    """
    p is a dictionary containing the column values for either a student or an emp
    """

    cur = sql_conn.cursor()

    # This function changes the data, converting '' to None.
    # So, we make a copy and change the copy.
    p_dict = copy.copy(p)
    for k,v in p_dict.items():
        if v == '':
            p_dict[k] = None

    # Is the person a student?
    #
    if p_dict['employee_type'] is None:

        # This provides a hint for what your stored procedure will look like.
        res = cur.callproc("dff9_s22_midterm.create_student",
                           # The following are in parameters
                           (p_dict['first_name'],
                           p_dict['middle_name'],
                           p_dict['last_name'],
                            p_dict['email'],
                           p_dict['enrollment_year'],
                            # The following are out parameters for uni and GUID.
                           None,
                           None))

        # After the procedure executes, the following query will select the out v
        res = cur.execute("""SELECT @_dff9_s22_midterm.create_student_5,
                                @_dff9_s22_midterm.create_student_6""")
        result = cur.fetchall()

    # The following does the same for employee.
    elif p_dict['enrollment_year'] is None:
        res = cur.callproc("dff9_s22_midterm.create_employee",
                           (p_dict['first_name'],
                           p_dict['middle_name'],
                           p_dict['last_name'],
                            p_dict['email'],
                           p_dict['employee_type'],
                           None,
                           None))
        res = cur.execute("""SELECT @_dff9_s22_midterm.create_student_5,
                                @_dff9_s22_midterm.create_student_6""")
        result = cur.fetchall()
    else:
        print("Huh")
        result = None
```

```
    sql_conn.commit()
    cur.close()
    return result
```

In [57]:
```
for p in people_info:
    add_person(p)
```

```
---------------------------------------------------------------------------
OperationalError                          Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_22164/1791746634.py in <module>
      1 for p in people_info:
----> 2     add_person(p)

~\AppData\Local\Temp/ipykernel_22164/1189769791.py in add_person(p)
     41     # The following does the same for employee.
     42     elif p_dict['enrollment_year'] is None:
---> 43         res = cur.callproc("dff9_s22_midterm.create_employee",
     44                            (p_dict['first_name'],
     45                            p_dict['middle_name'],

~\anaconda3\lib\site-packages\pymysql\cursors.py in callproc(self, procname, ar
gs)
    256             ",".join(["@_%s_%d" % (procname, i) for i in range(len(args
))]),
    257         )
--> 258         self._query(q)
    259         self._executed = q
    260         return args

~\anaconda3\lib\site-packages\pymysql\cursors.py in _query(self, q)
    308         self._last_executed = q
    309         self._clear_result()
--> 310         conn.query(q)
    311         self._do_get_result()
    312         return self.rowcount

~\anaconda3\lib\site-packages\pymysql\connections.py in query(self, sql, unbuff
ered)
    546             sql = sql.encode(self.encoding, "surrogateescape")
    547         self._execute_command(COMMAND.COM_QUERY, sql)
--> 548         self._affected_rows = self._read_query_result(unbuffered=unbuff
ered)
    549         return self._affected_rows
    550

~\anaconda3\lib\site-packages\pymysql\connections.py in _read_query_result(sel
f, unbuffered)
    773         else:
    774             result = MySQLResult(self)
--> 775             result.read()
    776         self._result = result
    777         if result.server_status is not None:

~\anaconda3\lib\site-packages\pymysql\connections.py in read(self)
   1154     def read(self):
   1155         try:
-> 1156             first_packet = self.connection._read_packet()
   1157
   1158             if first_packet.is_ok_packet():
```

```
~\anaconda3\lib\site-packages\pymysql\connections.py in _read_packet(self, packet_type)
    723             if self._result is not None and self._result.unbuffered_active is True:
    724                 self._result.unbuffered_active = False
--> 725             packet.raise_for_error()
    726         return packet
    727

~\anaconda3\lib\site-packages\pymysql\protocol.py in raise_for_error(self)
    219         if DEBUG:
    220             print("errno =", errno)
--> 221         err.raise_mysql_exception(self._data)
    222
    223     def dump(self):

~\anaconda3\lib\site-packages\pymysql\err.py in raise_mysql_exception(data)
    141     if errorclass is None:
    142         errorclass = InternalError if errno < 1000 else OperationalError
--> 143     raise errorclass(errno, errval)

OperationalError: (1305, 'PROCEDURE dff9_s22_midterm.create_employee does not exist')
```

In [ ]:
```
#
# Test that we loaded people.
#
%sql select * from people;
```

In [ ]:
```
#
# Test that we loaded students.
#
%sql select * from student;
```

In [ ]:
```
#
# Test that we loaded employees.
#
%sql select * from employee;
```

Your tests should be below. You should have test cells that test:

1. Successful execution of each procedure.

2. Execution of update procedures showing that your constraints prevent incorrect data entry and enforce the defined semantics and behavior.

3. Demonstrating that:
   A. You create a user `general_user`.
   B. `general_user` can query the data and call the procedures, but cannot perform `INSERT, UPDATE, DELETE`.

In [ ]:

In [ ]: