Environments

- *Different applications and different scenarios use different ways to interaction with databases.*
- *We use three different connection/interaction models to give students experience with the various options.*

ipython-SQL

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

In [2]:
```
#
# Set the userid and password for connecting to your instance of SQL.
#
mysql_user = "root"
mysql_password = "dbuserbdbuser"

mysql_url = f"mysql+pymysql://{mysql_user}:{mysql_password}@localhost"

print("Your connection URL is", mysql_url)
```

```
Your connection URL is mysql+pymysql://root:dbuserbdbuser@localhost
```

In [3]:
```
#
# Connect. See the ipython-sql documentation for the $variable syntax.
#
%sql $mysql_url
```

SQL Alchemy and Pandas

In [4]:
```
#
# Yes, I know the cool kids import as pd. I am not cool.
#
import pandas
```

In [5]:
```
#
# Pandas SQL operations require a SQL Alchemy engine.
#
from sqlalchemy import create_engine
```

In [6]:
```
sql_engine = create_engine(mysql_url)
```

pymysql

In [7]:
```
import pymysql
```

In [8]:
```
pymysql_con = pymysql.connect(
    user= mysql_user,
    password= mysql_password,
    host= "localhost",
    port= 3306,
```

```
    autocommit= True,
    cursorclass= pymysql.cursors.DictCursor)
```

Data Loading

Classic Models

- *We will use the Classic Models sample database for many of the questions on this exam.*
- *The directory containing this notebook contains a file* `classic-models-sample.sql`.
- *Load the data:*
  - *Open the file in DataGrip using* `File -> Open` *dialog.*
  - *Select all of the text/SQL in the file.*
  - *Click the green arrowhead to run the files contents.*
- *Running the following queries will test if the load worked.*

In [9]:
```
%sql use classicmodels;
```

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

Out[9]:
```
[]
```

In [10]:
```
%sql show tables;
```

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

Out[10]:

Tables_in_classicmodels
customers
employees
offices
orderdetails
orders
payments
productlines
products

In [11]:
```
%sql select count(*) as count from orders join orderdetails using(orderNumber)
```

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

Out[11]:

count
2996

Lahman's Baseball Database

- *You previously loaded information from Lahman's Baseball Database.*
- *If you have not done so, the following code will load the data into a new schema* `lahmansdb_midterm`.

In [12]:
```
%sql create schema lahmansdb_midterm
```

```
 * mysql+pymysql://root:***@localhost
(pymysql.err.ProgrammingError) (1007, "Can't create database 'lahmansdb_midterm'; dat
abase exists")
[SQL: create schema lahmansdb_midterm]
(Background on this error at: https://sqlalche.me/e/14/f405)
```

In [13]:
```
people_df = pandas.read_csv("./People.csv")
people_df.to_sql("people", schema="lahmansdb_midterm", con=sql_engine,index=False, if_
```

Out[13]: 20370

In [14]:
```
batting_df = pandas.read_csv("./Batting.csv")
batting_df.to_sql("batting", schema="lahmansdb_midterm", con=sql_engine,index=False, 
```

Out[14]: 110495

In [15]:
```
pitching_df = pandas.read_csv("./Pitching.csv")
pitching_df.to_sql("pitching", schema="lahmansdb_midterm", con=sql_engine,index=False,
```

Out[15]: 49430

- *This will test the data loading.*

In [16]:
```
%sql select count(*) as people_count from lahmansdb_midterm.people;
```

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

Out[16]:

people_count
20370

In [17]:
```
%sql select count(*) as batting_count from lahmansdb_midterm.batting;
```

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

Out[17]:

batting_count
110495

In [18]:
```
%sql select count(*) as pitching_count from lahmansdb_midterm.pitching;
```

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

Out[18]:

pitching_count
49430

Written Questions

W1

Question

- *Define the concept of immutable column and key.*
- *Why do some sources recommend that a primary key should be immutable?*
- *How would to implement immutability for a primary key in a table?*

Answer

***Immutable column**: a column in a database table that cannot be updated or deleted once the value is assigned.*

***Immutable key**: value of a primary key or unique key cannot be modified or deleted.*

***Primary key**: is unqiue representation of each rows. When tables are connected, referential integrity and data consisency need to be enforced.*

***Implement**: by setting column as IMMUTABLE PRIMARY KEY.*

```
CREATE TABLE student (
    student_id varchar(128) IMMUTABLE PRIMARY KEY
    )
```

W2

Question

Views are a powerful concept in relational database management systems. List and briefly explain 3 benefits of/reasons for creating a view.

Answer

***1**: Due to **securtity concerns**, view can be personalized collection of virtual relations.*

***2**: It's possible to **support a large number** of views on the top of actual relations, because it's not precomputed and stored. Take place of with clause.*

***3**: Database system stored view as definition. Therefore, content of view is recomputed when we query and **not out of date**.*

W3

Question

Briefly explain the concepts of procedural language and declarative language. SQL is primarily a declarative language. SQL added procedure language capabilities in functions, procedures and triggers. What is a reason for this addition?

Answer

***Procedural DML**: specify what need and how to get those data.*

***Declarative DML**:specify what need without specifying how to get those data.*

***capabilities in functions, procedures, and triggers**: They are more advanced features of SQL and taken for more complexed operations. They provide access to general-purpose programming language, like write reusable code, perform transformation, enforce data consistency, etc.*

W4

Question

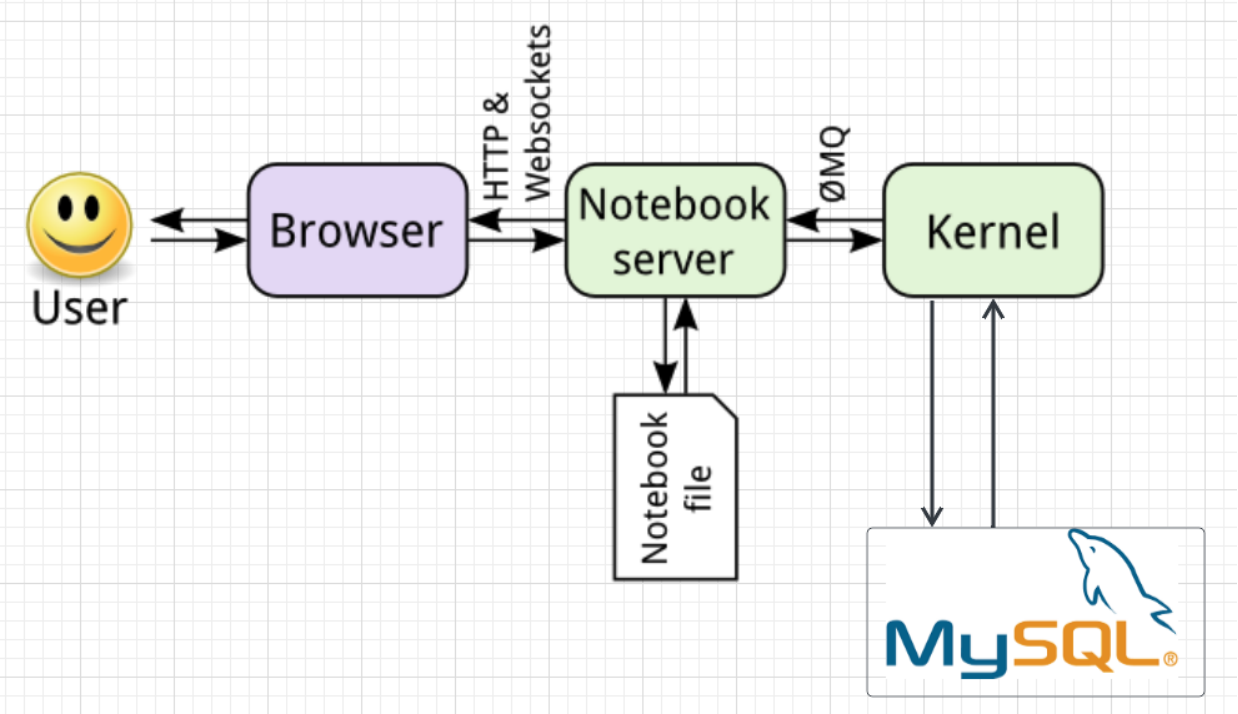
The following diagram is a simple representation of the architecture of a Jupyter notebook using MySQL. Is this a two-tier architecture or a three-tier architecture? Explain your answer briefly.

 *#this img can run, but cannot show after export html working, thus, I use import Image instead.*

In [19]:
```
#above <img src="./jupyter-notebook.png"> can run, but cannot show after export html w
from IPython.display import Image

Image("jupyter-notebook.png")
```

Out[19]:



Answer

***Three-tier achitecture**: web browser as front-end communicate with application server(notebook server). Notebook server communicate with mysql to access the data. Web browser does not contain any direct database calls.*

W5

Question

- *Consider a US Social Security Number. An example is "012-34-6789".*
- *The data type is character string.*
- *The relational model requires that columns (attributes) are from a domain.*
- *Use the Social Security Number example to explain the difference between a type and a domain.*

Answer

***Type** of SSN define it's a string with fixed length of nine.*

***Domain** acts as contraint on values that it can take to enforce referential integrity and authorization. SSN is unique with fixed 3-2-4 number pattern. People with alaries must own SSN. Not all users can read, insert, update, delete information of SSN in the tables. Domain can differentiate among the users.*

W6

Question

Briefly explain the differences between:

- *Database stored procedure*
- *Database function*
- *Database trigger*

Answer

Procedure

- *Can change data.*
- *Sometime return value by use in and out argument.*
- *Call to be executed repeatedly. Not automatic.*
- *Allow same name as long as arguments are different.*

Function

- *Cannot change data.*
- *Always return a table or a number.*
- *Called with prefixed name with inputs in a statement. Not automatic.*
- *Allow same name as long as arguments are different.*

Trigger

- *Can change data.*
- *Not return value.*
- *Carried out automatically on certain events such as insertion, deletion, or update in a specified relation.*
- *In different schemas can have same name.*

W7

Question

Briefly explain:

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

Answer

***Natural join**: match tuples with same value for all matched columns, and only keep one copy of replicated columns.*

***Equi-join**: return tuples from tables when values in the matched columns are equal.*

***Theta join**: use comparison operator other than equality to match tuples from tables.*

***Self-join**: join with self. Able to put one table in two distinct tables like employee and manager. Processing hierarchy. Generating pairs of rows based on condition.*

W8

Question

Briefly explain the difference between a unique (key) constraint and a primary key constraint?

Answer

***Primary key**: uniquely identify rows, not null.*

***Unqiue key**: a column or a set of columns are unique, not need to be primary identifier, allow null.*

W9

Question

Give two reasons for using an associative entity to implement a relationship instead of using a foreign key.

Answer

***1**: when foreign key is not sufficient to establish relationship between two entities, associative can used as additional attributes stored and associated with the relationship.*

***2**: to implement many to many relationship between two or more entities.*

W10

Question

Briefly explain the concepts of:

- *Conceptual model*
- *Logical model*
- *Physical model*

For data modeling.

Answer

***Conceptual model**: establish high-level, static business structures and concepts.*

***Logical model**: define entity types, data attributes and relationships between entities.*

***Physical model**: internal schema database design with tables, columns, keys, and indexes.*

W11

Question

Briefly explain the concepts of:

- *Data manipulation language*
- *Data definition language*

Given an example statement in SQL for DML and for DDL.

Answer

***Data manipulation language**: manipulate data in a database. EXA, insert, update, delete, retrieve data.*

```
insert into s23_w4111_hw2_bq2150.name_basics_all (nconst,
primaryName, birthYear) values ('nm0203893', 'Ton Ha', '1960')
```

***Data definition language**: manage database structure. EXA, create, modify, delete database schema(table, index).*

```
DROP TABLE name_basics_all
```

W12

Question

Codd's 4th rule is:

Rule 4 - Dynamic online catalog based on the relational model:

The data base description is represented at the logical level in the same way as ordinary data, so that authorized users can apply the same relational language to its interrogation as they apply query to the regular data.

Explain what this means, and use SQL to provide examples.

Answer

Database schema, stored in the same relational database system, should be accessible to authorized users in the same way as the regular data. Example below.

```
SELECT * FROM INFORMATION_SCHEMA.COLUMNS
where TABLE_SCHEMA='s23_w4111_hw2_bq2150' And
TABLE_NAME='name_basics_all';
```

W13

Question

The formal definition of a theta join is

$$r \bowtie_\theta s = \sigma_\theta(r \times s) \tag{1}$$

Briefly explain the definition and give an example.

Why is the fact that the relational algebra is closed is important to this definition? /

Answer

EXPLAIN: Result will include all columns from r and s, and rows that satisfy theta condition.

EXA:In cartesian-product of r × s, attribute names can only come from instructor and teaches.

Thus, it satisfies closed that result of operators is a defined set and produced from the element within the same set.

W14

Question

Consider two different statements in the relational algebra or SQL.

Despite being different statements, the statements may be equivalent. Briefly explain what this means.

Answer

We use different syntax, but result in the same output.

For example, in HW2P2, "Please write an equivalent query that does not use anti-join".

W15

Question

Consider the following relation definitions.

$$Customers(ID, last\_name, first\_name) \tag{2}$$

$$Accounts(ID, balance, customer\_ID) \tag{3}$$

What is problem with using natural join on the two tables?

Answer

Accounts.ID are view as same column with Customers.ID to pair up in natural join, which is actually not correct.

Accounts.customers_ID will be the correct column to join. We can clarify it by add ON statement

Entity Relationship Modeling

ER-1

Question

This question tests your ability to "bottom up" model or "reverse engineering" a SQL schema to produce an explanatory ER-diagram.

Use Lucidchart to draw a Crow's Foot notation diagram representing the following SQL.

You can use the simple table names, e.g. `students` *instead of* `s23_w4111_midterm.students`.

```
drop schema if exists s23_midterm;

create schema s23_midterm;

use s23_midterm;

drop table if exists departments;
create table if not exists departments
(
    dept_code varchar(4)   not null
        primary key,
    dept_name varchar(128) not null
);

drop table if exists instructors;
create table if not exists instructors
(
    UNI        varchar(12)  not null
```

```
        primary key,
    last_name  varchar(128) not null,
    first_name varchar(128) not null,
    dept_code  varchar(4)   null,
    constraint instructor_dept
        foreign key (dept_code) references departments (dept_code)
);

drop table if exists students;
create table if not exists students
(
    UNI        varchar(12)  not null
        primary key,
    last_name  varchar(128) null,
    first_name varchar(128) null
);

drop table if exists students_advisors;
create table if not exists students_advisors
(
    student_uni         varchar(12) not null,
    instructor_uni      varchar(12) not null,
    advising_start_date date        not null,
    advising_end_date   date        null,
    primary key (student_uni, instructor_uni, advising_start_date),
    constraint student_advisor_instructor
        foreign key (instructor_uni) references instructors (UNI),
    constraint student_advisors_student
        foreign key (student_uni) references students (UNI)
);
```

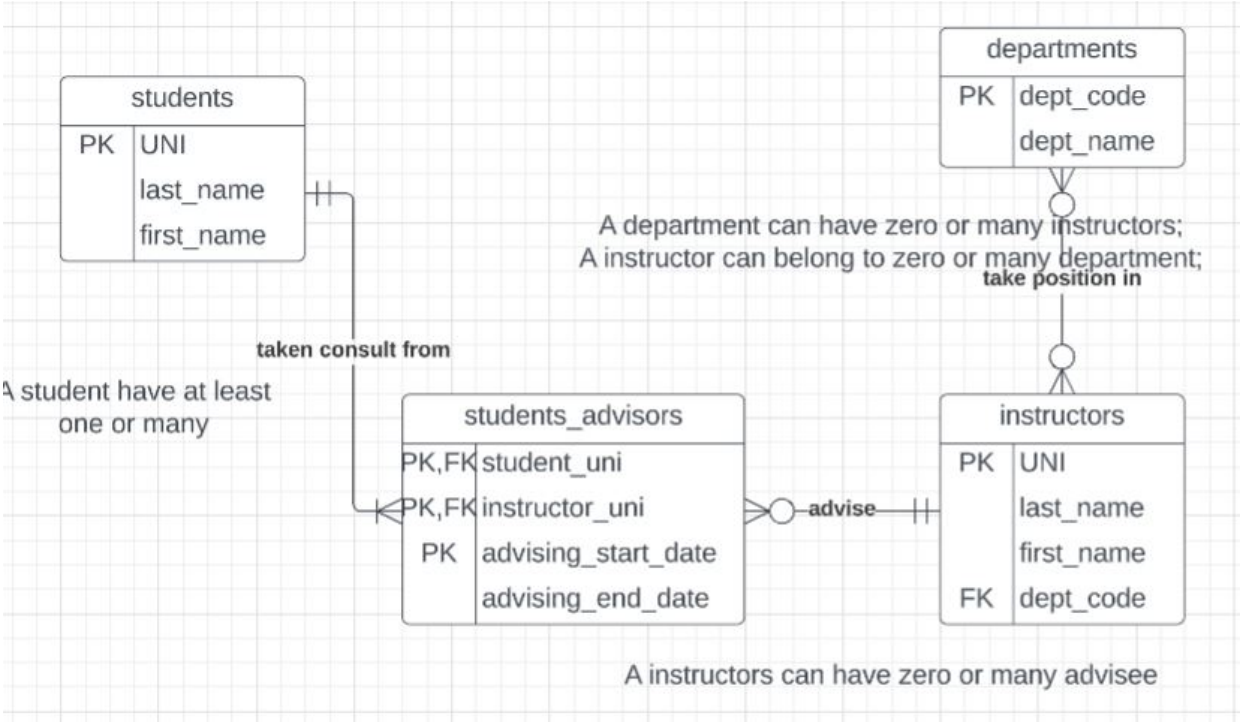
Answer

- *Put your screen capture in the same directory as the midterm.*
- *Add* `<img src='./myfile.name'>` *in the Markdown cell, using the actual file name.*

In [20]:
```
#above <img src="./jupyter-notebook.png"> can run, but cannot show after export html,
from IPython.display import Image

Image("ER1.jpg")
```

Out[20]:



ER-2

Question

- *This question tests your ability to convert a human language description of a data model into a Crow's Foot ER-Diagram.*
- *Consider the data model for Classic Models that you loaded.*
- `orders` *has a column* `comments.`

In [21]:
```
%sql select * from classicmodels.orders limit 10;
```

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

Out[21]:

orderNumber	orderDate	requiredDate	shippedDate	status	comments	customerNumber
10100	2003-01-06	2003-01-13	2003-01-10	Shipped	None	363
10101	2003-01-09	2003-01-18	2003-01-11	Shipped	Check on availability.	128
10102	2003-01-10	2003-01-18	2003-01-14	Shipped	None	181
10103	2003-01-29	2003-02-07	2003-02-02	Shipped	None	121
10104	2003-01-31	2003-02-09	2003-02-01	Shipped	None	141
10105	2003-02-11	2003-02-21	2003-02-12	Shipped	None	145
10106	2003-02-17	2003-02-24	2003-02-21	Shipped	None	278
10107	2003-02-24	2003-03-03	2003-02-26	Shipped	Difficult to negotiate with customer. We need more marketing materials	131
10108	2003-03-03	2003-03-12	2003-03-08	Shipped	None	385
10109	2003-03-10	2003-03-19	2003-03-11	Shipped	Customer requested that FedEx Ground is used for this shipping	486

- *There are several issues with this design:*
  - *If there are multiple comments or responses to comments, the* `comments` *field becomes multi-valued.*
  - *The approach does not have information on when the comment was made, who made the comment and whether it is a response or elaboration.*

- *You will solve this problem in a simplified version of classic models. In the simplified model, there are three entity types:*
  1. `person` *has the following attributes:*
     - `ID`
     - `last_name`
     - `first_name`
  2. `orders` *has the following attributes:*
     - `order_id`
     - `order_status`
     - `order_date`
  3. `comments` *has the following attributes:*

- *`comment_id` is a unique ID for all comments.*
- *`parent_comment_id` is the `comment_id` of a comment for which this comment is a response or elaboration.*
- *`comment_timestamp`, when the comment occured.*
- *`commenter_id` is the ID of the person making the comment.*
- *`order_id` is the ID of the order for to which this comment applies.*

- *Use Lucidchart to draw a logical model for the described datamodel.*
- *You may add notes to the diagram to document reasonable assumptions or desgn decisions.*

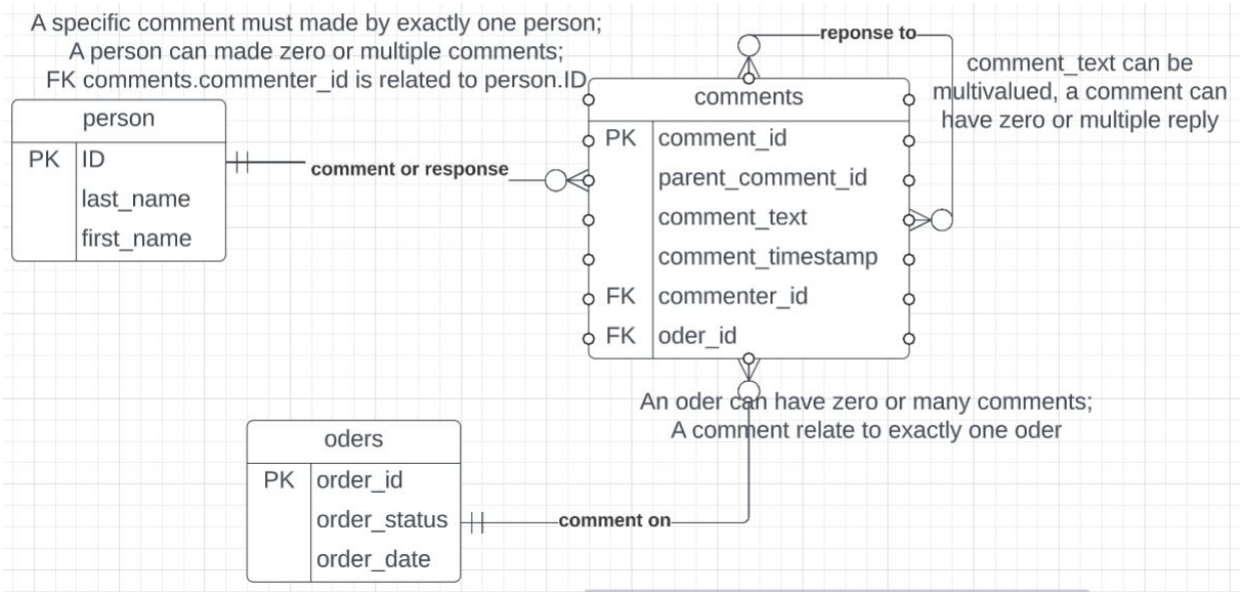
Answer

- *Put your screen capture in the same directory as the midterm.*
- *Add `<img src='./myfile.name'>` in the Markdown cell, using the actual file name.*

In [22]:
```
#above <img src="./jupyter-notebook.png"> can run, but cannot show after export html,
from IPython.display import Image

Image("ER3.jpg")
```

Out[22]:



Relational Algebra

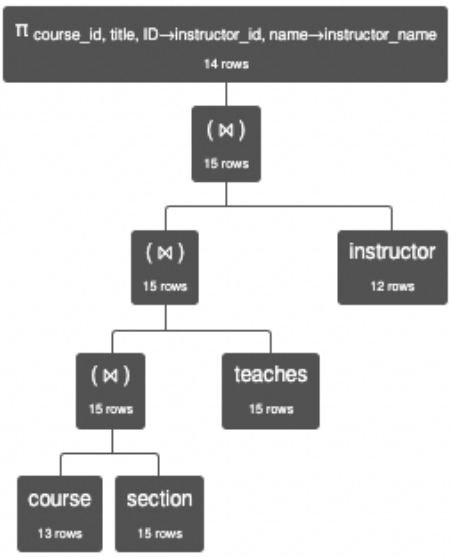
- *Use the RelaX Calculator and the Silberschatz calculator with the Silberschatz database for these questions.*
- *Your answers will have two Markdown cells. The first is the relational statement you used to solve the problem. The second is a screen capture of the query execution and first page of result rows. And example is:*

```
π course_id, title,
    instructor_id←ID,
    instructor_name←name
(
    (
        (course ⋈ section)
        ⋈
        teaches
    )
    ⋈
    instructor
)
```

In [23]:
```
#above <img src="./jupyter-notebook.png"> can run, but cannot show after export html w
from IPython.display import Image

Image("relational-example.png")
```

Out[23]:



π course_id, title, ID→instructor_id, name→instructor_name (((course ⋈ section) ⋈ teaches) ⋈ instructor)

Execution time: 1 ms

course.course_id	course.title	instructor_id	instructor_name
'BIO-101'	'Intro. to Biology'	76766	'Crick'
'BIO-301'	'Genetics'	76766	'Crick'
'CS-101'	'Intro. to Computer Science'	10101	'Srinivasan'
'CS-101'	'Intro. to Computer Science'	45565	'Katz'
'CS-190'	'Game Design'	83821	'Brandt'
'CS-315'	'Robotics'	10101	'Srinivasan'
'CS-319'	'Image Processing'	45565	'Katz'
'CS-319'	'Image Processing'	83821	'Brandt'
'CS-347'	'Database System Concepts'	10101	'Srinivasan'
'EE-181'	'Intro. to Digital Systems'	98345	'Kim'

R1

Question

- *Consider the relation produced by:*

  π *course_id, sec_id, building, room_number, time_slot_id (section)*

- *This contains sections, their time assignments and room assignments independent of the year and semester.*

- *Two sections in this derived table conflict if they have the same `building, room_number, time_slot_id`.*
- *My answer to this question is ... ...*

one.course_id	one.sec_id	one.building	one.room_number	one.time_slot_id	two.course_id	two.sec_id
CS-347	1	Taylor	3128	A	CS-190	2
EE-181	1	Taylor	3128	C	CS-319	2

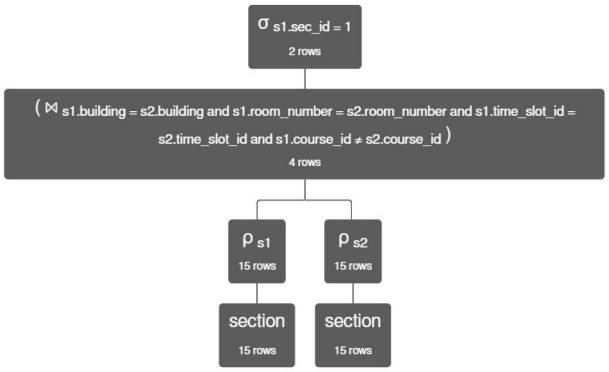
- *Your answer cannot include courses and sections that conflict with themselves, or have two rows that show the same conflict.*

$$\sigma\ s1.sec\_id = 1\ (\ \rho\ s1\ section \bowtie s1.building = s2.building\ and\ s1.room\_number = s2.room\_number\ and\ s1.time\_slot\_id = s2.time\_slot\_id\ and\ s1.course\_id \neq s2.course\_id\ \rho\ s2\ section\ )$$

In [24]:
```
#above <img src="./R1.png"> can run, but cannot show after export html, thus, I use im
from IPython.display import Image

Image("R1.jpg")
```

Out[24]:



σ s1.sec_id = 1 (ρ s1 section ⋈ s1.building = s2.building and s1.room_number = s2.room_number and s1.time_slot_id = s2.time_slot_id and s1.course_id ≠ s2.course_id ρ s2 section)

Execution time: 1 ms

s1.course_id	s1.sec_id	s1.semester	s1.year	s1.building	s1.room_number	s1.time_slot_id	s2.course_id	s2.sec_id	s2.semester	s2.year
'CS-347'	1	'Fall'	2009	'Taylor'	3128	'A'	'CS-190'	2	'Spring'	2009
'EE-181'	1	'Spring'	2009	'Taylor'	3128	'C'	'CS-319'	2	'Spring'	2010

R2

Question

- *You may use the following operators for this question: `π, σ, ρ, ←`.*

- *Use the `instructor, student, advisor` tables for this question.*
- *There are some students that do not have advisors. That are some instructors that are not advisors.*
- *An `instructor` can be an advisor for a `student` if they are in the same department (`dept_name`).*
- *Produce a relation of the form*

`(instructor_id, instructor_name, instructor_dept_name, student_id, student_name, student_dept_name)`

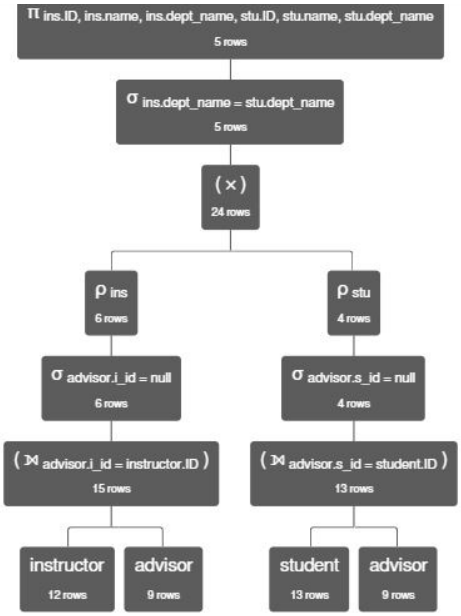
- *That matches instructors that do not advise students and students that do not have advisors and are in the same department.*

$$\pi\ ins.ID,\ ins.name,\ ins.dept\_name,\ stu.ID,\ stu.name,\ stu.dept\_name\ \sigma\ ins.dept\_name = stu.dept\_name\ (\ \rho\ ins\ (\ instructor \triangleright advisor.i\_id=instructor.ID\ advisor\ )\ \times\ \rho\ stu\ (student \triangleright advisor.s\_id=student.ID\ advisor\ )\ )$$

In [25]:
```
#above <img src="./R2.png"> can run, but cannot show after export html working, thus,
from IPython.display import Image

Image("R2.jpg")
```

Out[25]:



$$\pi_{\text{ins.ID, ins.name, ins.dept\_name, stu.ID, stu.name, stu.dept\_name}} \sigma_{\text{ins.dept\_name = stu.dept\_name}} (\rho_{\text{ins}} (\sigma_{\text{advisor.i\_id = null}} (\text{instructor} \bowtie_{\text{advisor.i\_id = instructor.ID}} \text{advisor})) \times \rho_{\text{stu}} (\sigma_{\text{advisor.s\_id = null}} (\text{student} \bowtie_{\text{advisor.s\_id = student.ID}} \text{advisor})))$$

Execution time: 2 ms

ins.ID	ins.name	ins.dept_name	stu.ID	stu.name	stu.dept_name
15151	'Mozart'	'Music'	55739	'Sanchez'	'Music'
32343	'El Said'	'History'	19991	'Brandt'	'History'
33456	'Gold'	'Physics'	70557	'Snow'	'Physics'
58583	'Califieri'	'History'	19991	'Brandt'	'History'
83821	'Brandt'	'Comp. Sci.'	54321	'Williams'	'Comp. Sci.'

SQL

S1

Question

- *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 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.***

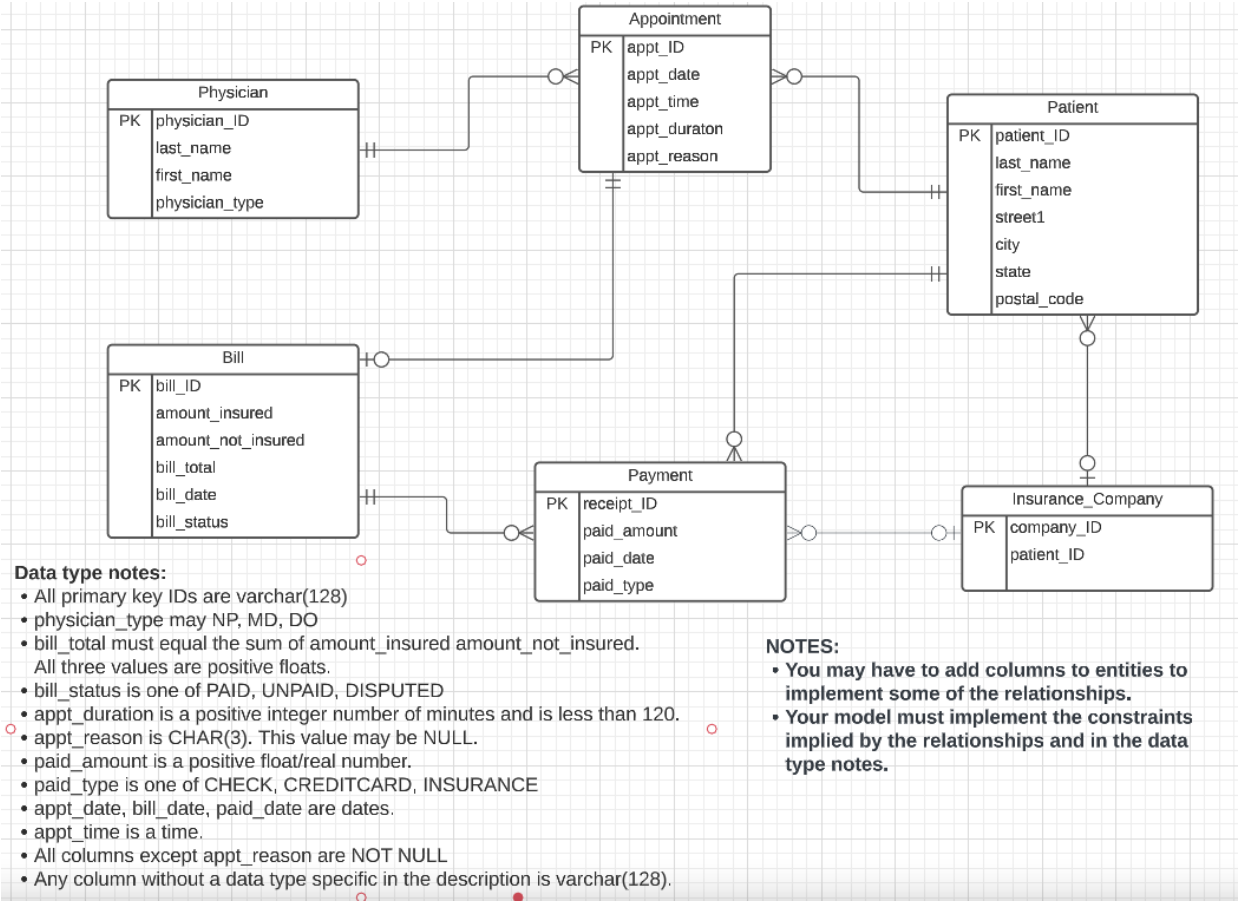
- *You should document any reasonable assumptions you make.*

ER Diagram

In [26]:
```
#above <img src="./jupyter-notebook.png"> can run, but cannot show after export html w
from IPython.display import Image

Image("s23-midterm-er-to-sql-v2.png")
```

Out[26]:



Answer

Design Decisions, Notes, etc:

1. *Add primiary key, foregin key by ALTER table. Impletement bill_total = amount_insured +amoutnt_not insured by* ***ADD CONSTRAINT check_total_amount****.*

2. *Because there are* ***amount_insured and amount_not_insured*** *in Bill table, it means bill can pay by two different paid_type. However,* ***paid_type*** *in Payment table can only have one value at a time. Thus, I will add one table* ***Pay_Method****, with extracted columns from Payment as weak entity of Payment table.*

DDL

- *Execute your DDL in the cell below. You may use DataGrip or other tools to help build the schema.*
- *You can copy and paste the* `SQL CREATE TABLE` *below, but you MUST execute the statements.*

In [27]:
```
%sql drop schema if exists s23_midterm_medical


%sql create schema s23_midterm_medical
```

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

Out[27]:
```
[]
```

In [28]:
```
%%sql
use s23_midterm_medical;

drop table if exists Appointments;
create table if not exists Appointments
(
    appt_ID varchar(128)   not null
        primary key,
    appt_date date not null,
    appt_time time not null,
    appt_duration int CHECK(appt_duration > 0 AND appt_duration < 120) not null,
    appt_reason char(3),
    patient_ID varchar(128)   not null,
    physician_ID varchar(128)   not null
);
```

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

Out[28]:
```
[]
```

In [29]:
```
%%sql
use s23_midterm_medical;

drop table if exists Patient;
create table if not exists Patient
(
    patient_ID varchar(128)   not null
        primary key,
    first_name varchar(128)   not null,
    last_name varchar(128)   not null,
    street1 varchar(128)   not null,
    city varchar(128)   not null,
    state varchar(128)   not null,
    postal_code varchar(128)   not null,
    company_ID varchar(128)   not null
);
```

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

Out[29]:
```
[]
```

In [30]:
```
%%sql
drop table if exists Insurance_Company;
create table if not exists Insurance_Company
(
    company_ID varchar(128)   not null
        primary key,
    patient_ID varchar(128)   not null
);

drop table if exists Payment;
create table if not exists Payment
(
    receipt_ID varchar(128)   not null
        primary key,
    paid_amount FLOAT CHECK(paid_amount > 0) not null,
    patient_ID varchar(128)   not null,
    company_ID varchar(128)   not null,
    bill_ID varchar(128)   not null
);

#This table is weak entity of Payment table. Reason is desribed in design note in begi
drop table if exists Payment_Method;
create table if not exists Payment_Method
(
    paid_date date not null,
    paid_type enum('CHECK','CREDITCARD','INSURANCE') not null
);

drop table if exists Bill;
create table if not exists Bill
(
    bill_ID varchar(128)   not null
        primary key,
    amount_insured FLOAT CHECK(amount_insured > 0) not null,
    amount_not_insured FLOAT CHECK(amount_not_insured > 0) not null,
    bill_total FLOAT CHECK(bill_total > 0) not null,
    bill_date date not null,
    bill_status enum ('PAID','UNPAID','DISPUTED') not null,
    appt_ID varchar(128)   not null
);

drop table if exists Physician;
create table if not exists Physician
(
    physician_ID varchar(128)   not null
        primary key,
    first_name varchar(128)   not null,
    last_name varchar(128)   not null,
    physician_type enum ('NP', 'MD', 'DO')  not null
)
```

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

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

In [31]:
```
%%sql

ALTER TABLE Bill ADD FOREIGN KEY(appt_ID) references Appointments(appt_ID);
ALTER TABLE Payment ADD FOREIGN KEY(patient_ID) references Patient (patient_ID);
ALTER TABLE Payment ADD FOREIGN KEY(company_ID) references Insurance_Company (company_
ALTER TABLE Payment ADD FOREIGN KEY(bill_ID) references Bill (bill_ID);
ALTER TABLE Insurance_Company ADD FOREIGN KEY(patient_ID) references Patient (patient_
ALTER TABLE Patient ADD FOREIGN KEY(company_ID) references Insurance_Company (company_
ALTER TABLE Appointments ADD FOREIGN KEY(physician_ID) references Physician (physician
ALTER TABLE Appointments ADD FOREIGN KEY(patient_ID) references Patient (patient_ID)
```

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

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

In [32]:
```
#add constraint to make sure total=insured+not insured
%sql ALTER TABLE Bill ADD CONSTRAINT check_total_amount CHECK (bill_total =amount_insu
```

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

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

S2

Question

- *Use the classic models database that you loaded.*
- *Write a query that returns the following results:*

`(customerNumber, customerName, no_of_orders, total_revenue)`

- *where:*
  - `customerNumber` *and* `customerName` *are from* `customers`.
  - `no_of_orders` *is the number of orders the customer has placed.*

- `total_revenue` is the sum of `quantityOrdered*priceEach` for all `orderDetails` in `orders` associated with a customer.

- If a customer has not placed any orders, `no_of_orders` and `total_revenue` must be `0`.

Answer

In [33]:
```
%%sql

use classicmodels;

SELECT
    c.customerNumber,
    c.customerName,
    COALESCE(o.no_of_orders, 0) AS no_of_orders,
    COALESCE(o.total_revenue, 0) AS total_revenue
FROM
    classicmodels.customers AS c
LEFT JOIN
    (
        SELECT
            o.customerNumber,
            COUNT(DISTINCT o.orderNumber) AS no_of_orders,
            SUM(od.quantityOrdered * od.priceEach) AS total_revenue
        FROM
            classicmodels.orders AS o
        LEFT JOIN
            classicmodels.orderdetails AS od ON o.orderNumber = od.orderNumber
        GROUP BY
            o.customerNumber
    ) o ON c.customerNumber = o.customerNumber
ORDER BY
    c.customerNumber;
```

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

Out[33]:

customerNumber	customerName	no_of_orders	total_revenue
103	Atelier graphique	3	22314.36
112	Signal Gift Stores	3	80180.98
114	Australian Collectors, Co.	5	180585.07
119	La Rochelle Gifts	4	158573.12
121	Baane Mini Imports	4	104224.79
124	Mini Gifts Distributors Ltd.	17	591827.34
125	Havel & Zbyszek Co	0	0.00
128	Blauer See Auto, Co.	4	75937.76
129	Mini Wheels Co.	3	66710.56
131	Land of Toys Inc.	4	149085.15
141	Euro+ Shopping Channel	26	820689.54
144	Volvo Model Replicas, Co	4	66694.82
145	Danish Wholesale Imports	5	129085.12
146	Saveley & Henriot, Co.	3	130305.35
148	Dragon Souveniers, Ltd.	5	156251.03
151	Muscle Machine Inc	4	177913.95
157	Diecast Classics Inc.	4	104358.69
161	Technics Stores Inc.	4	104545.22
166	Handji Gifts& Co	4	107746.75
167	Herkku Gifts	3	97562.47
168	American Souvenirs Inc	0	0.00
169	Porto Imports Co.	0	0.00
171	Daedalus Designs Imports	2	61781.70
172	La Corne D'abondance, Co.	3	86553.52
173	Cambridge Collectables Co.	2	32198.69
175	Gift Depot Inc.	3	95424.63
177	Osaka Souveniers Co.	2	62361.22
181	Vitachrome Inc.	3	72497.64
186	Toys of Finland, Co.	3	95546.46
187	AV Stores, Co.	3	148410.09
189	Clover Collections, Co.	2	49898.27
198	Auto-Moto Classics Inc.	3	21554.26
201	UK Collectables, Ltd.	3	106610.72
202	Canadian Gift Exchange Network	2	70122.19

204	Online Mini Collectables	2	55577.26
205	Toys4GrownUps.com	3	93803.30
206	Asian Shopping Network, Co	0	0.00
209	Mini Caravy	3	75859.32
211	King Kong Collectables, Co.	2	45480.79
216	Enaco Distributors	3	68520.47
219	Boards & Toys Co.	2	7918.60
223	Natürlich Autos	0	0.00
227	Heintze Collectables	2	89909.80
233	Québec Home Shopping Network	3	68977.67
237	ANG Resellers	0	0.00
239	Collectable Mini Designs Co.	2	80375.24
240	giftsbymail.co.uk	2	71783.75
242	Alpha Cognac	3	60483.36
247	Messner Shopping Network	0	0.00
249	Amica Models & Co.	2	82223.23
250	Lyon Souveniers	3	67659.19
256	Auto Associés & Cie.	2	58876.41
259	Toms Spezialitäten, Ltd	2	89223.14
260	Royal Canadian Collectables, Ltd.	2	66812.00
273	Franken Gifts, Co	0	0.00
276	Anna's Decorations, Ltd	4	137034.22
278	Rovelli Gifts	3	127529.69
282	Souveniers And Things Co.	4	133907.12
286	Marta's Replicas Co.	2	90545.37
293	BG&E Collectables	0	0.00
298	Vida Sport, Ltd	2	108777.92
299	Norway Gifts By Mail, Co.	2	69059.04
303	Schuyler Imports	0	0.00
307	Der Hund Imports	0	0.00
311	Oulu Toy Supplies, Inc.	3	95706.15
314	Petit Auto	3	70851.58
319	Mini Classics	2	78432.16
320	Mini Creations Ltd.	3	101872.52
321	Corporate Gift Ideas Co.	4	132340.78

323	Down Under Souveniers, Inc	5	154622.08
324	Stylish Desk Decors, Co.	3	80556.73
328	Tekni Collectables Inc.	3	81806.55
333	Australian Gift Network, Co	3	55190.16
334	Suominen Souveniers	3	103896.74
335	Cramer Spezialitäten, Ltd	0	0.00
339	Classic Gift Ideas, Inc	2	57939.34
344	CAF Imports	2	46751.14
347	Men 'R' US Retailers, Ltd.	2	41506.19
348	Asian Treasures, Inc.	0	0.00
350	Marseille Mini Autos	3	71547.53
353	Reims Collectables	5	126983.19
356	SAR Distributors, Co	0	0.00
357	GiftsForHim.com	3	94431.76
361	Kommission Auto	0	0.00
362	Gifts4AllAges.com	3	84340.32
363	Online Diecast Creations Co.	3	116449.29
369	Lisboa Souveniers, Inc	0	0.00
376	Precious Collectables	0	0.00
379	Collectables For Less Inc.	3	73533.65
381	Royale Belge	4	29217.18
382	Salzburg Collectables	4	137480.07
385	Cruz & Sons Co.	3	87468.30
386	L'ordine Souveniers	3	125505.57
398	Tokyo Collectables, Ltd	4	105548.73
406	Auto Canal+ Petit	3	86436.97
409	Stuttgart Collectable Exchange	0	0.00
412	Extreme Desk Decorations, Ltd	3	90332.38
415	Bavarian Collectables Imports, Co.	1	31310.09
424	Classic Legends Inc.	3	69214.33
443	Feuer Online Stores, Inc	0	0.00
447	Gift Ideas Corp.	3	49967.78
448	Scandinavian Gift Ideas	3	120943.53
450	The Sharp Gifts Warehouse	4	143536.27
452	Mini Auto Werke	3	51059.99

455	Super Scale Inc.	2	70378.65
456	Microscale Inc.	2	29230.43
458	Corrida Auto Replicas, Ltd	3	112440.09
459	Warburg Exchange	0	0.00
462	FunGiftIdeas.com	3	88627.49
465	Anton Designs, Ltd.	0	0.00
471	Australian Collectables, Ltd	3	55866.02
473	Frau da Collezione	2	25358.32
475	West Coast Collectables Co.	2	43748.72
477	Mit Vergnügen & Co.	0	0.00
480	Kremlin Collectables, Co.	0	0.00
481	Raanan Stores, Inc	0	0.00
484	Iberia Gift Imports, Corp.	2	50987.85
486	Motor Mint Distributors Inc.	3	77726.59
487	Signal Collectibles Ltd.	2	42570.37
489	Double Decker Gift Stores, Ltd	2	29586.15
495	Diecast Collectables	2	65541.74
496	Kelly's Gift Shop	4	137460.79

Best Baseball Players

Question

- *This question uses* `lahmansdb_midterm.batting` , `lahmansdb_midterm.pitching` *and* `lahmansdb_midterm.people` *. You previously loaded this information.*
- *There query computes performance metrics:*
  - *Batting:*
    - *On-base percentage: OBP is (sum(h) + sum(BB))/(sum(ab) + sum(BB)). This value is* `NULL` *if* `sum(ab) = 0` *.*
    - *Slugging percentage: SLG is defined by the function below. The value is* `NULL` *if* `sum(ab) = 0` *.*

```
(
(sum(h) - sum(`1b`) - sum(`2b`) - sum(`3b`) - sum(hr)) +
2*sum(`2b`) + 3*sum(`3b`) + 4*hr
)/sum(ab)
```

  - *Pitching:*
    - *total_wins is* `sum(w)` *.*

    - *total_loses is* `sum(l)`.
    - *win_percentage is* `sum(w)/(sum(w) + sum(l))`. *This value is NULL if* `sum(w) + sum(l) = 0`.

- *Professor Ferguson has two criteria for someone being a great baseball player. A play must meet at least one of the criteria to be a great baseball player.*
  - *Batting:*
    - *Total number of* `ab >= 1500`.
    - *SLG: Career* `SLG >= .575`
  - *Pitching:*
    - `(sum(w) + sum(l)) >= 200`.
    - `win_percentage >= 0.70)` *or* `sum(w) >= 300.`
- *In your result table there is some additional guidance.*
  - `great_because` *is either* `Pitcher` *or* `Batter` *based on whether the player matched the batting or pitching criteria.*
  - *The values from* `batting` *are* `None` *if the player did not qualify based on batting.*
  - *The values from* `pitching` *are* `None` *if the player did not qualify on pitching.*

Note: *For this query to run efficiently, you will need to create indexes on the tables.*

Answer

- *Execute your create index statements below.*

In [34]:
```
%%sql
# DROP INDEX index_batting;
CREATE INDEX index_batting
ON lahmansdb_midterm.batting (AB, H, 2B,3B, HR,BB);
#1B

# DROP INDEX index_pitching;
CREATE INDEX index_pitching
ON lahmansdb_midterm.pitching (W, L)
```

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

Out[34]:
```
[]
```

- *Execute your SQL statement producing the query result below.*

In [35]:
```
%%sql
with career_basic as (
    select
        playerid,
        (sum(h) - sum(hr) - sum(`2b`) - sum(`3b`)) as career_singles,
        sum(`2b`) as career_doubles,
        sum(`3b`) as career_triples,
```

```
        sum(hr) as  career_hrs,
        sum(ab) as career_abs,
        sum(h) as career_hits,
        sum(bb) as career_walks
    from
        lahmansdb_midterm.batting
    group by playerid
),
    pi_summary as (
        select playerid,
               sum(w) as pi_win,
               sum(l) as pi_lose,
               sum(w)+sum(l) as pi_decision
        from
            lahmansdb_midterm.pitching
        group by playerid
    ),
    career_averages as (
        select
            playerid, career_abs, career_hits,
            career_singles, career_doubles, career_triples, career_hrs,
            career_walks,
            if(career_abs=0, NULL, (career_hits + career_walks)/(career_abs + career_w
            if(career_abs = 0,
                null,
               (career_singles + 2 * career_doubles + 3 * career_triples + 4 * career_
                ) as slg
        from career_basic
    ),
    pi_average as (
        select
            playerid, pi_win, pi_lose,pi_decision,
            if (pi_decision=0, NULL, ((pi_win)/pi_decision)) as win_percentage
            from pi_summary
    ),
    career_end as(
        select playerid,career_abs,career_singles,career_doubles,career_triples,career
        from career_averages
        having slg>0.575 and career_abs>=1500
    ),
    pi_end as (
        select playerid, pi_win, pi_lose, pi_decision, win_percentage
        from pi_average
        having pi_decision>=200 and (win_percentage>0.7 or pi_win>=300)
    )


SELECT playerid,career_abs,career_singles,career_doubles,career_triples,career_hrs,obp
FROM career_end
UNION
SELECT playerid,Null as career_abs,Null as career_singles,Null as career_doubles,Null
FROM pi_end;
```

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

Out[35]:

playerid	career_abs	career_singles	career_doubles	career_triples	career_hrs	obp	slg	pi_wi
ruthba01	8398	1517	506	136	714	0.4718	0.6898	Non
hornsro01	8173	1919	541	169	301	0.4308	0.5765	Non
gehrilo01	8001	1531	534	163	493	0.4447	0.6324	Non
foxxji01	8134	1529	458	125	534	0.4275	0.6093	Non
greenha01	5193	847	379	71	331	0.4103	0.6050	Non
dimagjo01	6821	1333	389	131	361	0.3947	0.5788	Non
willite01	7706	1537	525	71	521	0.4806	0.6338	Non
bondsba01	9847	1495	601	77	762	0.4428	0.6069	Non
mcgwima01	6187	785	252	6	583	0.3922	0.5882	Non
ramirma02	8244	1452	547	20	555	0.4077	0.5854	Non
troutmi01	4656	792	268	49	310	0.4137	0.5831	Non
spaldal01	None	None	None	None	None	None	None	25
galvipu01	None	None	None	None	None	None	None	36
keefeti01	None	None	None	None	None	None	None	34
welchmi01	None	None	None	None	None	None	None	30
radboch01	None	None	None	None	None	None	None	31
clarkjo01	None	None	None	None	None	None	None	32
nichoki01	None	None	None	None	None	None	None	36
youngcy01	None	None	None	None	None	None	None	51
mathech01	None	None	None	None	None	None	None	37
planked01	None	None	None	None	None	None	None	32
johnswa01	None	None	None	None	None	None	None	41
alexape01	None	None	None	None	None	None	None	37
grovele01	None	None	None	None	None	None	None	30
wynnea01	None	None	None	None	None	None	None	30
spahnwa01	None	None	None	None	None	None	None	36
perryga01	None	None	None	None	None	None	None	31
niekrph01	None	None	None	None	None	None	None	31
carltst01	None	None	None	None	None	None	None	32
ryanno01	None	None	None	None	None	None	None	32
suttodo01	None	None	None	None	None	None	None	32
seaveto01	None	None	None	None	None	None	None	31
clemero02	None	None	None	None	None	None	None	35
maddugr01	None	None	None	None	None	None	None	35

glavito02	None	None	None	None	None	None	None	30

Data and Schema Cleanup

Explanation and Setup

- *There are several issues with the schema for* `clasicmodels.` *Two of the issues are:*
  - `customers.country:` *Having programs or people enter country names is prone to errors.*
  - `products.productCode` *is clearly not an atomic value.*
- *The following SQL creates a schema with copies of the data. The SQL also loads a table of ISO country codes.*

In [36]:
```
%sql create schema classicmodels_midterm;
```

```
 * mysql+pymysql://root:***@localhost
(pymysql.err.ProgrammingError) (1007, "Can't create database 'classicmodels_midterm';
database exists")
[SQL: create schema classicmodels_midterm;]
(Background on this error at: https://sqlalche.me/e/14/f405)
```

In [37]:
```
iso_df = pandas.read_csv('./wikipedia-iso-country-codes.csv')
iso_df.to_sql('country_codes', schema='classicmodels_midterm',
              con=sql_engine, index=False, if_exists="replace")
```

Out[37]:
```
246
```

In [38]:
```
%%sql

use classicmodels_midterm;

alter table classicmodels_midterm.country_codes
    change `English short name lower case` short_name text null;

alter table classicmodels_midterm.country_codes
    change `Alpha-2 code` alpha_2_code text null;

alter table classicmodels_midterm.country_codes
    change `Alpha-3 code` alpha_3_code text null;

alter table classicmodels_midterm.country_codes
    change `Numeric code` numberic_code bigint null;

alter table classicmodels_midterm.country_codes
    change `ISO 3166-2` iso_text text null;
```

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

Out[38]:

In [39]:
```
%%sql

use classicmodels_midterm;

drop table if exists customers;
create table customers as select * from classicmodels.customers;

drop table if exists products;
create table products as select * from classicmodels.products;
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
0 rows affected.
122 rows affected.
0 rows affected.
110 rows affected.
[]
```

Out[39]:

DE-1

Question

- *There are four* `country` *values in* `customers` *that are not in* `short_names` *of* `country_codes.`
- *The four missing values are:*

country
USA
Norway
UK
Russia

- *Write an SQL query that returns the information about by querying* `customers` *and* `country_codes`

Answer

In [40]:
```
%%sql
SELECT distinct(customers.country)
FROM classicmodels_midterm.customers
```

```
LEFT JOIN classicmodels_midterm.country_codes ON customers.country = country_codes.sho
WHERE country_codes.short_name IS NULL;
```

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

Out[40]:

country
USA
Norway
UK
Russia

DE-2

Question

- *Norway is on the list because there are spaces in the entry. The following query shows this fact.*

In [41]:
```
%%sql

select customerNumber, customerName, country
from customers where length(country) != length(trim(country));
```

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

Out[41]:

customerNumber	customerName	country
167	Herkku Gifts	Norway
299	Norway Gifts By Mail, Co.	Norway

- *The mapping of the other country names is:*

customers.country	country_codes.short_name
USA	United States
UK	United Kingdom
Russia	Russian Federation

- *Write a **single** `update` statement that corrects the values for `customers.country`.*

Answer

In [42]:
```
%%sql
UPDATE classicmodels_midterm.customers
SET country =
    CASE
        WHEN country = 'USA' THEN 'United States'
        WHEN country = 'UK' THEN 'United Kingdom'
```

```
        WHEN country = 'Russia' THEN 'Russian Federation'
        WHEN country = ' Norway' THEN 'Norway'
        ELSE country
    END;
```

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

Out[42]:
```
[]
```

DE-3

Question

- *The final tasks are:*
  - *Add a column* `iso_code` *to* `customers` *that is the* `alpha_2_code` *from* `country_codes`.
  - *Create a foreign key relationship* `customers.iso_code -> country_codes.alpha_2_code`.
  - *Drop* `country` *from* `customers`.
  - *Create a view* `customers_country` *of the form* `(customerNumber, customerName, country, iso_code)`.

Answer

In [43]:
```
%%sql

/*
Update classicmodels_midterm.country_codes set alpha_2_code='NA' where short_name='Nam

ALTER TABLE classicmodels_midterm.customers ADD COLUMN iso_code VARCHAR(2);

UPDATE classicmodels_midterm.customers
SET customers.iso_code = (
    SELECT country_codes.alpha_2_code
    FROM country_codes
    WHERE country_codes.short_name = customers.country
);

ALTER TABLE customers ADD FOREIGN KEY(iso_code) references country_codes(alpha_2_code)

ALTER TABLE customers DROP COLUMN country;

create or replace view customers_country as
    select
        customerNumber, customerName, country, iso_code
        from
        customers;
*/
```

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

Out[43]:
```
[]
```

In [44]:

```
%%sql

/* Write a SELECT that displays 25 customers sorted by customerName */

SELECT *
FROM customers
ORDER BY customerName
LIMIT 25;
```

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

Out[44]:

customerNumber	customerName	contactLastName	contactFirstName	phone	addressLine1	addl
242	Alpha Cognac	Roulet	Annette	61.77.6555	1 rue Alsace-Lorraine	
168	American Souvenirs Inc	Franco	Keith	2035557845	149 Spinnaker Dr.	
249	Amica Models & Co.	Accorti	Paolo	011-4988555	Via Monte Bianco 34	
237	ANG Resellers	Camino	Alejandra	(91) 745 6555	Gran Vía, 1	
276	Anna's Decorations, Ltd	O'Hara	Anna	02 9936 8555	201 Miller Street	
465	Anton Designs, Ltd.	Anton	Carmen	+34 913 728555	c/ Gobelas, 19-1 Urb. La Florida	
206	Asian Shopping Network, Co	Walker	Brydey	+612 9411 1555	Suntec Tower Three	8
348	Asian Treasures, Inc.	McKenna	Patricia	2967 555	8 Johnstown Road	
103	Atelier graphique	Schmitt	Carine	40.32.2555	54, rue Royale	
471	Australian Collectables, Ltd	Clenahan	Sean	61-9-3844-6555	7 Allen Street	
114	Australian Collectors, Co.	Ferguson	Peter	03 9520 4555	636 St Kilda Road	
333	Australian Gift Network, Co	Calaghan	Ben	61-7-3844-6555	31 Duncan St. West End	
256	Auto Associés & Cie.	Tonini	Daniel	30.59.8555	67, avenue de l'Europe	
406	Auto Canal+ Petit	Perrier	Dominique	(1) 47.55.6555	25, rue Lauriston	
198	Auto-Moto Classics Inc.	Taylor	Leslie	6175558428	16780 Pompton St.	
187	AV Stores, Co.	Ashworth	Rachel	(171) 555-1555	Fauntleroy Circus	
121	Baane Mini Imports	Bergulfsen	Jonas	07-98 9555	Erling Skakkes gate 78	
415	Bavarian Collectables Imports, Co.	Donnermeyer	Michael	+49 89 61 08 9555	Hansastr. 15	
293	BG&E Collectables	Harrison	Ed	+41 26 425 50 01	Rte des Arsenaux 41	
128	Blauer See Auto, Co.	Keitel	Roland	+49 69 66 90 2555	Lyonerstr. 34	

	219	Boards & Toys Co.	Young	Mary	3105552373	4097 Douglas Av.	
	344	CAF Imports	Fernandez	Jesus	+34 913 728 555	Merchants House	M
	173	Cambridge Collectables Co.	Tseng	Jerry	6175555555	4658 Baden Av.	
	202	Canadian Gift Exchange Network	Tamuri	Yoshi	(604) 555-3392	1900 Oak St.	
	339	Classic Gift	Cervantes	Francisca	2155554695	782 First	

DE-4

- *Just kidding.*
- *My first intent was to have you fix* `products` .
- *Then, I thought I would make this an extra credit question.*
- *Finally, I decided that all students get 5 points added to there score for this exam. Since I never "curve up," you all get a bonus on final grade for putting up with the class.*

In [45]:
```
import pandas as pd
df = pd.read_csv("C:/Users/11139/Desktop/W4111-02/Midterm/pitching.csv")
a = []
for i in df.columns:
    a.append(i)
print(a)
```

```
['playerID', 'yearID', 'stint', 'teamID', 'lgID', 'W', 'L', 'G', 'GS', 'CG', 'SHO',
'SV', 'IPouts', 'H', 'ER', 'HR', 'BB', 'SO', 'BAOpp', 'ERA', 'IBB', 'WP', 'HBP', 'B
K', 'BFP', 'GF', 'R', 'SH', 'SF', 'GIDP']
```