COMS W4111: Introduction to Databases
Spring 2023, Sections 002

Homework 2, Part 1
Continuing Core Concepts, ER Modeling, Relational Algebra, SQL

* </i>*

Introduction and Overview

HW Objectives

- *HW 2, part 1 is for **both tracks.***
- *We have covered additional concepts since HW 1. HW 2, part 1 tests and reinforces learning the new concepts.*

Submission Instructions

Complete all the tests in this notebook and submit only this notebook as a PDF to GradeScope. To convert the jupyter notebook into a pdf you can use either of the following methods:

- *File --> Print Preview --> Print --> Save to PDF*
- *File --> Download As HTML --> Print --> Save to PDF*

Due date: March 1, 11:59 PM EDT on GradeScope

It is recommended that you put the screenshots into the same folder as this notebook so you do not have to alter the path to include your images.

Please read all the instructions thoroughly!

Guidelines

*You may not work with or collaborate with anyone in any way to complete the homework. You may speak with the professor and TAs. You may ask **private** questions on Ed if you need clarification.*

You may use lecture slides, the textbook slides, the textbook or public information on the web to help you answer your questions. You may not "cut and past" information. Your answer must be in your own words and demonstrate the you understand the concept. If you use information for sources other than lectures, lecture slides, textbook slides or the textbook, you MUST provide a URL to the source you used.

Add Student Information

1. *Replace my name with your full name.*
2. *Replace my UNI with your UNI.*
3. *Replace "Cool Track" with either "Programming" or "Non-programming."*

In [58]:
```
# Print your name, uni, and track below

name = "Haoqing Wang"
uni = "hw2888"
track = "Programming"

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

```
Haoqing Wang
hw2888
Programming
```

Testing Environment

Run the following cells to ensure that your environment is set up.

You may need to change passwords.

General Packages

In [59]:
```
import json
```

In [60]:
```
import csv
```

In [61]:
```
import pandas
```

In [62]:
```
import os
```

pymysql

In [63]:
```
import pymysql
```

In [64]:
```
#
# Run this cell but change your user ID and password.
#
pymysql_conn = pymysql.connect(
    user="root",
    password="dbuserdbsuer",
    host="localhost",
    port=3306,
    autocommit=True,
    cursorclass=pymysql.cursors.DictCursor
)
```

In [65]:
```
# You must have installed the db_book DB for this to work.
#
cursor = pymysql_conn.cursor()
sql = "select * from db_book.student"
res = cursor.execute(sql)
result = cursor.fetchall()
```

In [66]:
```
#
# You must have installed the db_book DB for this to work.
#
df1 = pandas.DataFrame(result)
df1
```

Out[66]:

	ID	name	dept_name	tot_cred
0	00128	Zhang	Comp. Sci.	102
1	12345	Shankar	Comp. Sci.	32
2	19991	Brandt	History	80
3	23121	Chavez	Finance	110
4	44553	Peltier	Physics	56
5	45678	Levy	Physics	46
6	54321	Williams	Comp. Sci.	54
7	55739	Sanchez	Music	38
8	70557	Snow	Physics	0
9	76543	Brown	Comp. Sci.	58
10	76653	Aoi	Elec. Eng.	60
11	98765	Bourikas	Elec. Eng.	98
12	98988	Tanaka	Biology	120

ipython-SQL

In [67]:
```
%load_ext sql
```

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

In [68]:
```
#
# Remember to change your user ID and password.
#
%sql mysql+pymysql://root:dbuserdbsuer@localhost
```

In [69]:
```
%sql select * from db_book.student where ID=12345
```

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

Out[69]:

ID	name	dept_name	tot_cred
12345	Shankar	Comp. Sci.	32

SQLAlchemy

In [70]:
```
from sqlalchemy import create_engine
```

In [71]:
```
#
# Remember to change your user ID and password.
#
sql_url = "mysql+pymysql://root:dbuserdbsuer@localhost"
```

In [72]:
```
engine = create_engine(sql_url)
```

In [73]:
```
sql = "select * from db_book.student"
df = pandas.read_sql(sql, con=engine)
```

In [74]:
```
df
```

Out[74]:

	ID	name	dept_name	tot_cred
0	00128	Zhang	Comp. Sci.	102.0
1	12345	Shankar	Comp. Sci.	32.0
2	19991	Brandt	History	80.0
3	23121	Chavez	Finance	110.0
4	44553	Peltier	Physics	56.0
5	45678	Levy	Physics	46.0
6	54321	Williams	Comp. Sci.	54.0
7	55739	Sanchez	Music	38.0
8	70557	Snow	Physics	0.0
9	76543	Brown	Comp. Sci.	58.0
10	76653	Aoi	Elec. Eng.	60.0
11	98765	Bourikas	Elec. Eng.	98.0
12	98988	Tanaka	Biology	120.0

Relational Algebra

Instructions

- *Use the schema and data associated with the recommended textbook. Clicking the link should take you directly to the RelaX page configured with the DB.*
- *Your submission format for each question is:*
  - *A Markdown cell with a copy of your relational algebra statement.*
  - *A screen shot of the execution. If the expression returns several result pages, you only need to show the first page.*
  - *There is an example below.*

Question R1: Courses and Prereq

Question

- *Please run the following relational algebra statement. The statement produces a table of course that do not have preqs and courses that are not prereq.*

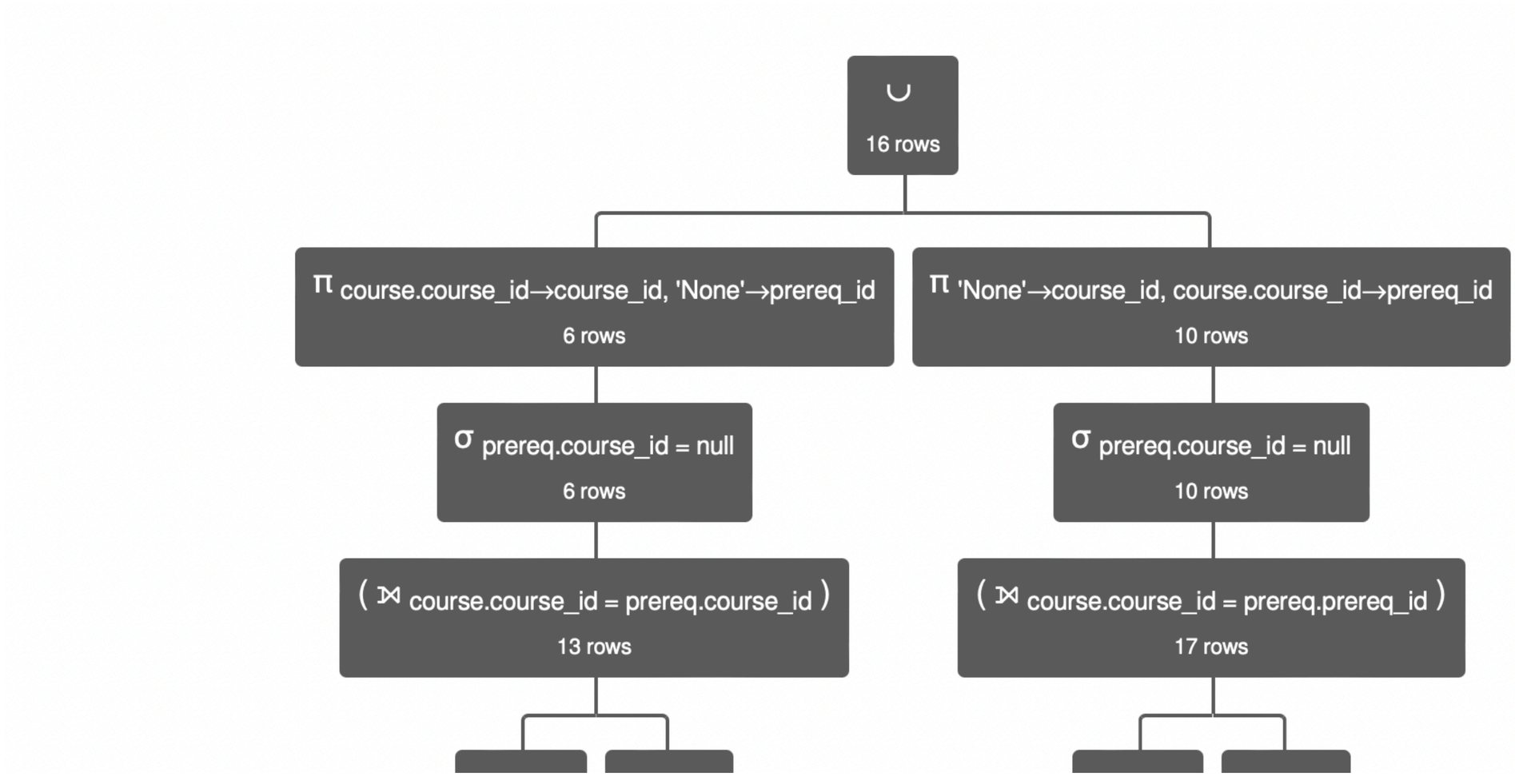
```
(π course_id←course.course_id, prereq_id←'None' (course ▷ prereq))
∪
(π
    course_id←'None', prereq_id←course.course_id
    (course ▷ course.course_id=prereq.prereq_id prereq)
)
```

- *Please write an equivalent query that does not use* `anti-join.`

(π course_id←course.course_id, prereq_id←'None' (σ prereq.course_id = null (course ⟗ course.course_id = prereq.course_id prereq)))

∪

(π course_id←'None', prereq_id←course.course_id (σ prereq.course_id = null (course ⟗ course.course_id = prereq.prereq_id prereq)))



course
13 rows

prereq
7 rows

course
13 rows

prereq
7 rows

$$(\pi_{\text{course.course\_id} \rightarrow \text{course\_id},\ \text{'None'} \rightarrow \text{prereq\_id}} (\sigma_{\text{prereq.course\_id} = \text{null}} (\text{course} \bowtie_{\text{course.course\_id} = \text{prereq.course\_id}} \text{prereq}))) \cup (\pi_{\text{'None'} \rightarrow \text{course\_id},\ \text{course.course\_id} \rightarrow \text{prereq\_id}} (\sigma_{\text{prereq.course\_id} = \text{null}} (\text{course} \bowtie_{\text{course.course\_id} = \text{prereq.prereq\_id}} \text{prereq})))$$

Execution time: 0 ms

course_id	prereq_id
'BIO-101'	'None'
'CS-101'	'None'
'FIN-201'	'None'
'HIS-351'	'None'
'MU-199'	'None'
'PHY-101'	'None'
'None'	'BIO-301'
'None'	'BIO-399'
'None'	'CS-190'
'None'	'CS-315'

Question R2: Instructors and Credits

Question

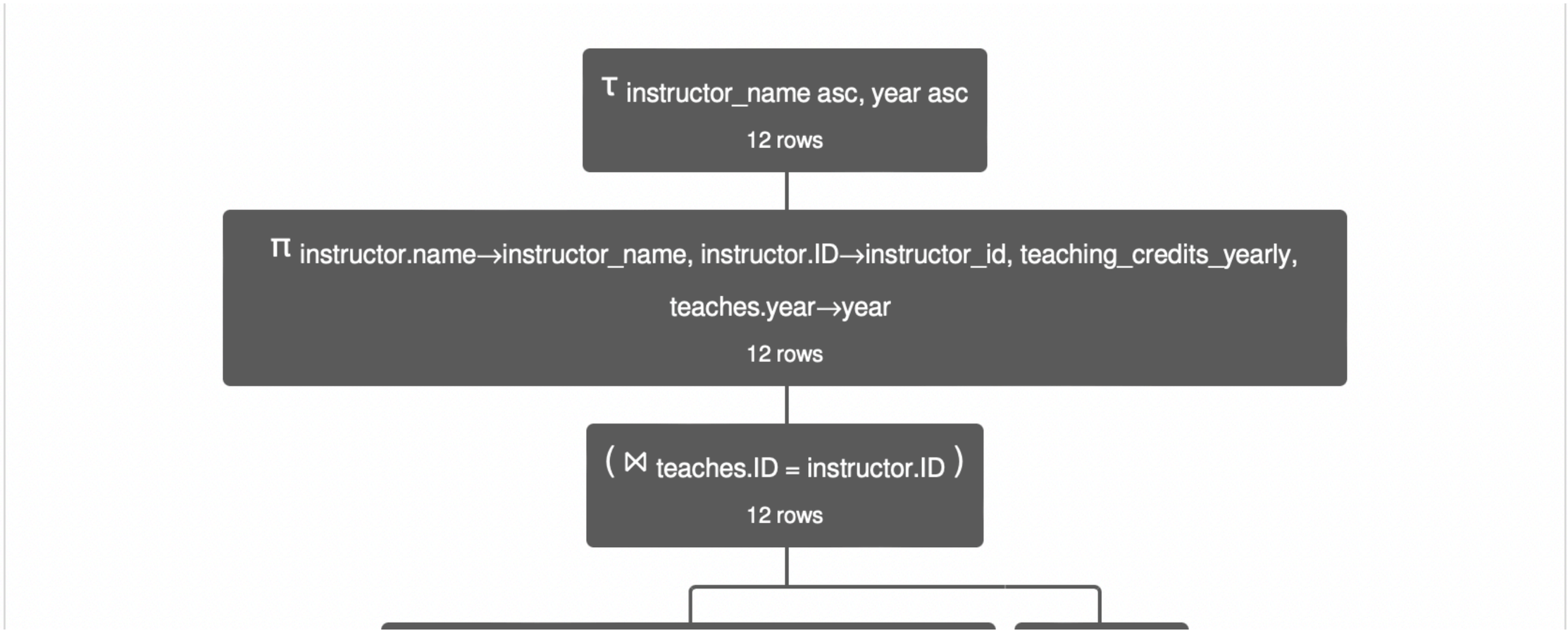
Please write a relational algebra statement that produces a relation of the form:

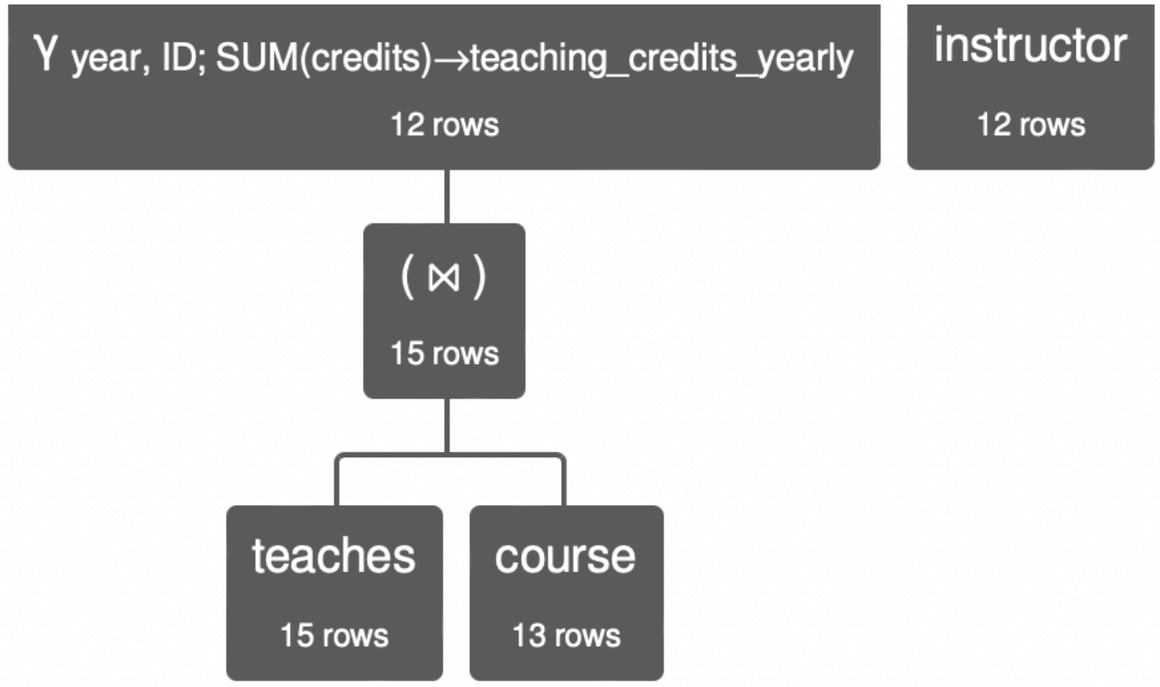
- *Columns*
  - `instructor_name` *is* `instructor.name`
  - `instructor_id` *is* `instructor.ID`
  - `teaching_credits_yearly` *is the total course credits that the instructor taught in a year. The information needed is in* `teaches` *and* `course`.
  - `year` *is from* `teaches`.
- *Sort the result by* `instructor_name` *and then ``year```.*

τ instructor_name, year

(π instructor_name ← instructor.name, instructor_id ← instructor.ID, teaching_credits_yearly, year ← teaches.year

((γ year, ID; sum(credits) → teaching_credits_yearly (teaches ⋈ course)) ⋈ teaches.ID = instructor.ID instructor))





$$\tau \text{ instructor\_name asc, year asc } (\pi \text{ instructor.name} \rightarrow \text{instructor\_name, instructor.ID} \rightarrow \text{instructor\_id, teaching\_credits\_yearly, teaches.year} \rightarrow \text{year } ((\gamma \text{ year, ID; SUM(credits)} \rightarrow \text{teaching\_credits\_yearly } (\text{teaches} \bowtie \text{course})) \bowtie_{\text{teaches.ID = instructor.ID}} \text{instructor}))$$

Execution time: 2 ms

instructor_name	instructor_id	teaching_credits_yearly	year
'Brandt'	83821	8	2009

'Brandt'	83821	3	2010
'Crick'	76766	4	2009
'Crick'	76766	4	2010
'Einstein'	22222	4	2009
'El Said'	32343	3	2010
'Katz'	45565	7	2010
'Kim'	98345	3	2009
'Mozart'	15151	3	2010
'Srinivasan'	10101	7	2009

ER Modeling and Implementation

Data

- *We will do bottom up modeling and implementation starting with data.*
- *There are two entity sets for reengineering:*
  1. `customers`
  2. `reservations` *is a single set containing three types of reservation:*
     A. `Flight`
     B. `Rental car`
     C. `Hotel`
- *The input data is:*

In [75]:
```
customers_df = pandas.read_csv('./customers-3.csv')
```

In [76]:
```
customers_df
```

Out[76]:

	customer_id	first_name	last_name	email
0	4fc2a5e7-859c-494f-9f82-9ef3da9c6265	Lew	Jagiela	ljagiela0@comsenz.com
1	e649dc5e-5570-43f6-837c-89379800179f	Karin	De Ruggiero	kderuggiero1@linkedin.com
2	a20afabf-86af-4070-9c7c-4b4643202013	Georgena	Gapp	ggapp2@ed.gov
3	12d17b18-f009-49fd-a887-eaab9cd60274	Enrika	Thripp	ethripp3@newsvine.com
4	d9a14dd2-0000-4227-bf30-6907efb4caa7	Edin	Seyler	eseyler4@geocities.jp
...	...	...	...	...
995	bbe5ca2a-b5e1-4ff6-a1d6-450bdf41f6ec	Sean	Wagen	swagenr8@abc.net.au
996	a6632fc7-316b-4ad5-9b2c-977f7a732d54	Delmar	Benoy	dbenoyr9@multiply.com
997	3f05996b-5c79-429f-aacb-2367255cc495	Crin	Smooth	csmoothra@blogger.com
998	8e88da67-2e28-4330-bfa2-17c4369c96cf	Riobard	Ricket	rricketrb@va.gov
999	493ca6a1-b2c7-4c1d-817f-3a0d73f8f24c	Neville	Blowen	nblowenrc@etsy.com

1000 rows × 4 columns

In [77]:
```
reservations_df = pandas.read_csv("./reservations-3.csv")
```

In [78]:
```
import numpy as np
```

In [79]:
```
reservations_df = reservations_df.replace({np.nan: None})
```

In [80]:
```
reservations_df
```

Out[80]:

	reservation_id	customer_id	reservation_type	price	reservation_date	city	reservation_start	reservation_end	car_category	room_category	bed_category	flight_departure_time	flight_ar
0	b097458e-898a-40a0-85eb-cedb0ea47466	4fc2a5e7-859c-494f-9f82-9ef3da9c6265	Car	$126.30	2/23/2022	Granja	2/21/2022	8/18/2022	Midsize	None	None	None	
1	67070e81-81f0-4d95-9302-98486a08bf73	e649dc5e-5570-43f6-837c-89379800179f	Hotel	$792.89	1/30/2023	Kavalerovo	8/23/2022	1/19/2023	None	Studio	King	None	
2	f164e71e-4874-4c42-beb4-27c2c6f5af0f	a20afabf-86af-4070-9c7c-4b4643202013	Hotel	$660.09	9/30/2022	Daxi	8/20/2022	8/23/2022	None	Suite	Two Doubles	None	
3	7b088670-9567-4068-8571-946dd085dab5	12d17b18-f009-49fd-a887-eaab9cd60274	Flight	$423.59	1/9/2023	None	None	None	None	None	None	12:59 AM	
4	ca849908-b88e-49d8-b52b-2aeb88b3c949	d9a14dd2-0000-4227-bf30-6907efb4caa7	Hotel	$552.07	11/5/2022	Albuquerque	8/24/2022	4/15/2022	None	Suite	King	None	
...	...	...	...	...	...	...	...	...	...	...	...	...	
995	16b955d3-0181-4ad3-9654-0fea52bf1094	bbe5ca2a-b5e1-4ff6-a1d6-450bdf41f6ec	Hotel	$169.73	2/3/2023	Cibunar	12/17/2022	4/16/2022	None	Studio	King	None	
996	caa55cf1-3032-42b4-94d5-2f8ac70585ce	a6632fc7-316b-4ad5-9b2c-977f7a732d54	Hotel	$517.19	6/12/2022	Quiling	3/12/2022	3/17/2022	None	Studio	King	None	
997	e99b7c4f-1e14-42e1-9598-5010172f3454	3f05996b-5c79-429f-aacb-2367255cc495	Car	$636.63	2/11/2023	Hukeng	8/6/2022	7/19/2022	SUV	None	None	None	
998	c54b4398-3eb7-46a0-9906-d2acc04f0e90	8e88da67-2e28-4330-bfa2-17c4369c96cf	Car	$174.76	11/10/2022	Pravdinsk	10/16/2022	12/7/2022	Compact	None	None	None	
999	4c942bd8-2d80-4d57-9281-0e6387ed7f51	493ca6a1-b2c7-4c1d-817f-3a0d73f8f24c	Car	$662.24	11/10/2022	Kongolo	10/15/2022	4/3/2022	Midsize	None	None	None	

1000 rows × 15 columns

Model

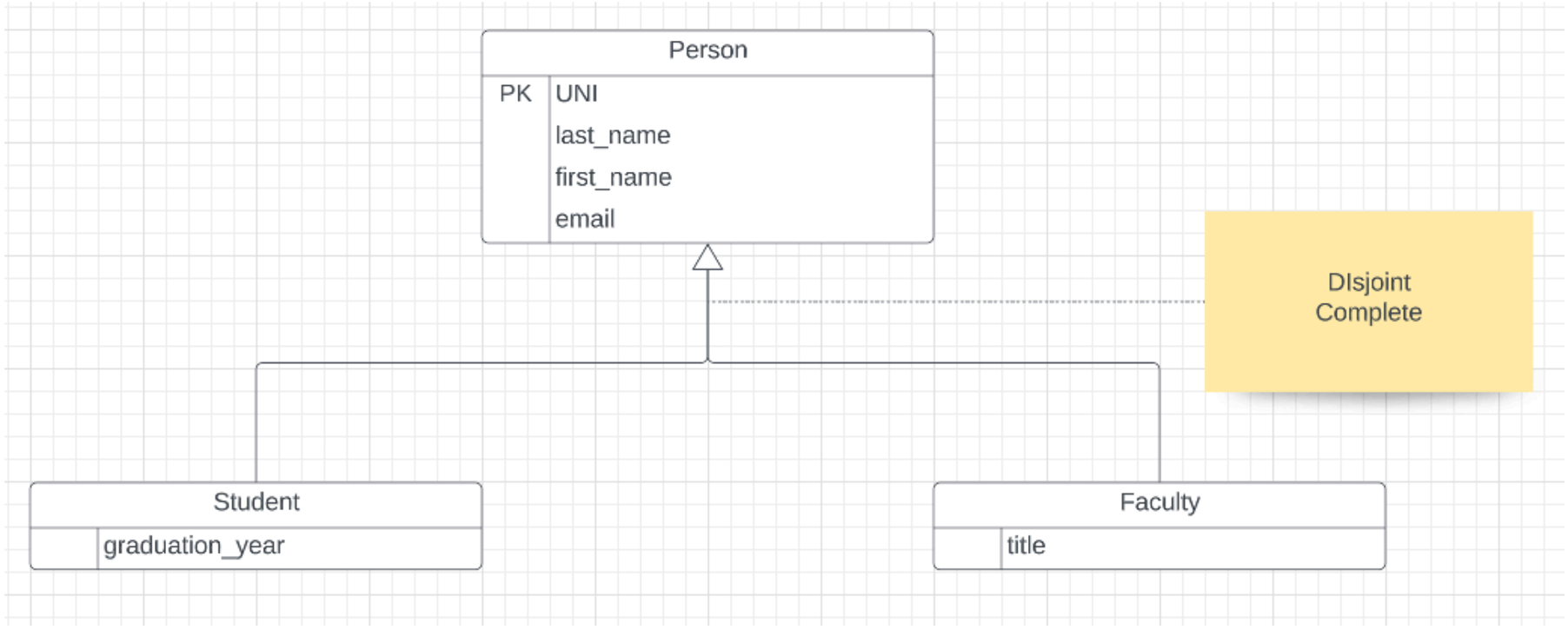
- *You are going to model interitance/specialization.*
  - *There is a base entity type Reservation.*
  - *CarReservation **isA** Reservation.*
  - *FlightReservation **isA** Reservation.*
  - *HotelReservation **isA** Reservation.*
  - *The specialization is complete and disjoin.*

- *The following table shows which properties in the input data apply to which reservation type.*

Property	Meaning	Car	Flight	Hotel
reservation_id	*Unique ID, primary key*	*X*	*X*	*X*
customer_id	*Unique ID, foreign key*	*X*	*X*	*X*
reservation_type	*Enum for type of reservation*	*X*	*X*	*X*
price	*Price to charge*	*X*	*X*	*X*
reservation_date	*Date reservation made/changed*	*X*	*X*	*X*
city	*City for reservation*	*X*		*X*
reservation_start	*Start date*	*X*		*X*
reservation_end	*End date*	*X*		*X*
car_category	*Enum*	*X*		
room_category	*Enum*			*X*
bed_category	*Enum*			*X*
flight_departure_time	*Takeoff time*		*X*	
flight_arrival_time	*Arrival time*		*X*	
departure_airport	*Departure airport code*		*X*	
arrival_airport	*Arrival airport code*		*X*	

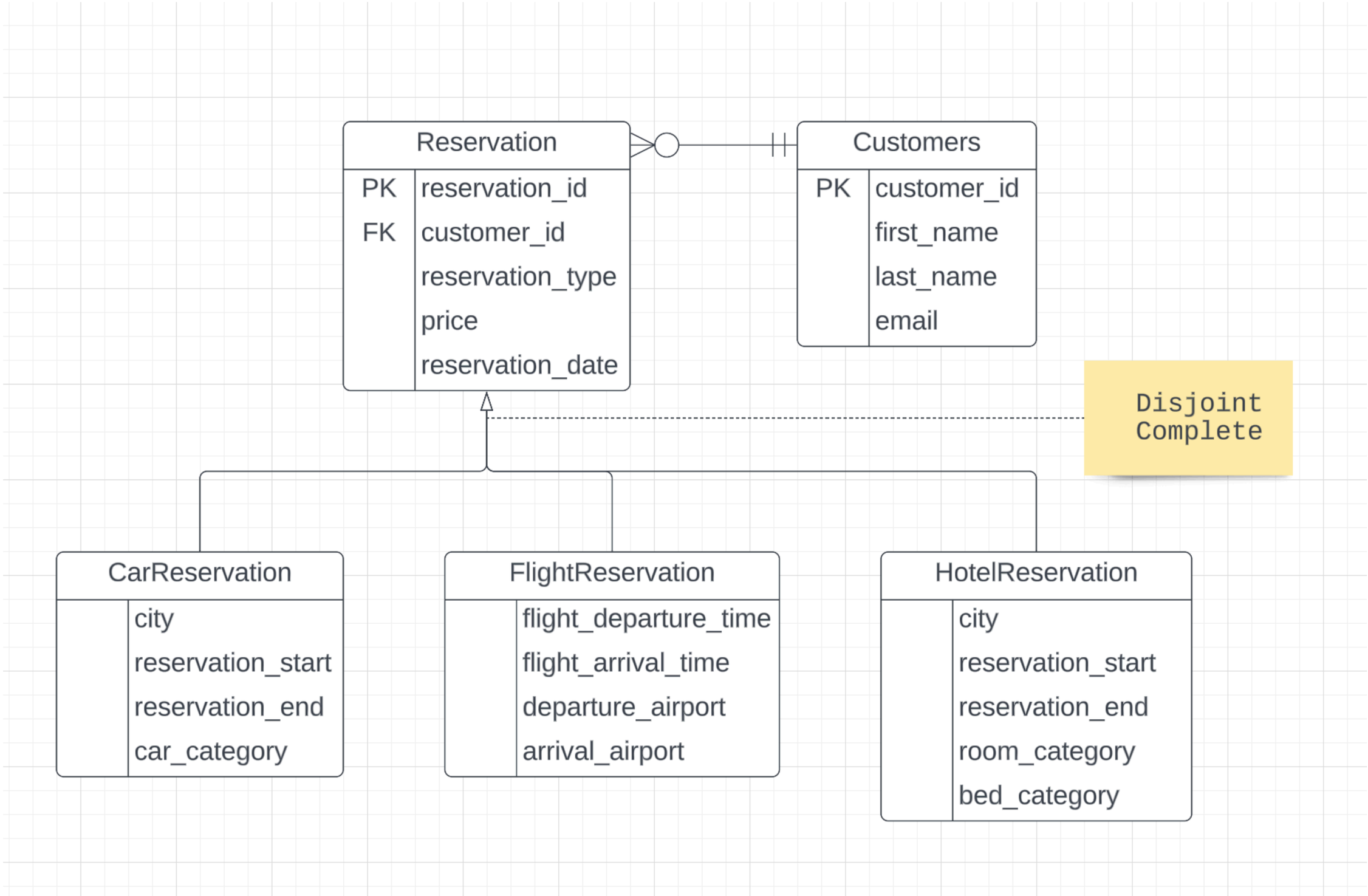
- *The implementation pattern for inheritance will be:*
  - *Reservations is a view.*
  - *There is a table for each of CarReservation, FlightReservation, HotelReservation.*

- *We covered a simple example of modeling inheritance in class. The diagram notiation we used for Person, Student, Faculty is:*



- *Draw diagram for Reservation, CarReservation, FlightReservation, HotelReservation using the pattern above.*
- *Your diagram should include a foreign key relationship from Reservations to Customers.*

Diagram

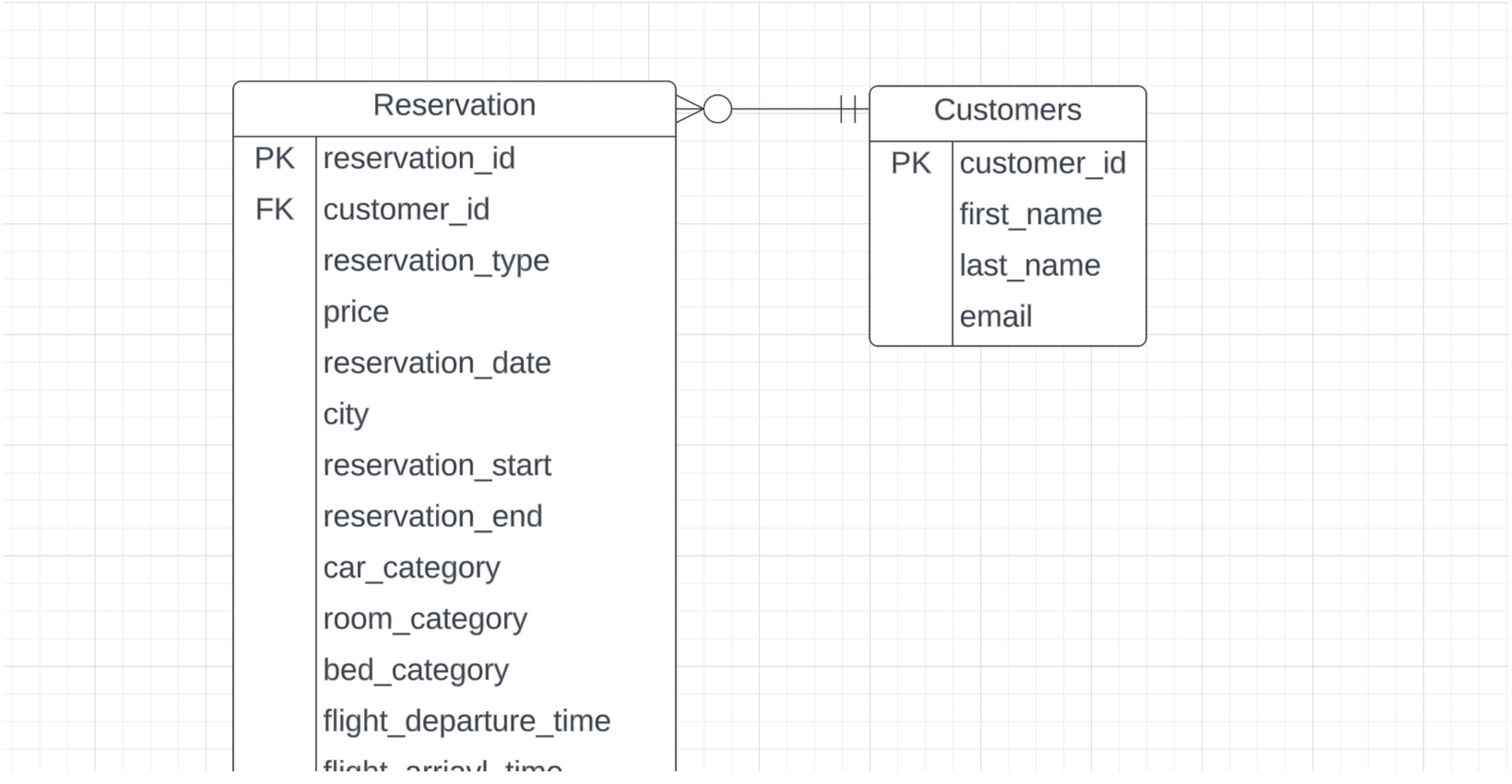

Reservation
PK reservation_id
FK customer_id
reservation_type
price
reservation_date
Customers
PK customer_id
first_name
last_name
email
Disjoint
Complete
CarReservation
city
reservation_start
reservation_end
car_category
FlightReservation
flight_departure_time
flight_arrival_time
departure_airport
arrival_airport
HotelReservation
city
reservation_start
reservation_end
room_category
bed_category

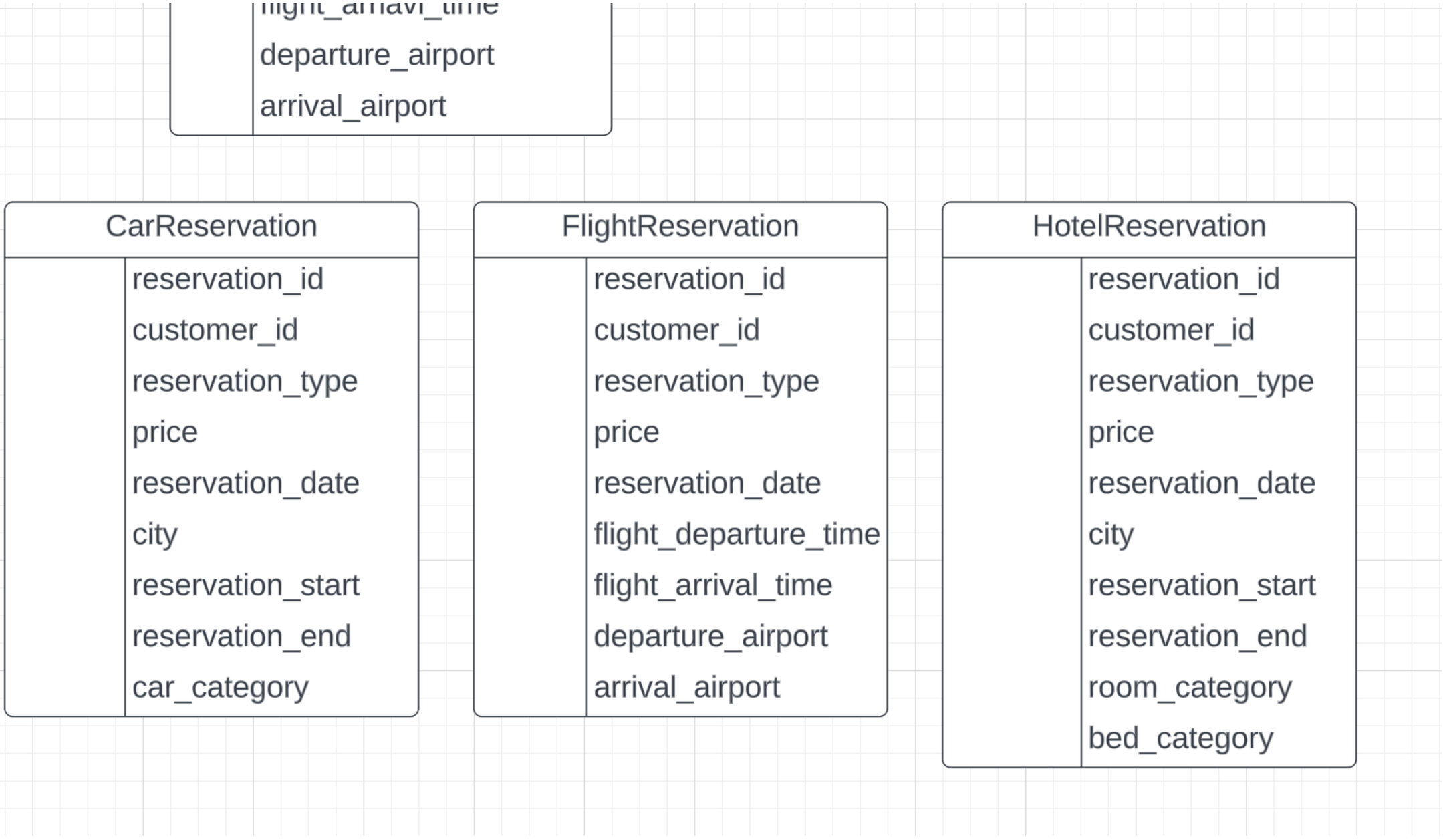
Implement

- *You must draw a logical ER diagram for the four tables you will create. Your diagram does not have to include the view.*
- *Your diagram must include primary keys and foreign keys.*

Diagram

One table approach




flight_arrival_time
departure_airport
arrival_airport
CarReservation
reservation_id
customer_id
reservation_type
price
reservation_date
city
reservation_start
reservation_end
car_category
FlightReservation
reservation_id
customer_id
reservation_type
price
reservation_date
flight_departure_time
flight_arrival_time
departure_airport
arrival_airport
HotelReservation
reservation_id
customer_id
reservation_type
price
reservation_date
city
reservation_start
reservation_end
room_category
bed_category

- *Save the two dataframes into two tables (using pandas `to_sql` and schema `s23_w4111_hw2_p1_<uni>`)*
  - `customers`
  - `reservations`
- *Use SQL to create the tables and data for the three types of reservation.*
- *Set appropriate data types and keys for the tables.*
- *Show and execute your SQL for creating the tables and modifying the schema.*

In [81]:
```
%sql drop schema s23_w4111_hw2_p1_hw2888
%sql create schema s23_w4111_hw2_p1_hw2888
%sql use s23_w4111_hw2_p1_hw2888
```

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

Out[81]:
```
[]
```

In [82]:
```
customers_df = pandas.read_csv('./customers-3.csv')
customers_df.to_sql('customers', con=engine, schema='s23_w4111_hw2_p1_hw2888',
                    if_exists='replace', index=False)
```

```
reservations_df = pandas.read_csv("./reservations-3.csv")
reservations_df.to_sql('reservations', con=engine, schema='s23_w4111_hw2_p1_hw2888',
                       if_exists='replace', index=False)
```

Out[82]:
```
1000
```

PK should handle the unique and not null constriant

In [83]:
```
%%sql
alter table customers modify customer_id varchar(64) not null;
alter table customers modify first_name varchar(50) not null;
alter table customers modify last_name varchar(50) not null;
alter table customers modify email varchar(50) not null;

alter table customers add primary key (customer_id);
```

```
 * mysql+pymysql://root:***@localhost
1000 rows affected.
1000 rows affected.
1000 rows affected.
1000 rows affected.
0 rows affected.
```

Out[83]:
```
[]
```

Add primary key and foreign key for reservations

In [84]:
```
%%sql
alter table reservations modify reservation_id varchar(64) not null;
alter table reservations modify customer_id varchar(64) not null;

alter table reservations add primary key (reservation_id);
alter table reservations add foreign key (customer_id) references customers(customer_id)
```

```
 * mysql+pymysql://root:***@localhost
1000 rows affected.
1000 rows affected.
0 rows affected.
1000 rows affected.
```

Out[84]:
```
[]
```

Change price to double and drop the dollar sign, this will make more sense when recording currency.

In [85]:
```
%%sql
update reservations
set price=replace(price, '$', '');
alter table reservations modify price double(10, 2) not null;
```

```
 * mysql+pymysql://root:***@localhost
1000 rows affected.
```

Out[85]:
```
[]
```

In [86]:
```
%%sql
alter table reservations modify city varchar(50);
alter table reservations modify departure_airport varchar(3);
alter table reservations modify arrival_airport varchar(3);
```

```
 * mysql+pymysql://root:***@localhost
1000 rows affected.
1000 rows affected.
1000 rows affected.
```

Out[86]:
```
[]
```

Change string to date

In [87]:
```
%%sql
update reservations set reservation_date=STR_TO_DATE(reservation_date, '%m/%e/%Y');
update reservations set reservation_start=STR_TO_DATE(reservation_start, '%m/%e/%Y');
update reservations set reservation_end=STR_TO_DATE(reservation_end, '%m/%e/%Y');

alter table reservations modify reservation_date date not null;
alter table reservations modify reservation_start date;
alter table reservations modify reservation_end date;
```

```
 * mysql+pymysql://root:***@localhost
1000 rows affected.
1000 rows affected.
1000 rows affected.
1000 rows affected.
1000 rows affected.
1000 rows affected.
```

Out[87]:
```
[]
```

Find enum value for each enum data type attribute

In [88]:
```
%%sql
select distinct reservation_type
from reservations;
```

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

Out[88]:

reservation_type
Flight
Car
Hotel

In [89]:
```
%%sql
select distinct car_category
from reservations;
```

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

Out[89]:

car_category
None
SUV
Midsize
Compact

In [90]:
```
%%sql
select distinct room_category
from reservations;
```

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

Out[90]:

room_category
None
Suite
One Room
Studio

In [91]:
```
%%sql
select distinct bed_category
from reservations;
```

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

Out[91]:

bed_category
None
King
Two Doubles

In [92]:
```
%%sql
alter table reservations modify reservation_type enum('Flight', 'Car', 'Hotel') not null;
alter table reservations modify car_category enum('SUV', 'Midsize', 'Compact');
alter table reservations modify room_category enum('Suite', 'One Room', 'Studio');
alter table reservations modify bed_category enum('King', 'Two Doubles');
```

```
 * mysql+pymysql://root:***@localhost
1000 rows affected.
1000 rows affected.
1000 rows affected.
1000 rows affected.
```

Out[92]:
```
[]
```

Change the string to 24 hour time format

In [93]:
```
%%sql
update reservations set flight_departure_time=STR_TO_DATE(flight_departure_time,  "%h:%i %p");
update reservations set flight_arrival_time=STR_TO_DATE(flight_arrival_time,  "%h:%i %p");
alter table reservations modify flight_departure_time time;
alter table reservations modify flight_arrival_time time;
```

```
 * mysql+pymysql://root:***@localhost
1000 rows affected.
1000 rows affected.
1000 rows affected.
1000 rows affected.
```

Out[93]:
```
[]
```

Show the table after setting appropriate data type

In [94]:
```
%sql describe customers
```

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

Out[94]:

Field	Type	Null	Key	Default	Extra
customer_id	varchar(64)	NO	PRI	None	
first_name	varchar(50)	NO		None	
last_name	varchar(50)	NO		None	
email	varchar(50)	NO		None	

In [95]:
```
%sql describe reservations
```

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

Out[95]:

Field	Type	Null	Key	Default	Extra
reservation_id	varchar(64)	NO	PRI	None	
customer_id	varchar(64)	NO	MUL	None	
reservation_type	enum('Flight','Car','Hotel')	NO		None	
price	double(10,2)	NO		None	
reservation_date	date	NO		None	
city	varchar(50)	YES		None	
reservation_start	date	YES		None	
reservation_end	date	YES		None	
car_category	enum('SUV','Midsize','Compact')	YES		None	
room_category	enum('Suite','One Room','Studio')	YES		None	
bed_category	enum('King','Two Doubles')	YES		None	
flight_departure_time	time	YES		None	
flight_arrival_time	time	YES		None	
departure_airport	varchar(3)	YES		None	
arrival_airport	varchar(3)	YES		None	

Create View for each of the 3 reservations

In [96]:
```
%%sql
create or replace view car_reservation as
select reservation_id, customer_id, reservation_type, price, reservation_date,
city, reservation_start, reservation_end, car_category
from reservations
where reservation_type = 'Car'
```

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

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

In [97]:
```
%%sql
create or replace view hotel_reservation as
select reservation_id, customer_id, reservation_type, price, reservation_date,
city, reservation_start, reservation_end, room_category, bed_category
from reservations
where reservation_type = 'Hotel'
```

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

Out[97]:
```
[]
```

In [98]:
```
%%sql
create or replace view flight_reservation as
select reservation_id, customer_id, reservation_type, price, reservation_date,
flight_departure_time, flight_arrival_time, departure_airport, arrival_airport
from reservations
where reservation_type = 'Flight'
```

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

Out[98]:
```
[]
```

Show example for each view

In [99]:
```
%%sql
SELECT *
from car_reservation limit 10;
```

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

Out[99]:

reservation_id	customer_id	reservation_type	price	reservation_date	city	reservation_start	reservation_end	car_category
00286f61-0cab-4d84-a3f6-06f62d905421	04c8292b-f332-4f68-a5b9-23048bfd0158	Car	901.91	2022-07-16	Beiwa	2022-11-17	2022-08-07	SUV
0040aecb-be4a-403e-9f14-d9fc034745ed	b1493d4e-8d20-4f9f-bcb3-cb74297409e1	Car	227.57	2022-04-20	Bileća	2022-12-27	2022-03-15	SUV
01bb4d70-4080-433b-a8ab-0076092366b6	3dc1ffc5-a2ff-4194-abdf-c6b20a91955f	Car	373.72	2022-12-04	Gävle	2022-09-22	2023-01-04	Midsize
03bd05af-eee8-40da-a826-0acf3ae1a693	10653a51-d23e-4e77-9e77-560cce6f4de0	Car	286.06	2022-05-10	Chernelytsya	2022-09-30	2022-08-06	SUV
05a82fa8-6356-48f4-91bf-2fd9a82c60e4	8db8ff96-6089-4dce-9e2f-d98405abbc8e	Car	256.83	2022-07-25	Alkmaar	2022-04-13	2022-02-22	SUV
05fb6841-470e-4e84-b4f8-7dd07265c5e3	fd05f02c-89d6-4278-bc74-48aeeaf4cf23	Car	942.55	2022-04-19	Changzheng	2022-09-19	2022-07-06	Midsize
07529b64-8491-4b97-84fe-4db97f317394	790b5b32-7455-4fb2-b8c9-2e6b2ba744e7	Car	792.48	2022-04-06	Mojo	2022-07-23	2023-01-19	SUV
077a3d5c-f759-43e6-9c81-ce48a846014d	760301aa-24bd-44a9-b3d7-4b893f0c9e92	Car	232.14	2022-03-25	Néa Sánta	2022-09-18	2022-03-10	Midsize
08434a56-921d-4fe3-ac0e-20ccfbbdaa81	93b70aac-9725-4c72-91b1-0259a20a2453	Car	682.98	2023-02-01	Huasta	2022-09-04	2022-05-22	Midsize
0b4db617-6b35-4bcd-a280-2ed8ac6aa5ec	6071a446-fd8e-4482-9a17-5565f5b41fae	Car	316.74	2022-06-16	Yepocapa	2022-10-01	2022-08-05	SUV

In [100…
```
%%sql
SELECT *
from hotel_reservation limit 10;
```

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

Out[100]:

reservation_id	customer_id	reservation_type	price	reservation_date	city	reservation_start	reservation_end	room_category	bed_category
010eb023-4fe2-42f7-bac4-35a7e4290b6b	40085bc3-80e8-4b8e-83ec-1ac7b524da29	Hotel	428.67	2022-10-23	Jingjiao	2023-01-02	2022-10-05	Suite	King
01601bea-efa9-4784-95bc-d124affe91c4	cec93f53-7984-438c-8ce1-f3e5f340cfcb	Hotel	200.43	2022-08-14	Xidajie	2022-12-10	2022-12-21	One Room	Two Doubles
021998b0-171f-4eb8-86ce-16e753277d9a	add731ae-a6e6-4698-b2d9-17cdffcaa095	Hotel	715.87	2022-11-29	Belo Jardim	2022-08-15	2022-10-25	One Room	Two Doubles
056ab12e-8609-4239-947f-27b77d818f24	84fdad0d-0835-40a0-ac3f-ded127305848	Hotel	554.83	2022-05-07	Třinec	2022-03-07	2022-10-23	Studio	King
07cf8a9b-75e9-4717-9d85-f672a5824834	66394ac3-cbeb-44b1-8933-8df224d91817	Hotel	608.56	2022-05-02	Zaozerne	2022-04-19	2022-04-26	Studio	Two Doubles
0b292693-adfd-4313-bca0-48e1f08345f6	d9544b31-d41e-46d3-ae95-2fb0d71b539b	Hotel	558.76	2022-03-08	Turija	2022-03-28	2022-11-30	One Room	King
0b6d6ce9-493a-4b7a-bd8c-538e3f595a64	5da8e62a-26ad-4497-998d-3c714161e7bb	Hotel	322.19	2022-03-08	Jiedu	2022-12-10	2022-09-18	Studio	King
0be9d85a-feb4-432e-98a7-ee016a8f1db0	51f59855-47d3-4ba7-8478-5ebefa8f7d93	Hotel	221.48	2023-01-07	Dūāb	2022-09-30	2023-01-18	Studio	Two Doubles
0c015520-51fd-492e-af49-e42d094af906	0f07114b-7bd3-456d-bb63-51a2ba5c7617	Hotel	343.86	2022-05-04	Roissy Charles-de-Gaulle	2022-06-28	2023-01-16	Studio	Two Doubles
0c463afb-3079-46f0-9214-bc24a026c2ec	ae9c8fbc-c47d-477f-881a-3278ed53d3c1	Hotel	419.14	2023-01-02	Penang	2022-05-23	2022-06-02	Suite	Two Doubles

In [101…

```
%%sql
SELECT *
from flight_reservation limit 10;
```

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

Out[101]:

reservation_id	customer_id	reservation_type	price	reservation_date	flight_departure_time	flight_arrival_time	departure_airport	arrival_airport
001df543-977c-4f59-9547-8d097a49c6d5	2d1fc0e9-ec65-4e13-981c-06428f24d820	Flight	143.17	2022-06-30	2:08:00	5:52:00	MSW	PLS
0066306f-1691-4d3f-9883-7b7dcbbb57a0	79a536d2-dc93-4247-84fb-13fae6dea683	Flight	348.16	2022-06-26	14:21:00	18:53:00	TYR	DOV
01a4b0ee-2b79-43a9-9ce4-f06c944e7ab7	59f379e9-a981-4bb6-bc1e-6dfee479cf37	Flight	861.07	2022-04-23	6:21:00	4:21:00	LSN	WMN
03412b08-6698-45db-9439-21d0ce091928	9b099903-06f9-45a0-b630-850eba4aed19	Flight	306.88	2022-04-21	20:37:00	7:54:00	YYZ	BER
037a0420-8f36-4a15-aee4-ceef249bea5c	5240a226-cb67-49fe-9a08-5fb12ffc0baa	Flight	518.37	2022-08-16	17:24:00	21:58:00	KBC	AVG
04ba2594-3556-4f1f-9f27-f335a72df786	fe0a335c-6ced-45c7-a30f-23b83a2d1147	Flight	188.93	2023-02-09	6:48:00	16:23:00	NHF	BJL
05978490-2889-4ef0-ac1c-82236c90e905	c01d2573-e406-4490-9955-53e8ea32b62f	Flight	512.72	2022-10-16	21:27:00	21:52:00	BLN	ERA
06bc222a-5301-4066-84fb-074849517c5c	c227de5d-bda5-4d7b-bbbe-5c7f1d110141	Flight	906.91	2022-07-20	0:52:00	16:02:00	KOL	HAH
07fb30e9-9e25-4138-8137-440fa79da669	6ee8ccf5-4c32-4f58-99e1-fa907e5ae56b	Flight	783.12	2023-01-26	11:14:00	21:43:00	ACD	KMQ
0ad3083e-5d46-42e3-9f3f-065d5bb0f9db	f8e538cf-5b2a-41ce-843b-33122970c2ba	Flight	772.81	2022-07-24	4:13:00	8:53:00	GUV	ARE