COMS W 4111-002
W4111 - Introduction to Databases, Section 002, Fall 2021
Take Home Final

Exam Instructions

Overview

The Final Exam is worth 30 points out of the semester's total points. There are 10 questions of varying difficulty worth varying points. The amount of points is not necessarily indicative of difficulty/length of the question, but you can use it as a rough guide. The grade for the final is in the range 0-100. We map the score to final point by multiplying by $\frac{30}{100}$.

The Final Exam is open note, open book, open internet. You **may not collaborate** with other students. Posts on EdStem must be made private for you and the instructors only. Any common questions or clarifications will be made by the instructors on the Final Exam pinned thread. Students **are responsible** for monitoring the thread for corrections and clarifications.

You must cite any online sources in the comments Markdown cell for each questions.

Overview of Questions

1. Written — Core Databases Concepts (10 pts)
2. Relational Algebra (10 pts)
3. SQL Design and Query (10 pts)
4. Neo4j Design and Query Queries (10 pts)
5. MongoDB Design and Query (10 pts)
6. Implementation Scenario 1: Modeling and Implementing [RACI] in a Database (15 pts)
7. ~~Implementation Scenario 2: Data Model Comparisons (20 pts)~~
8. Impementation Scenario 3: Data Model Transformation (15 pts)

Note: I decided to drop the data model comparison to make the exam easier. So, everyone get's a free 20 points. Also, remember that **I never curve down.**

Submission Information

This exam is **due Sunday, December 19 at 11:59pm ET** to Gradescope. **You may not use Late Days.**

You submit a zip file containing the main Jupyter Notebook (this file), a PDF of this notebook, and several files in the folder. Each questions provides detailed instructions of how to complete

the question.

Your PDFs must be high enough resolution that the text is legible. It must be printed onto standard 8.5x11in pages. Any images that you embed MUST be visible in the PDF. Do not use HTML to embed your images or they will not be visible when you export to PDF.

Failure to meet these formatting specifications will result in a 0.

As always, respect for the individual is paramount. We will accommodate special circumstances, but we must be notified and discuss in advance.

Environment Setup and Test

Note: If you have already done the environment setup tests and succeeded, you only need to run the cells that:

1. Import `mysql_check`, `neo4j_check` and `mongodb_check`.
2. Run the cells that set the DB connection information (user ID, password, URL, ...) for the various databases.
3. You can go directly to the questions.

Instructions

This section tests your environment. You **MUST** completely follows and comply with the instructions.

Implementation Files

- Several of the questions requiring calling databases from Python code. The python code is simple and implements database queries and operations. This complies with the department's guidelines for *non-programming*.
- There is a section for testing access to each of MySQL, MongoDB and Neo4j. You **must** have installed or have access to the databases, and if locally installed the database must be running.

MySQL

- Download and load the Classic Models database into MySQL. The download site provides installation instructions.
- The comments in the code snippets below provide instructions for completing and executing each cell.

In [1]:
```
# Import the MySQL test and implementation template/helper functions from the local di
# You do not need to modify this cell. You only need to implement it.
```

```
#
import mysql_check
```

In [2]:

```
#
# Call the function below to set the user, password and host for your instance of MySQ
# YOU MUST set the variables to the correct names for instance.
#
db_user = "root"
db_password = "Zyj7725869"
db_host = "localhost"

mysql_check.set_connect_info(db_user, db_password, db_host)
```

In [3]:

```
#
# Execute the code below. Your answer should be the same as the exampple out.
#
df = mysql_check.test_pymysql()
df
```

Out[3]:

	Tables_in_classicmodels
0	customers
1	employees
2	offices
3	orderdetails
4	orders
5	payments
6	productlines
7	products

In [4]:

```
#
# Execute the cell below. Your result should match the example.
#
result_df = mysql_check.test_sql_alchemy()
result_df
```

Out[4]:

	customerNumber	customerName	country
0	103	Atelier graphique	France
1	119	La Rochelle Gifts	France
2	146	Saveley & Henriot, Co.	France
3	171	Daedalus Designs Imports	France
4	172	La Corne D'abondance, Co.	France
5	209	Mini Caravy	France
6	242	Alpha Cognac	France
7	250	Lyon Souveniers	France
8	256	Auto Associés & Cie.	France

	customerNumber	customerName	country
9	350	Marseille Mini Autos	France
10	353	Reims Collectables	France
11	406	Auto Canal+ Petit	France

Neo4j

In [5]:
```
#
# Run this cell.
#
import neo4j_check
```

In [7]:
```
#
# Set the neo4j user and password for connecting to your database. The user is probabl
# You set the password when you created the project and graph.
#
db_user = "neo4j"
db_password = "Zyj7725869"

neo4j_check.set_neo4j_connect_info(db_user, db_password)
```

In [8]:
```
#
# You database MUST have the Movie DB installed. You had to do this for HW3. Running t
# the sample output.
#
res = neo4j_check.get_people_in_matrix()
res
```

Out[8]:

	name	born
0	Keanu Reeves	1964
1	Emil Eifrem	1978
2	Carrie-Anne Moss	1967
3	Hugo Weaving	1960
4	Laurence Fishburne	1961

MongoDB

In [9]:
```
# Import the MongoDB test and helper functions.
#
import mongodb_check
```

In [10]:
```
#
# Set the connection URL to get to your instance of MongoDB. You have used this URL
# in HW3 and when using Mongo Compass.
#
connect_url = "mongodb://localhost:27017/"
mongodb_check.set_connect_url(connect_url)
```

In [13]:
```
#
# Run the following function. This will load information into MongoDB and test the loa
#
df = mongodb_check.load_and_test_mongo()
df
```

Out[13]:

	_id	customerNumber	customerName	country
0	61b6a499b2283066b3199b9c	103	Atelier graphique	France
1	61b6a499b2283066b3199b9f	119	La Rochelle Gifts	France
2	61b6a499b2283066b3199ba9	146	Saveley & Henriot, Co.	France
3	61b6a499b2283066b3199bb2	171	Daedalus Designs Imports	France
4	61b6a499b2283066b3199bb3	172	La Corne D'abondance, Co.	France
5	61b6a499b2283066b3199bc1	209	Mini Caravy	France
6	61b6a499b2283066b3199bcb	242	Alpha Cognac	France
7	61b6a499b2283066b3199bce	250	Lyon Souveniers	France
8	61b6a499b2283066b3199bcf	256	Auto Associ茅s & Cie.	France
9	61b6a499b2283066b3199beb	350	Marseille Mini Autos	France
10	61b6a499b2283066b3199bec	353	Reims Collectables	France
11	61b6a499b2283066b3199bfa	406	Auto Canal+ Petit	France

1. Database Core Concepts (10 points)

- There is a Google Doc.
- Make a copy of the Google Doc. Answer the questions in the document. You will submit a PDF of the document and your answers in the zip file you submit. The file must be in the folder and name **question1.pdf.**

2. Relational Algebra

Instructions

You will use the online relational calculator to answer some of the subquestions. For these questions, your answer must contain:

- The text of the relational statement. The TAs may cut, paste and run the statement and it must work.
- An image showing the results of your execution.
- There is an example below.

Example

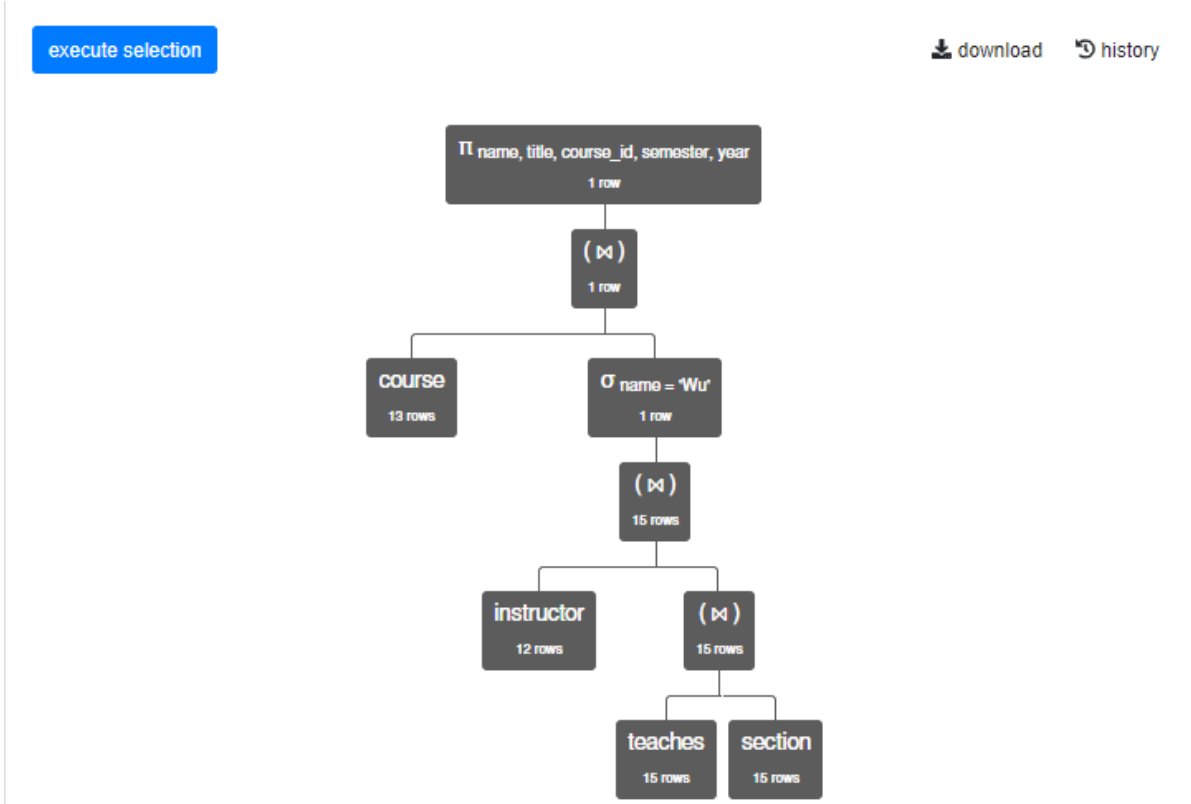
Question

- Use the "Silberschatz - UniversityDB" for this question.
- Professor Wu taught only one section. Produce the following information for the section.

instructor.name	course.title	course.course_id	teaches.semester	teaches.year
'Wu'	'Investment Banking'	'FIN-201'	'Spring'	2010

Answer

```
π name, title, course_id, semester, year
    (course ⋈ (σ name='Wu' (instructor ⋈ (teaches ⋈ section))))
```



instructor.name	course.title	course.course_id	teaches.semester	teaches.year
'Wu'	'Investment Banking'	'FIN-201'	'Spring'	2010

2.1 Relation Model Schema (2 points)

Question

- The following is a simple MySQL table definition.

```
CREATE TABLE `new_table` (
  `product_category` INT NOT NULL,
  `produce_code` VARCHAR(45) NOT NULL,
  `product_name` VARCHAR(45) NULL,
```

```
`product_description` VARCHAR(45) NULL,
PRIMARY KEY (`product_category`, `produce_code`));
```

- Using the notation from chapter 2 slides for defining a relational schema, provide the corresponding relation schema definition.
  - Ignore the column types, `NOT NULL`, etc.
  - Two under-bar text, you can use `$\underline{cat}$` to produce $\underline{cat}$.

Answer (In Markdown cell below)

new_table($\underline{product\_category}$,$\underline{produce\_code}$,product_name,product_description)

2.2 Relational Algebra (4 points)

Question

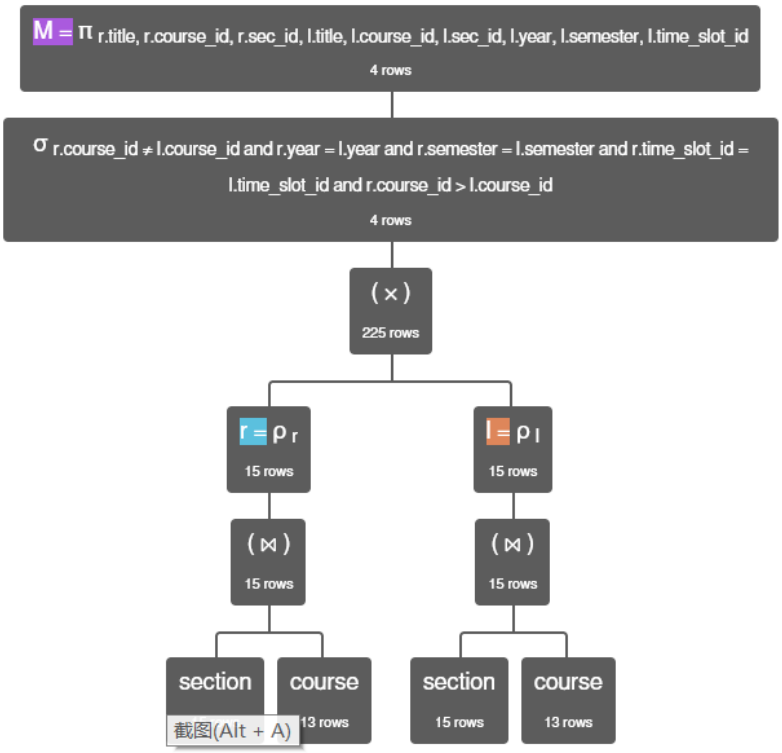
- Provide your answer following the examples' format.
- Use the RelaX calculator the "Silberschatz - UniversityDB" for this question.
- A section `r` overlaps with another section `l` if and only if: They occured at the same time (`year`, `semester`, `time_salot_id`).
- Produce the following table that shows the overlapping sections.

r.title	r.course_id	r.sec_id	l.title	l.course_id	l.sec_id	l.year	l.semester	l.time_slot_id
'Image Processing'	'CS-319'	2	'World History'	'HIS-351'	1	2010	'Spring'	'C'

Answer

Answer

```
r = ρ r (section ⋈ course)
l = ρ l (section ⋈ course)
M = π
r.title,r.course_id,r.sec_id,l.title,l.course_id,l.sec_id,l.year,l.semester,l
 ( σ r.course_id ≠ l.course_id ∧ r.year=l.year ∧
r.semester=l.semester ∧ r.time_slot_id=l.time_slot_id ∧
r.course_id>l.course_id ( r × l ))
M
```



$$\pi\ r.title, r.course\_id, r.sec\_id, l.title, l.course\_id, l.sec\_id, l.year, l.semester, l.time\_slot\_id\ (\ \sigma\ r.course\_id \neq l.course\_id \text{ and } r.year = l.year \text{ and } r.semester = l.semester \text{ and } r.time\_slot\_id = l.time\_slot\_id \text{ and } r.course\_id > l.course\_id\ (\ \rho_{\ r}\ (\ section \bowtie course\ ) \times \rho_{\ l}\ (\ section \bowtie course\ )\ )\ )$$

r.title	r.course_id	r.sec_id	l.title	l.course_id	l.sec_id	l.year	l.semester	l.time_slot_id
'Investment Banking'	'FIN-201'	1	'Image Processing'	'CS-319'	1	2010	'Spring'	'B'
'World History'	'HIS-351'	1	'Image Processing'	'CS-319'	2	2010	'Spring'	'C'
'Music Video Production'	'MU-199'	1	'Robotics'	'CS-315'	1	2010	'Spring'	'D'
'Physical Principles'	'PHY-101'	1	'Database System Concepts'	'CS-347'	1	2009	'Fall'	'A'

2.3 Relational Algebra (4 points)

Question

- The relation algebra has additional operators for ordering, aggregation/group by, etc.
- A simple analysis of the data in the data in "Silberschatz - UniversityDB" shows that the `takes` table must be incomplete. Produce the following table, where `sum_of_credits` is the sum of a student's credits based on the information in `takes`.

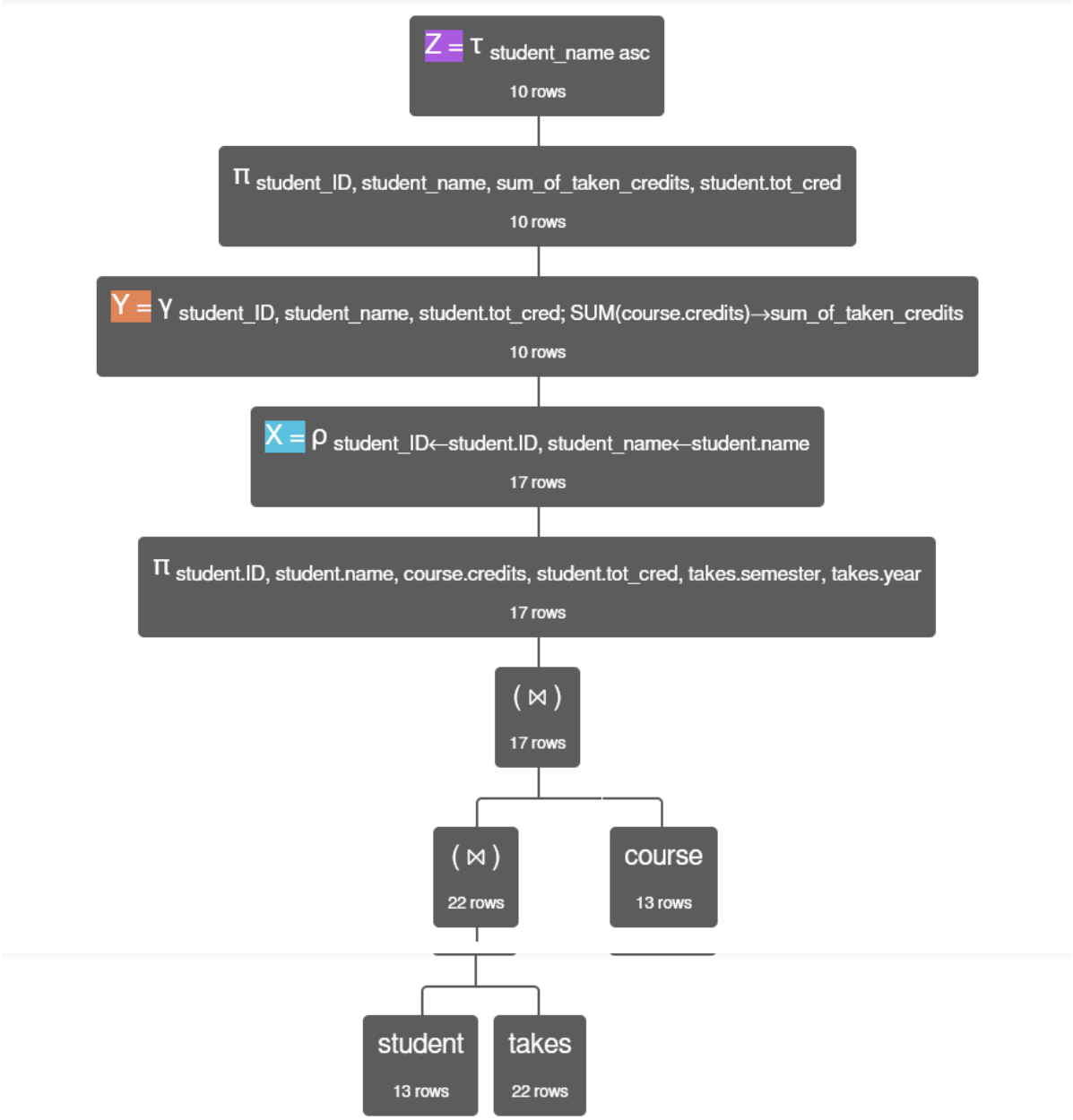
student_ID	student_name	sum_of_taken_credits	student.tot_cred

student_ID	student_name	sum_of_taken_credits	student.tot_cred
76653	'Aoi'	3	60
19991	'Brandt'	3	80
76543	'Brown'	7	58
23121	'Chavez'	3	110
44553	'Peltier'	4	56
55739	'Sanchez'	3	38
12345	'Shankar'	14	32
98988	'Tanaka'	8	120
54321	'Williams'	8	54
128	'Zhang'	7	102

Answer

Answer

```
X = ρ student_ID←student.ID,student_name←student.name (π
student.ID,student.name,course.credits,student.tot_cred,takes.semester,takes.
 ((student ⋈ takes) ⋈ course))
Y = γ
student_ID,student_name,student.tot_cred;sum(course.credits)→sum_of_taken_cre
 (X)
Z = τ student_name asc (π
student_ID,student_name,sum_of_taken_credits,student.tot_cred (Y))
Z
```



τ student_name asc (π student_ID, student_name, sum_of_taken_credits, student.tot_cred γ student_ID, student_name, student.tot_cred; SUM(course.credits)→sum_of_taken_credits ρ student_ID←student.ID, student_name←student.name (π student.ID, student.name, course.credits, student.tot_cred, takes.semester, takes.year ((student ⋈ takes) ⋈ course)))

student.student_ID	student.student_name	sum_of_taken_credits	student.tot_cred
76653	'Aoi'	3	60
19991	'Brandt'	3	80
76543	'Brown'	7	58
23121	'Chavez'	3	110
44553	'Peltier'	4	56
55739	'Sanchez'	3	38
12345	'Shankar'	14	32
98988	'Tanaka'	8	120
54321	'Williams'	8	54
128	'Zhang'	7	102

3. SQL Query

Instructions and Example

You must follow and comply with the instructions for completing the questions in this section. Any deviation from the format is a score of 0.

1. The zip file you downloaded contains a file `question_2_sql.py` . The file contains:
   - An example of the format and approach to answers.
   - An empty function for each answer. You answer the question by completing the function's implementation.

1. The sample returns a Pandas data frame with `customerNumber, customerName` and `Country.` The country is a parameter to the function call.

In [1]:
```
import question_3_sql
```

In [5]:
```
#
# Call the function with the parameter France and display the resulting data frame.
#
result = question_3_sql.question_3_example_get_customers('France')
result
```

Out[5]:

customerNumber	customerName	country

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

3.1 Revenue by Country (2 points)

Question

1. An `order` is a a set of `orderdetails.`
2. The value/revenue for an `orderdetails` is `priceEach*quantityOrdered`
3. The value/revenue for an `order` is the sum of the value/revenue of the `orderdetails.`

- Implement the function `revenue_by_country` . We provide an example for the output. The company can only claim revenue if the order has `shipped.`

Answer

In [3]:
```
result = question_3_sql.question_3_revenue_by_country()
result
```

Out[3]:

	revenue	country
0	3032204.26	USA
1	196470.99	Germany
2	270846.30	Norway
3	947470.01	Spain
4	176791.44	Denmark
5	360616.81	Italy
6	87468.30	Philippines
7	391503.90	UK

	revenue	country
8	120457.09	Sweden
9	965750.58	France
10	91471.03	Belgium
11	263997.78	Singapore
12	161418.16	Austria
13	509385.82	Australia
14	416114.03	New Zealand
15	295149.35	Finland
16	205911.86	Canada
17	45480.79	Hong Kong
18	167909.95	Japan
19	49898.27	Ireland
20	108777.92	Switzerland

3.2 Customer Payments and Customer Purchases (2 points)

Question

1. `classicmodels.payments` records customer payments.

2. You can use the the formula above for computing the cost of an order.

3. The total owed by a customer is the total value/revenue for all orders. For the purposes of this problem, you should include all orders and not just the ones that shipped.

4. Implement the functions `purchases_and_payments.` The function returns a data frame with the following columns.
   - `customerNumber`
   - `customerName`
   - `total_spent` is the total value/cost over all orders by the customer.
   - `total_payments` is the total paid by the customer over all payments.
   - `total_unpaid` is the difference between `total_spent` and `total_payments.`

5. Order the result by `customerName.`

6. You must use at least one sub-query in your answer.

Answer

In [2]:
```
#
# Execute this cell to display your answer.
#
```

```
result = question_3_sql.question_3_purchases_and_payments()
result
```

Out[2]:

	customerNumber	customerName	total_spent	total_payments	total_unpaid
0	242	Alpha Cognac	60483.36	60483.36	0.00
1	168	American Souvenirs Inc	0.00	0.00	0.00
2	249	Amica Models & Co.	82223.23	82223.23	0.00
3	237	ANG Resellers	0.00	0.00	0.00
4	276	Anna's Decorations, Ltd	137034.22	137034.22	0.00
...	...	...	...	...	...
117	298	Vida Sport, Ltd	108777.92	108777.92	0.00
118	181	Vitachrome Inc.	72497.64	72497.64	0.00
119	144	Volvo Model Replicas, Co	66694.82	43680.65	23014.17
120	459	Warburg Exchange	0.00	0.00	0.00
121	475	West Coast Collectables Co.	43748.72	43748.72	0.00

122 rows × 5 columns

3.3 What Customers Buy What? (1 point)

Question

1. `Products` are in `productLines`.

2. Product a table that contains the `customerNumber` and `customerName` for all customers that have not orders a product from line `Planes` and not ordered a product from line `Trucks and Buses`.

Answer

In [4]:

```
#
# Run the cell below.
#
result = question_3_sql.question_3_customers_and_lines()
result
```

Out[4]:

	customerNumber	customerName
0	103	Atelier graphique
1	112	Signal Gift Stores
2	125	Havel & Zbyszek Co
3	168	American Souvenirs Inc
4	169	Porto Imports Co.
5	171	Daedalus Designs Imports
6	187	AV Stores, Co.

	customerNumber	customerName
7	201	UK Collectables, Ltd.
8	204	Online Mini Collectables
9	206	Asian Shopping Network, Co
10	209	Mini Caravy
11	219	Boards & Toys Co.
12	223	Natürlich Autos
13	237	ANG Resellers
14	247	Messner Shopping Network
15	259	Toms Spezialitäten, Ltd
16	273	Franken Gifts, Co
17	293	BG&E Collectables
18	298	Vida Sport, Ltd
19	303	Schuyler Imports
20	307	Der Hund Imports
21	314	Petit Auto
22	335	Cramer Spezialitäten, Ltd
23	344	CAF Imports
24	348	Asian Treasures, Inc.
25	350	Marseille Mini Autos
26	356	SAR Distributors, Co
27	357	GiftsForHim.com
28	361	Kommission Auto
29	362	Gifts4AllAges.com
30	369	Lisboa Souveniers, Inc
31	376	Precious Collectables
32	406	Auto Canal+ Petit
33	409	Stuttgart Collectable Exchange
34	443	Feuer Online Stores, Inc
35	456	Microscale Inc.
36	459	Warburg Exchange
37	465	Anton Designs, Ltd.
38	477	Mit Vergnügen & Co.
39	480	Kremlin Collectables, Co.
40	481	Raanan Stores, Inc
41	486	Motor Mint Distributors Inc.

4 MongoDB

Instructions and Example

You must follow and comply with the instructions for completing the questions in this section. Any deviation from the format is a score of 0.

1. The final exam folder has a subdirectory `MongoDB` that contains MongoDB collections dumped in JSON format.
   - actors_imdb.json
   - got_characters.json
   - got_episodes.json
   - imdb_titles.json
   - title_ratings.json

2. Use MongoDB Compass:
   - Create a MongoDB database F21_Final.
   - Import the data from the files into collections. You can do this by using MongoDB Compass to create a collection, and then selecting the import data function.

3. You will implement your answers in functions in the file `

1. The sample returns a data frame of the form `(seasonNum, episodeNum, sceneNum, characterName)` for the characters that appeared in season one, episode one.

In [1]:
```
import question_4_mongo
```

In [6]:
```
result = question_4_mongo.question_4_example()
result
```

Out[6]:

	seasonNum	episodeNum	sceneNum	characterName
0	1	1	1	Gared
1	1	1	1	Waymar Royce
2	1	1	1	Will
3	1	1	2	Gared
4	1	1	2	Waymar Royce
...	...	...	...	...
148	1	1	35	Summer
149	1	1	36	Bran Stark
150	1	1	36	Summer
151	1	1	36	Jaime Lannister
152	1	1	36	Cersei Lannister

153 rows × 4 columns

4.1 Implementing a JOIN (2 points)

Question

1. You will need to implement an aggregation for this problem. You can use MongoDB Compass to produce and test the aggregation, and then copy into the implementation template.

2. The aggregation operator `$lookup` implements a join-like function for MongoDB.

3. The aggregation operator (in a project) for getting substrings is `$substr`.

4. Write a query that joins episodes and ratings and produces a list of documents of the form:
   - `seasonNum, episodeNum, episodeTitle, episodeDescription, episodeDate` from `got_episodes.`
   - `tconst, averageRating, numVotes` from title ratings.

Answer

In [2]:
```
#
# Run your test here.
#
result = question_4_mongo.question_4_ratings()
result
```

Out[2]:

	seasonNum	episodeNum	episodeTitle	episodeDescription	episodeDate	tconst	averageRati
0	1	1	Winter Is Coming	Jon Arryn, the Hand of the King, is dead. King...	2011-04-17	tt1480055	
1	1	2	The Kingsroad	While Bran recovers from his fall, Ned takes o...	2011-04-24	tt1668746	
2	1	3	Lord Snow	Lord Stark and his daughters arrive at King's ...	2011-05-01	tt1829962	
3	1	4	Cripples, Bastards, and Broken Things	Eddard investigates Jon Arryn's murder. Jon be...	2011-05-08	tt1829963	
4	1	5	The Wolf and the Lion	Catelyn has captured Tyrion and plans to bring...	2011-05-15	tt1829964	
...	...	...	...	...	...	...	
68	8	2	A Knight of the Seven Kingdoms	The battle at Winterfell is approaching. Jaime...	2019-04-21	tt6027908	
69	8	3	The Long Night	The Night King and his army have arrived at Wi...	2019-04-28	tt6027912	

	seasonNum	episodeNum	episodeTitle	episodeDescription	episodeDate	tconst	averageRati
70	8	4	The Last of the Starks	In the wake of a costly victory, Jon and Daene...	2019-05-05	tt6027914	
71	8	5	The Bells	Daenerys and Cersei weigh their options as an ...	2019-05-12	tt6027916	
72	8	6	The Iron Throne	In the aftermath of the devastating attack on ...	2019-05-19	tt6027920	

73 rows × 8 columns

4.2 Just Kidding

- We did not spend a lot of time on MongoDB and that previous query was not fun.
- So, 4.1 is actually with 5 points and you are done with MongoDB. For now.

5 Neo4j

Instructions and Example

You must follow and comply with the instructions for completing the questions in this section. Any deviation from the format is a score of 0.

1. You will use the Movie Graph for this question.

2. Implement the answers in functions in the Python file

The example function returns a table with information about which people directed Tom hanks in which movies.

In [1]:
```
import question_5_neo4j
```

In [20]:
```
result = question_5_neo4j.directed_tom_hanks()
result
```

Out[20]:

	0	1	2
0	Tom Hanks	You've Got Mail	Nora Ephron
1	Tom Hanks	Sleepless in Seattle	Nora Ephron
2	Tom Hanks	Joe Versus the Volcano	John Patrick Stanley
3	Tom Hanks	That Thing You Do	Tom Hanks
4	Tom Hanks	Cloud Atlas	Tom Tykwer
5	Tom Hanks	Cloud Atlas	Lilly Wachowski

	0	1	2
6	Tom Hanks	Cloud Atlas	Lana Wachowski
7	Tom Hanks	The Da Vinci Code	Ron Howard
8	Tom Hanks	The Green Mile	Frank Darabont
9	Tom Hanks	Apollo 13	Ron Howard
10	Tom Hanks	Cast Away	Robert Zemeckis
11	Tom Hanks	Charlie Wilson's War	Mike Nichols
12	Tom Hanks	The Polar Express	Robert Zemeckis
13	Tom Hanks	A League of Their Own	Penny Marshall

5.1 People Who Directed Themseves (2 points)

Question

- Implement the function `people_who_directed_themselves.`
- The format of the answer is a data frame of the form `(name, title, name)` where the person `ACTED_IN` and `DRECTED` the movie.

Answer

In [2]:
```
#
# Test you answer
#
result = question_5_neo4j.directed_themselves()
result
```

Out[2]:

	0	1	2
0	Tom Hanks	That Thing You Do	Tom Hanks
1	Clint Eastwood	Unforgiven	Clint Eastwood
2	Danny DeVito	Hoffa	Danny DeVito

5.2 People Who Reviewed the same Movie (3 points)

Question

- Implement the function `both_reviewed(person_1_name, person_2_name)`
- The function returns a data frame of the form `person_1_name, movie_title, person_2_name` if the two people with the names rviewed the movie.
- Test you answer with the names below. You cannot hard code names in your query.

Answer

In [2]:
```
#
result = question_5_neo4j.both_reviewed('James Thompson', 'Jessica Thompson')
result
```

Out[2]:

	0	1	2
0	James Thompson	The Replacements	Jessica Thompson
1	James Thompson	The Da Vinci Code	Jessica Thompson

6 Data Modeling — RACI

Question

- RACI is an acronym for an approach to defining the relationships between people/stakeholders and a project.
- For this question, you will:
  - Do a Crow's Foot ER diagram defining a data model for representing RACI.
  - Create a SQL schema to represent the tables, constraints, etc. that you determine are necessary.
- The core entity types are:
  - `Project(project_id, project_name, start_date, end_date)`
  - `Person(UNI, last_name, first_name, email)`
- Implementing RACI is about understanding relationships between people and projects. The table below explains the concept.

Role	Description
Responsible	Who is responsible for doing the actual work for the project task.
Accountable	Who is accountable for the success of the task and is the decision-maker. Typically the project manager.*
Consulted	Who needs to be consulted for details and additional info on requirements. Typically the person (or team) to be consulted will be the subject matter expert.
Informed	Who needs to be kept informed of major updates. Typically senior leadership.

- There are two constraints:
  - There is exactly one person who is Accountable for a project.
  - A specific person can have at most one relationship to a project, for example "Bob" cannot be both Consulted and Informed for the same project.
- To answer this question, you must:
  - Draw the Crow's Foot ER diagram using LucidChart.
  - Create a database schema implementing the data model you define.
- You do not need to populate the data model with data or query the data, but YOU MUST execute your DDL statements.
- You execute the DDL statements implementing functions in the file `question_6_schema.`
- You may use DataGrip or other tools to design the schema and test your statements, but for the final answer you most have one function in `question_6_schema` for each DDL statement and you must execute each function in a cell below.

- Name your database schema `RACI`.
- There is no single, correct answer. Document any assumptions or design decisions that you make.

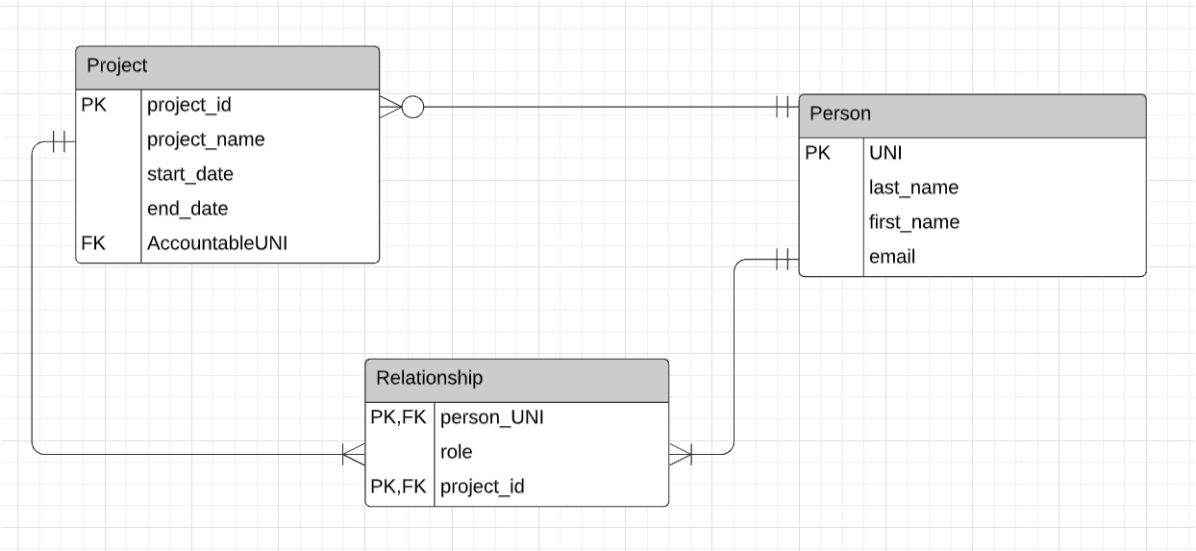
Design Decisions and Assumptions

Document any design decisions or assumptions that you make.

- One person can take charge of one or many projects. One project have four people.
- We assume UNI and project_id have data type of char(6) and char(8) since they have the uniform format like "yz3899" for UNI.
- Since a specific person can have at most one relationship to a project. I set triggers to constrain this in relationship table.
- Since the combination of projects and person's UNI is unique, we can set primary key.
- AccountableUNI should be unique, so I set trigger to constrain UNI is not duplicate.

ER Diagram

- Put your ER diagram here. You will receive instructions for how to submit on GradeScope.



Schema Creation

In [1]:
```
#
#
import question_6_schema
```

In [2]:
```
#
# Execute each function in a single cell.
#
res = question_6_schema.schema_operation_1()
res
```

Out[2]: 1

In [3]:

```
#
# Execute each function in a single cell.
#
res = question_6_schema.schema_operation_2()
res
```

Out[3]: 0

In [4]:
```
#
# Execute each function in a single cell.
#
res = question_6_schema.schema_operation_3()
res
```

Out[4]: 0

In [5]:
```
#
# Execute each function in a single cell.
#
res = question_6_schema.schema_operation_4()
res
```

Out[5]: 0

In [6]:
```
#
# Execute each function in a single cell.
#
res = question_6_schema.schema_operation_5()
res
```

Out[6]: 0

In [7]:
```
#
# Execute each function in a single cell.
#
res = question_6_schema.schema_operation_6()
res
```

Out[7]: 0

In [8]:
```
#
# Execute each function in a single cell.
#
res = question_6_schema.schema_operation_7()
res
```

Out[8]: 0

In [9]:
```
#
# Execute each function in a single cell.
#
res = question_6_schema.schema_operation_8()
res
```

Out[9]: 0

7. Data Transformation

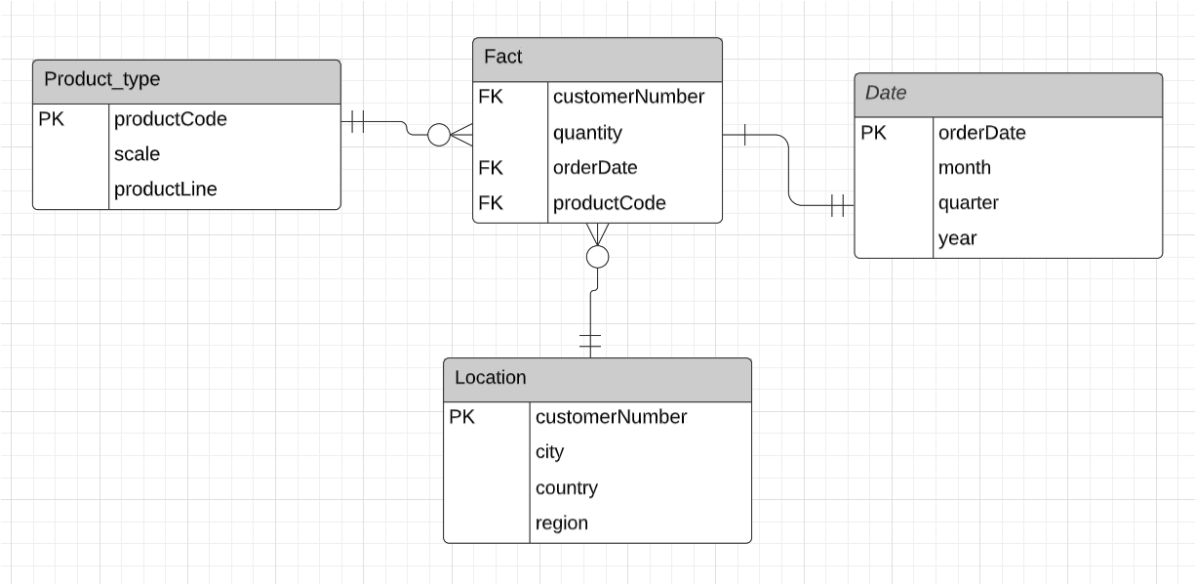
Question

- In this question, you will produce a star schema and populate with data from `classicmodels.`
- A star schema has a fact table and dimensions. The core fact is:
  - A customer (`customerNumber`)
  - Some quantity of a product (`quantityOrdered`) at at a price (`priceEach`)
  - On a given date (`orderDate`)
- We will consider three dimensions:
  - `date` is `(month, quarter, year)`
  - `location` is the dimension representing where the customer is and is of the form `(city, country, region).` Region is one of `(EMEA, NA, AP).`
    - `USA` and `Canada` are in `NA.`
    - `Philipines, Hong Kong, Singapore, Japan, Australia` and `New Zealand` are in `AP`
    - All other countries are in `EMEA.`
  - `product_type` is `(scale, product line).`
- You will follow the same approach for implementation as for question 6.
- There is an implementation template `question_7_sql.` You will implement three sets of SQL operations.
  - The functions of the form `schema_operation_n()` implement creating the star schema, tables, constraints, etc. There is one function for each statement. Name your schema `classicmodels_star`
  - The functions `data_transformation_n()` contain SQL statements for loading the `classicmodels_star` schema. You can have at most 3 SQL statement per function.
  - There are three queries you must implement:
    - `sales_by_year_region()` returns the total value of orders broken down by region and year.
    - `sales_by_quarter_year_county_region()` drills down to show the same information expanded to include quarter and year.
    - `sales_by_product_line_scale_year()` shows sales by product line, product scale and year.

Answer

In the following cells, execute your various functions that invoke SQL.

ER Diagram



Design Decisions and Assumptions

Document any design decisions or assumptions that you make.

- The new schema's data type is same as classicmodels in general.

In [1]:
```
#
#
import question_7_sql
```

In [2]:
```
#
# Execute each function in a single cell.
#
res = question_7_sql.schema_operation_1()
res
```

Out[2]: 1

In [3]:
```
res = question_7_sql.schema_operation_2()
res
```

Out[3]: 0

In [4]:
```
res = question_7_sql.schema_operation_3()
res
```

Out[4]: 0

In [5]:
```
res = question_7_sql.schema_operation_4()
res
```

Out[5]: 0

In [6]:
```
res = question_7_sql.schema_operation_5()
res
```

Out[6]: 0

In [7]:
```
res = question_7_sql.data_transformation_1()
res
```

Out[7]: 265

In [8]:
```
res = question_7_sql.data_transformation_2()
res
```

Out[8]: 2996

In [9]:
```
res = question_7_sql.data_transformation_3()
res
```

Out[9]: 67

In [10]:
```
res = question_7_sql.sales_by_year_region()
res
```

Out[10]:

	sales	region	year
0	1519511.84	EMEA	2003
1	2171244.36	EMEA	2004
2	1225638.04	NA	2003
3	1649903.68	NA	2004
4	572198.51	AP	2003
5	694757.47	AP	2004
6	829956.08	EMEA	2005
7	603650.19	NA	2005
8	337330.44	AP	2005

In [11]:
```
res = question_7_sql.sales_by_quarter_year_county_region()
res
```

Out[11]:

	sales	quarter	year	country	region
0	115479.71	2	2003	France	EMEA
1	55951.77	3	2004	France	EMEA
2	220409.99	4	2004	France	EMEA
3	136218.56	2	2003	USA	NA
4	369912.42	3	2004	USA	NA
...	...	...	...	...	...
117	27966.54	1	2005	Sweden	EMEA
118	42252.87	4	2003	Austria	EMEA

	sales	quarter	year	country	region
119	8807.12	1	2005	Austria	EMEA
120	7612.06	1	2004	Italy	EMEA
121	32077.44	3	2003	New Zealand	AP

122 rows × 5 columns

In [12]:
```
res = question_7_sql.sales_by_product_line_scale_year()
res
```

Out[12]:

	sales	productLine	productScale	year
0	461947.22	Vintage Cars	1:18	2003
1	127471.05	Vintage Cars	1:24	2003
2	146538.41	Classic Cars	1:10	2003
3	78911.11	Trucks and Buses	1:12	2003
4	149827.91	Trucks and Buses	1:18	2003
...	...	...	...	...
85	61905.07	Planes	1:700	2005
86	34950.65	Planes	1:72	2005
87	8014.35	Ships	1:72	2005
88	11663.53	Vintage Cars	1:32	2005
89	8576.51	Vintage Cars	1:50	2005

90 rows × 4 columns

In [ ]:
```
```