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 (https://www.mysqltutorial.org/mysql-sample-database.aspx) 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 f
# 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
# YOU MUST set the variables to the correct names for instance.
#
db_user = "admin"
db_password = "7Senses_kiki"
db_host = "tutorialdb.cbezzskgwcl3.us-east-2.rds.amazonaws.com"

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 exampp
#
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
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 [6]:
```
#
# Set the neo4j user and password for connecting to your database. The
# You set the password when you created the project and graph.
#
db_user = "neo4j"
db_password = "7Senses_kiki"

neo4j_check.set_neo4j_connect_info(db_user, db_password)
```

In [7]:

```
#
# You database MUST have the Movie DB installed. You had to do this fo
# the sample output.
#
res = neo4j_check.get_people_in_matrix()
res
```

Out[7]:

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

MongoDB

In [8]:

```
# Import the MongoDB test and helper functions.
#
import mongodb_check
```

In [9]:

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

In [11]:

```
#
# Run the following function. This will load information into MongoDB
#
df = mongodb_check.load_and_test_mongo()
df
```

Out[11]:

	_id	customerNumber	customerName	country
0	61b92940a52a45eb192b40ce	103	Atelier graphique	France
1	61b92940a52a45eb192b40d1	119	La Rochelle Gifts	France
2	61b92940a52a45eb192b40db	146	Saveley & Henriot, Co.	France
3	61b92940a52a45eb192b40e4	171	Daedalus Designs Imports	France
4	61b92940a52a45eb192b40e5	172	La Corne D'abondance, Co.	France
5	61b92940a52a45eb192b40f3	209	Mini Caravy	France
6	61b92940a52a45eb192b40fd	242	Alpha Cognac	France
7	61b92940a52a45eb192b4100	250	Lyon Souveniers	France
8	61b92940a52a45eb192b4101	256	Auto Associés & Cie.	France
9	61b92940a52a45eb192b411d	350	Marseille Mini Autos	France
10	61b92940a52a45eb192b411e	353	Reims Collectables	France
11	61b92940a52a45eb192b412c	406	Auto Canal+ Petit	France

1. Database Core Concepts (10 points)

- There is a Google Doc (https://docs.google.com/document/d/1b0VVAS_LC25iMjIBx9zP9UhDg5eqsVoQ6h6Wdb usp=sharing).
- 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

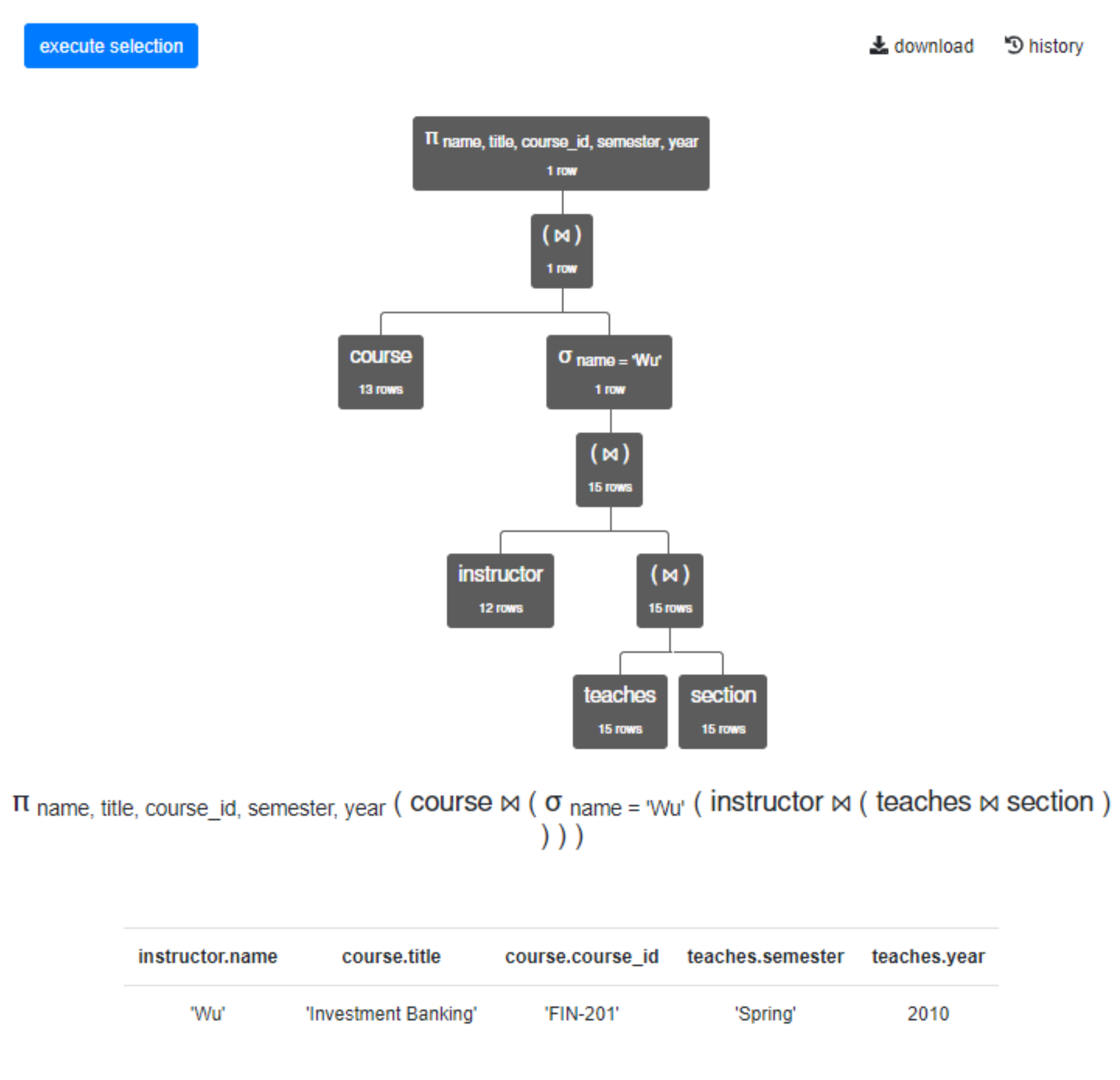
Question

- 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))
))
```



π $_{\text{name, title, course\_id, semester, year}}$ (course ⋈ (σ $_{\text{name = 'Wu'}}$ (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

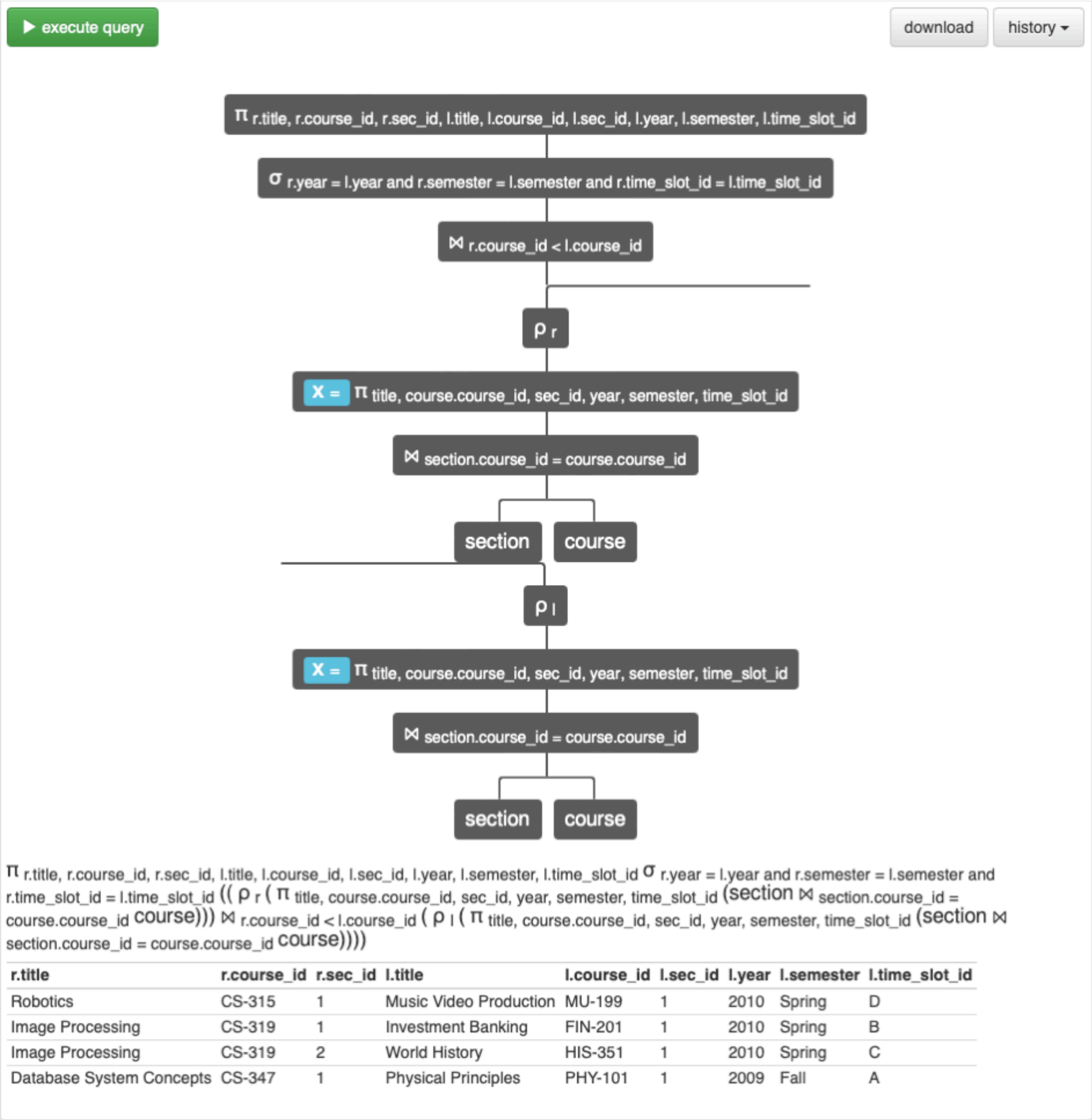
- Provide your answer following the examples' format.
- Use the RelaX calculator the "Silberschatz - UniversityDB" for this question.
- A section `r` overlaps with another section `l` if and only if: They occured at the same time (`year` , `semester` , `time_salot_id`).
- Produce the following table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

X = π title, course.course_id, sec_id, year, semester, time_slot_id (section ⋈ section.course_id = course.course_id course)

π r.title, r.course_id, r.sec_id, l.title, l.course_id, l.sec_id, l.year, l.semester, l.time_slot_id σ r.year = l.year and r.semester = l.semester and r.time_slot_id = l.time_slot_id ((ρ r X) ⋈ r.course_id < l.course_id (ρ l X))


▶ execute query
download
history ▾
π r.title, r.course_id, r.sec_id, l.title, l.course_id, l.sec_id, l.year, l.semester, l.time_slot_id
σ r.year = l.year and r.semester = l.semester and r.time_slot_id = l.time_slot_id
⋈ r.course_id < l.course_id
ρ r
X = π title, course.course_id, sec_id, year, semester, time_slot_id
⋈ section.course_id = course.course_id
section
course
ρ l
X = π title, course.course_id, sec_id, year, semester, time_slot_id
⋈ section.course_id = course.course_id
section
course

π r.title, r.course_id, r.sec_id, l.title, l.course_id, l.sec_id, l.year, l.semester, l.time_slot_id σ r.year = l.year and r.semester = l.semester and r.time_slot_id = l.time_slot_id ((ρ r (π title, course.course_id, sec_id, year, semester, time_slot_id (section ⋈ section.course_id = course.course_id course))) ⋈ r.course_id < l.course_id (ρ l (π title, course.course_id, sec_id, year, semester, time_slot_id (section ⋈ section.course_id = course.course_id 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
Robotics	CS-315	1	Music Video Production	MU-199	1	2010	Spring	D
Image Processing	CS-319	1	Investment Banking	FIN-201	1	2010	Spring	B
Image Processing	CS-319	2	World History	HIS-351	1	2010	Spring	C
Database System Concepts	CS-347	1	Physical Principles	PHY-101	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`.

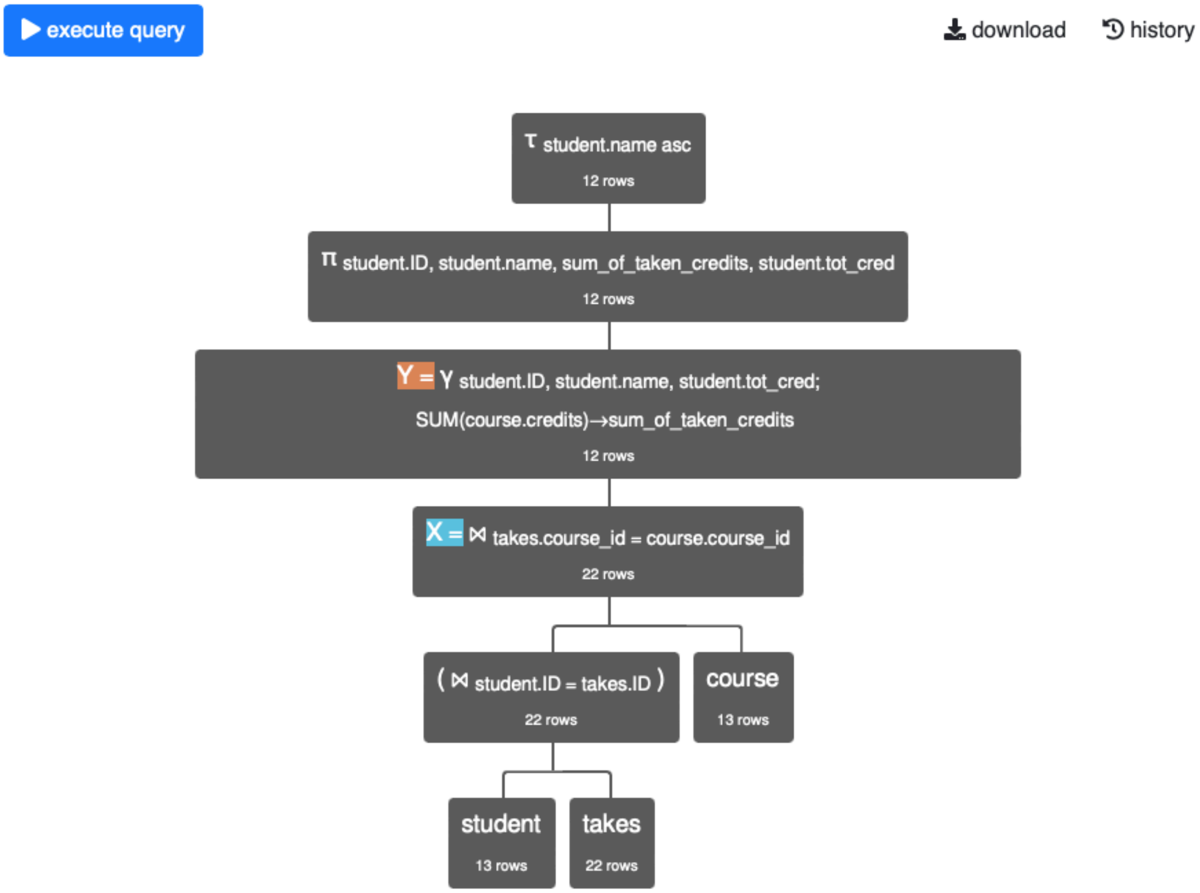
student_ID	student_name	sum_of_taken_credits	student.tot_cred
76653	'Aoi'	3	60
19991	'Brandt'	3	80
76543	'Brown'	7	58
23121	'Chavez'	3	110
44553	'Peltier'	4	56
55739	'Sanchez'	3	38
12345	'Shankar'	14	32
98988	'Tanaka'	8	120
54321	'Williams'	8	54
128	'Zhang'	7	102

Answer

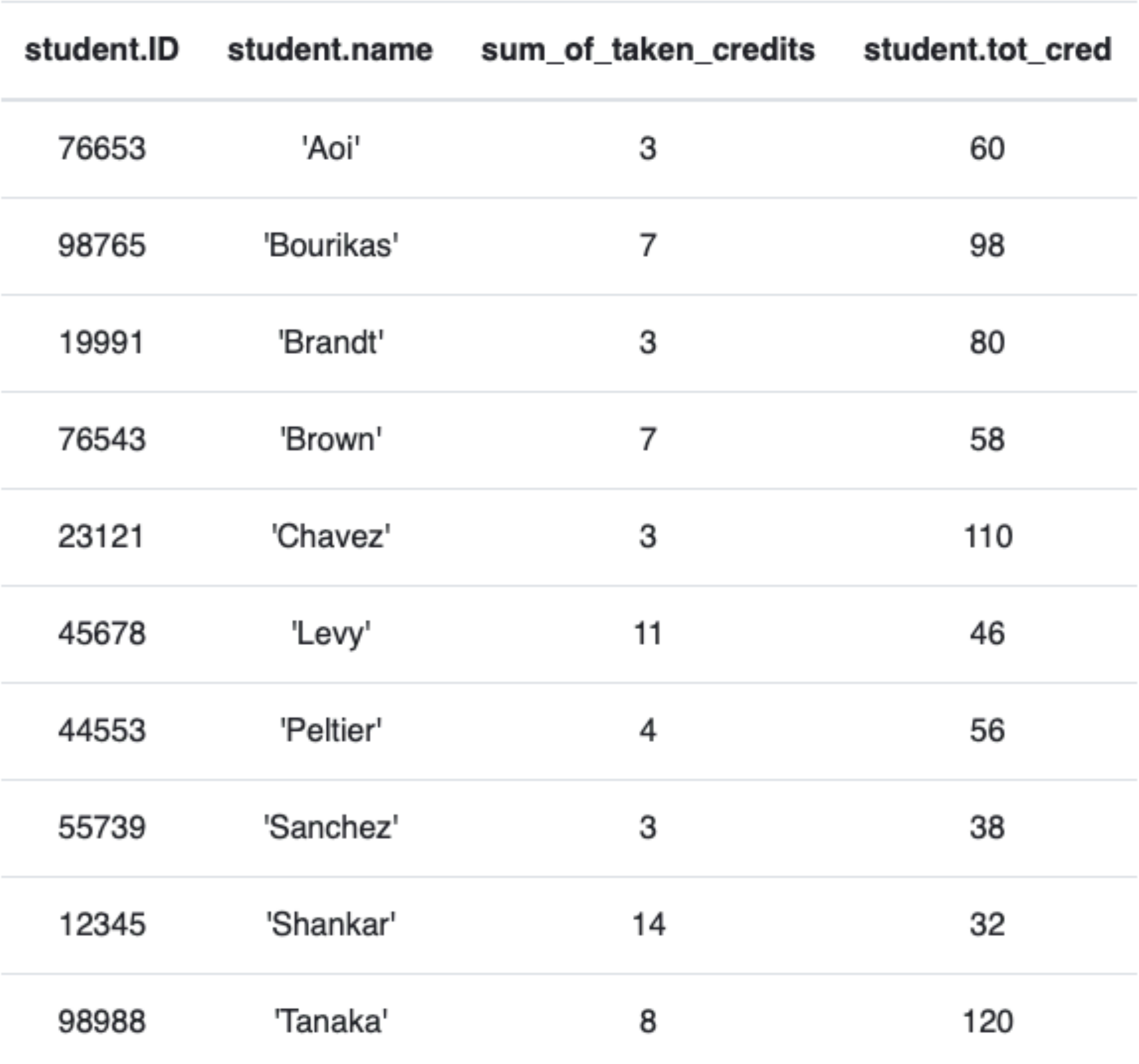
X = (student ⋈ student.ID = takes.ID takes) ⋈ takes.course_id = course.course_id course

Y = γ student.ID, student.name, student.tot_cred ; sum(course.credits)→sum_of_taken_credits (X)

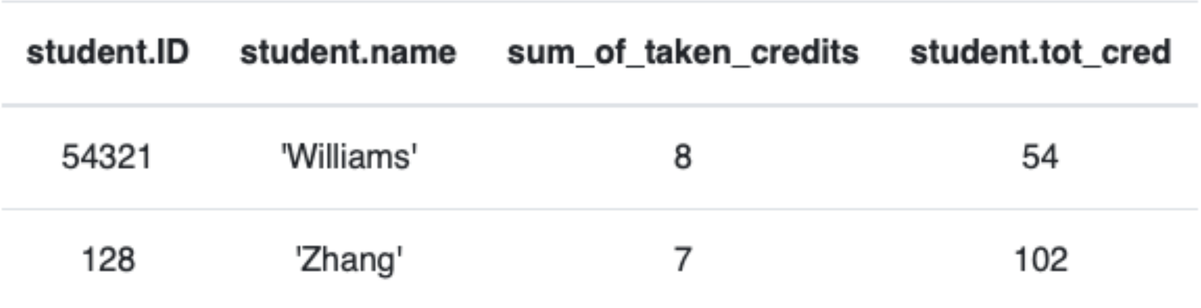
τ student.name (π student.ID, student.name, sum_of_taken_credits, student.tot_cred Y)


▶ execute query
download
history
τ student.name asc
12 rows
π student.ID, student.name, sum_of_taken_credits, student.tot_cred
12 rows
Y = γ student.ID, student.name, student.tot_cred;
SUM(course.credits)→sum_of_taken_credits
12 rows
X = ⋈ takes.course_id = course.course_id
22 rows
(⋈ student.ID = takes.ID)
22 rows
course
13 rows
student
13 rows
takes
22 rows

$\tau_{\text{student.name asc}}\ (\ \pi_{\text{student.ID, student.name, sum\_of\_taken\_credits, student.tot\_cred}}\ \gamma_{\text{student.ID, student.name, student.tot\_cred; SUM(course.credits)}\rightarrow\text{sum\_of\_taken\_credits}}\ (\ (\ \text{student} \bowtie_{\text{student.ID = takes.ID}} \text{takes}\ ) \bowtie_{\text{takes.course\_id = course.course\_id}} \text{course}\ )\ )$



student.ID	student.name	sum_of_taken_credits	student.tot_cred
76653	'Aoi'	3	60
98765	'Bourikas'	7	98
19991	'Brandt'	3	80
76543	'Brown'	7	58
23121	'Chavez'	3	110
45678	'Levy'	11	46
44553	'Peltier'	4	56
55739	'Sanchez'	3	38
12345	'Shankar'	14	32
98988	'Tanaka'	8	120



student.ID	student.name	sum_of_taken_credits	student.tot_cred
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.
2. 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 [2]:

```
#
# Call the function with the parameter France and display the resultin
#
result = question_3_sql.question_3_example_get_customers('France')
result
```

Out[2]:

	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]:

	country	revenue
0	USA	3032204.26
1	Germany	196470.99
2	Norway	270846.30
3	Spain	947470.01
4	Denmark	176791.44
5	Italy	360616.81
6	Philippines	87468.30
7	UK	391503.90
8	Sweden	120457.09
9	France	965750.58
10	Belgium	91471.03
11	Singapore	263997.78
12	Austria	161418.16
13	Australia	509385.82
14	New Zealand	416114.03
15	Finland	295149.35
16	Canada	205911.86
17	Hong Kong	45480.79
18	Japan	167909.95
19	Ireland	49898.27
20	Switzerland	108777.92

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 [4]:

```
#
# Execute this cell to display your answer.
#
result = question_3_sql.question_3_purchases_and_payments()
result
```

Out[4]:

	customerNumber	customerName	total_spent	total_payments	total_unpaid
0	242	Alpha Cognac	60483.36	60483.36	0.00
1	249	Amica Models & Co.	82223.23	82223.23	0.00
2	276	Anna's Decorations, Ltd	137034.22	137034.22	0.00
3	103	Atelier graphique	22314.36	22314.36	0.00
4	471	Australian Collectables, Ltd	55866.02	44920.76	10945.26
...	...	...	...	...	...
93	201	UK Collectables, Ltd.	106610.72	61167.18	45443.54
94	298	Vida Sport, Ltd	108777.92	108777.92	0.00
95	181	Vitachrome Inc.	72497.64	72497.64	0.00
96	144	Volvo Model Replicas, Co	66694.82	43680.65	23014.17
97	475	West Coast Collectables Co.	43748.72	43748.72	0.00

98 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 [5]:

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

Out[5]:

	customerNumber	customerName
0	103	Atelier graphique
1	112	Signal Gift Stores
2	125	Havel & Zbyszek Co
3	145	Danish Wholesale Imports
4	167	Herkku Gifts
...	...	...
67	480	Kremlin Collectables, Co.
68	481	Raanan Stores, Inc
69	486	Motor Mint Distributors Inc.
70	489	Double Decker Gift Stores, Ltd
71	496	Kelly's Gift Shop

72 rows × 2 columns

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 ```

2. 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 [2]:

```
result = question_4_mongo.question_4_example()
result
```

Out[2]:

	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 [3]:

```
#
# Run your test here.
#
result = question_4_mongo.question_4_ratings()
result
```

Out[3]:

	seasonNum	episodeNum	episodeTitle	episodeAirDate	episodeDescription	tconst	ave
0	1	1	Winter Is Coming	2011-04-17	Jon Arryn, the Hand of the King, is dead. King...	tt1480055	
1	1	2	The Kingsroad	2011-04-24	While Bran recovers from his fall, Ned takes o...	tt1668746	
2	1	3	Lord Snow	2011-05-01	Lord Stark and his daughters arrive at King's ...	tt1829962	
3	1	4	Cripples, Bastards, and Broken Things	2011-05-08	Eddard investigates Jon Arryn's murder. Jon be...	tt1829963	
4	1	5	The Wolf and the Lion	2011-05-15	Catelyn has captured Tyrion and plans to bring...	tt1829964	
...	...	...	...	...	...	...	
68	8	2	A Knight of the Seven Kingdoms	2019-04-21	The battle at Winterfell is approaching. Jaime...	tt6027908	
69	8	3	The Long Night	2019-04-28	The Night King and his army have arrived at Wi...	tt6027912	
70	8	4	The Last of the Starks	2019-05-05	In the wake of a costly victory, Jon and Daene...	tt6027914	
71	8	5	The Bells	2019-05-12	Daenerys and Cersei weigh their options as an ...	tt6027916	
72	8	6	The Iron Throne	2019-05-19	In the aftermath of the devastating attack on ...	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 [2]:
```
result = question_5_neo4j.directed_tom_hanks()
result
```

Out[2]:

	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
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 [3]:

```
#
# Test you answer
#
result = question_5_neo4j.directed_themselves()
result
```

Out[3]:

	name_1	title	name_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 [4]:

```
#
result = question_5_neo4j.both_reviewed('James Thompson', 'Jessica Tho
result
```

Out[4]:

	name_1	title	name_2
0	James Thompson	The Replacements	Jessica Thompson
1	James Thompson	The Da Vinci Code	Jessica Thompson

6 Data Modeling — RACI

Question

- RACI (https://www.softwareadvice.com/resources/what-is-a-raci-chart/) 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.

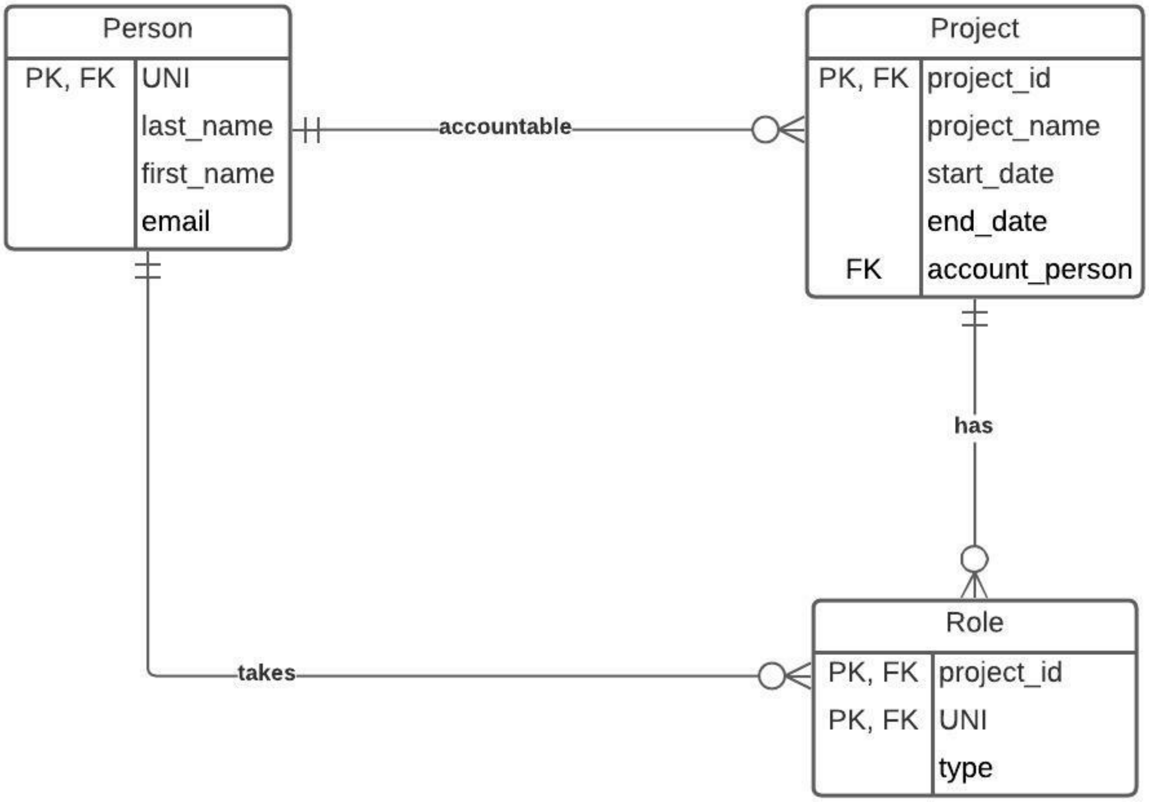
Design Decisions and Assumptions

Document any design decisions or assumptions that you make.

- The Person table should choose UNI as the primary key and the Project table should choose project_id as the primary key.
- I add a column called account_person in the Project table as a foreign key referencing the UNI column in the Person table. A Person can be accountable for many projects but a Project can have exactly one person to be accountable.
- I add a table Role with project_id and UNI as both primary key and foreign key referencing to the Person table and the Project table, and add a column type to represent the role of the person in the range of [Responsible, Consulted, Informed]. The uniqueness of primary key ensures that a project cannot have one person to take multiple roles. A role is associated with exactly one person and one project, but a person can have multiple roles in different projects and a project can have multiple roles assigned to different people.
- Also, to ensure that a person cannot take multiple roles including Accountable, I will set a trigger to abort the operation that tries to set the same person to take one of the roles as well as accountable of a project.

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
```

```
DDL statement:

    create table RACI.Person
    (
        UNI varchar(6) not null,
        last_name varchar(128) null,
        first_name varchar(128) null,
        email varchar(128) null,
        constraint Person_pk
            primary key (UNI)
    );
```

Out[2]:

```
0
```

In [3]:

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

```
DDL statement:

    create table RACI.Project
    (
        project_id varchar(128) not null,
        project_name varchar(128) null,
        start_date date null,
        end_date date null,
        account_person varchar(6) null,
        constraint Project_pk
            primary key (project_id),
        constraint Project_Person_UNI_fk
            foreign key (account_person) references Person (UNI)
    );
```

Out[3]:

```
0
```

In [4]:

```
#
# Execute each function in a single cell.
#
res = question_6_schema.schema_operation_3()
res
```

```
DDL statement:

    create table RACI.Role
    (
        project_id varchar(128) not null,
        UNI varchar(6) not null,
        type varchar(16) null,
        constraint Role_pk
            primary key (UNI, project_id),
        constraint Role_Person_UNI_fk
            foreign key (UNI) references Person (UNI),
        constraint Role_Project_project_id_fk
            foreign key (project_id) references Project (project_id)
    );
```

Out[4]:

```
0
```

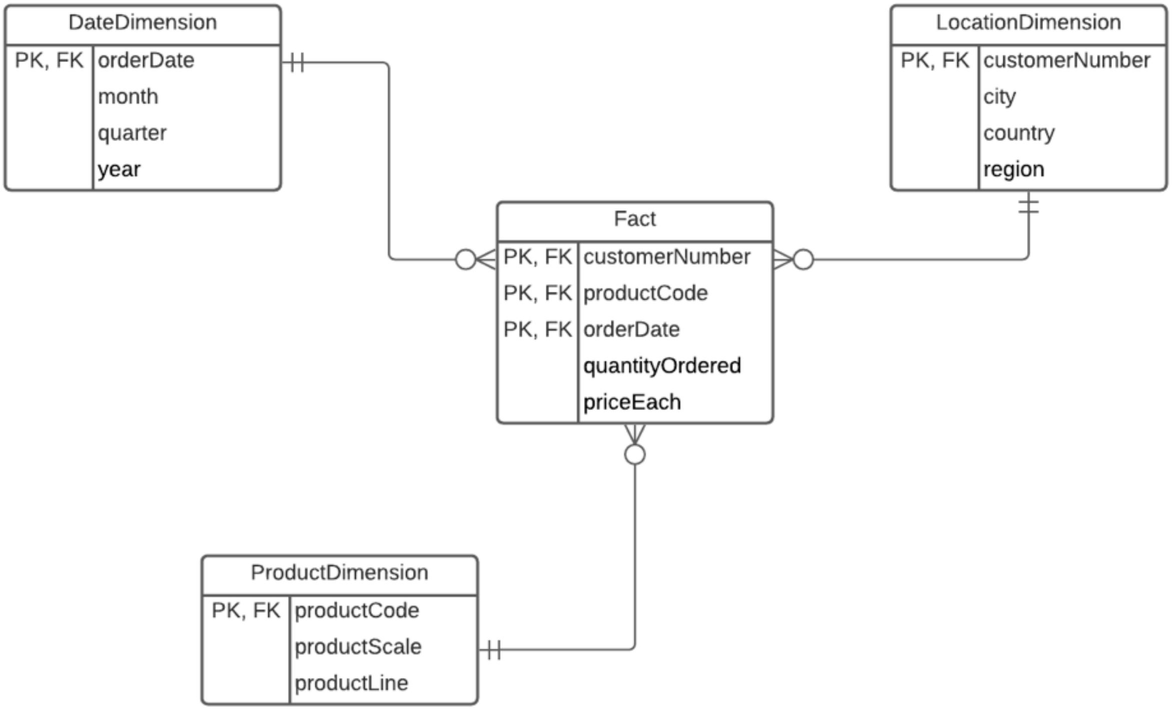
7. Data Transformation

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.



Design Decisions:
Since there is no primary key specified in the Fact table, so I cannot add foreign key constraints in the actual ddl statement. I just include the above ER diagram to show the foreign key.

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

In [2]:
```
res = question_7_sql.schema_operation_1()
res
```

```
DDL statement:

    create table classicmodels_star.date_dimension
    (
        orderDate date not null,
        month int null,
        quarter int null,
        year int null,
        constraint date_dimension_pk
            primary key (orderDate)
    );
```

Out[2]:
```
0
```

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

```
DDL statement:

    create table classicmodels_star.location_dimension
    (
        customerNumber int not null,
        city varchar(50) null,
        country varchar(50) null,
        region varchar(4) null,
        constraint location_dimension_pk
            primary key (customerNumber)
    );
```

Out[3]:
```
0
```

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

```
DDL statement:

    create table classicmodels_star.product_dimension
    (
        productCode varchar(15) not null,
        productScale varchar(10) null,
        productLine varchar(50) null,
        constraint product_dimension_pk
            primary key (productCode)
    );
```

Out[4]:
```
0
```

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

```
DDL statement:

    create table classicmodels_star.fact
    (
        customerNumber int not null,
        productCode varchar(15) not null,
        orderDate date not null,
        quantityOrdered int null,
        priceEach decimal(10,2) null,
        constraint fact_pk
            primary key (customerNumber, productCode, orderDate),
        constraint fact_location_dimension_fk
            foreign key (customerNumber) references location_dimensio
n (customerNumber),
        constraint fact_product_dimension_fk
            foreign key (productCode) references product_dimension (p
roductCode),
        constraint fact_date_dimension_fk
            foreign key (orderDate) references date_dimension (orderD
ate)
    );
```

Out[5]:
```
0
```

In [6]:
```
res = question_7_sql.data_transformation_1()
res
```

```
Data Transformation for table date_dimension:

    insert into classicmodels_star.date_dimension
    select distinct orderDate, month(orderDate),
            quarter(orderDate), year(orderDate)
    from classicmodels.orders


Data Transformation for table location_dimension:

    insert into classicmodels_star.location_dimension
    select customerNumber, city, country,
    case country
            when 'USA' then 'NA'
            when 'Canada' then 'NA'
            when 'Philippines' then 'AP'
            when 'Hong Kong' then 'AP'
            when 'Singapore' then 'AP'
            when 'Japan' then 'AP'
            when 'Australia' then 'AP'
            when 'New Zealand' then 'AP'
            else 'EMEA'
    end as region
    from classicmodels.customers
```

Out[6]:
```
(265, 122)
```

In [7]:

```
res = question_7_sql.data_transformation_2()
res
```

```
Data Transformation for table product_dimension:

    insert into classicmodels_star.product_dimension
    select productCode, productScale, productLine
    from classicmodels.products


Data Transformation for table fact:

    insert into classicmodels_star.fact
    select customers.customerNumber, products.productCode,
    orders.orderDate, orderdetails.quantityOrdered, orderdetails.pric
eEach
    from (((classicmodels.customers join classicmodels.orders
        on customers.customerNumber = orders.customerNumber)
        join classicmodels.orderdetails
        on orders.orderNumber = orderdetails.orderNumber)
        join classicmodels.products
        on orderdetails.productCode = products.productCode)
```

Out[7]:

```
(110, 2996)
```

In [8]:

```
res = question_7_sql.sales_by_year_region()
res
```

```
Query:

    select year, region, sum(quantityOrdered * priceEach) as order_to
tal_value
    from ((classicmodels_star.date_dimension join classicmodels_star.
fact
    on date_dimension.orderDate = fact.orderDate) join classicmodels_
star.location_dimension
    on fact.customerNumber = location_dimension.customerNumber)
    group by year, region
```

Out[8]:

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

In [9]:

```
res = question_7_sql.sales_by_quarter_year_county_region()
res
```

```
Query:

    select year, quarter, region, country, sum(quantityOrdered * pric
eEach) as order_total_value
    from ((classicmodels_star.date_dimension join classicmodels_star.
fact
    on date_dimension.orderDate = fact.orderDate) join classicmodels_
star.location_dimension
    on fact.customerNumber = location_dimension.customerNumber)
    group by year, quarter, region, country
```

Out[9]:

	year	quarter	region	country	order_total_value
0	2004	3	EMEA	France	55951.77
1	2003	2	EMEA	France	115479.71
2	2004	4	EMEA	France	220409.99
3	2004	3	NA	USA	369912.42
4	2004	4	NA	USA	662452.25
...	...	...	...	...	...
118	2005	1	EMEA	Sweden	27966.54
119	2003	4	EMEA	Austria	42252.87
120	2005	1	EMEA	Austria	8807.12
121	2004	1	EMEA	Italy	7612.06
122	2003	3	AP	New Zealand	32077.44

123 rows × 5 columns

In [10]:

```
res = question_7_sql.sales_by_product_line_scale_year()
res
```

```
Query:

    select productLine, productScale, year, sum(quantityOrdered * pri
ceEach) as order_total_value
    from ((classicmodels_star.date_dimension join classicmodels_star.
fact
    on date_dimension.orderDate = fact.orderDate) join classicmodels_
star.product_dimension
    on fact.productCode = product_dimension.productCode)
    group by productLine, productScale, year
```

Out[10]:

	productLine	productScale	year	order_total_value
0	Motorcycles	1:10	2004	179505.19
1	Motorcycles	1:10	2005	76366.86
2	Motorcycles	1:10	2003	114970.54
3	Classic Cars	1:10	2004	210238.23
4	Classic Cars	1:10	2003	146538.41
...	...	...	...	...
85	Planes	1:700	2004	169632.37
86	Planes	1:700	2005	61905.07
87	Ships	1:72	2004	22806.48
88	Ships	1:72	2003	16729.57
89	Ships	1:72	2005	8014.35

90 rows × 4 columns

In [ ]: