COMS W 4111-002
W4111 - Introduction to Databases
Section 003/V03, Fall 2022
Take Home Final

Exam Instructions

- We will publish instructions on Ed.

Environment Setup and Test

MySQL

- Replace `root` and `dbuserdbuser` for the correct values for you MySQL instance from previous homework assignments and exams.
- You will need the sample database that comes with the recommended textbook to execute the setup test.
  - You should have already installed the database because you need for previous assignments.
  - I named my database

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

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

In [3]:
```
%sql select * from COMS4111.student
```

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

Out[3]:

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

Neo4j

- Please set the values for your Neo4j database below.

- Make sure that your database is active. If you have not used it for a while, you need to log in through the website and restart the database.

In [4]:
```
neo4j_url = "neo4j+s://d4736a05.databases.neo4j.io"
neo4j_user = "neo4j"
neo4j_password = "KCvzV_oo2gWXDp1AfdRKBqThznmEwY5McMesMrQ6y3c"
```

In [5]:
```
from py2neo import Graph
```

In [6]:
```
def t1():
    graph = Graph(neo4j_url, auth=(neo4j_user, neo4j_password))
    q = "match (r:Person) where r.name='Tom Hanks' return r"
    res = graph.run(q)

    for r in res:
        print(r)
```

- Please rerun the following cell.

In [7]:
```
t1()
```

```
Node('Person', born=1956, name='Tom Hanks')
```

MongoDB

- Please set your URL for MongoDB Atlas and make sure that your cluster is not suspended.

In [8]:
```
mongodb_url = "mongodb+srv://lily:dbuserdbuser@cluster0.q3j5jvq.mongodb.net/?retryWrites=true&w=majority
```

In [9]:
```
import pymongo
```

In [10]:
```
def connect():
    client = pymongo.MongoClient(
        mongodb_url
    )
    return client


def t_connect():
    c = connect()
    print("Databases = ", list(c.list_databases()))
```

In [11]:
```
#
# Note, you list of local databases will be different. The values do not matter.
#
t_connect()
```

```
Databases =  [{'name': 'hw4', 'sizeOnDisk': 405504, 'empty': False}, {'name': 'testdb', 'sizeOnDisk': 73
728, 'empty': False}, {'name': 'admin', 'sizeOnDisk': 344064, 'empty': False}, {'name': 'local', 'sizeOn
Disk': 6464942080, 'empty': False}]
```

Written Questions — General Knowledge

- The written questions require a short, succinct answer.
- Remember, "If you can't explain it simply, you don't understand it well enough."
- Some questions will research using the web, lecture slides, etc. You cannot cut and paste from sources. Your answer must show that you read the material and understand the concept.

- If you use a source other than lecture material, please provide a URL to the source(s) you read.

G1

Question: List at least two reasons why database systems support data manipulation using a declarative query language such as SQL, instead of just providing a library of C or C + + functions to carry out data manipulation.

Answer:

1. Declarative query language lets us define what data we want to compute and lets the engine underneath take care of seamlessly retrieving it
2. Declarative language is typically easy to learn and understand, such as SELECT, INSERT INTO, UPDATE... the syntactical rules are not complex in SQL, which makes it a user-friendly language
3. Using a declarative query language may provide better code than can be produced manually

G2

Question: List four significant differences between:

- Processing data by writing programs that manipulate files.
- Using a database management system and query language.

Answer:

1. The file system has a higher rate of data inconsistency, while a database management system has relative low rate of data inconsistency
2. Redundant data is present in the file system, while in contrast there is no presence of redundant data in database management system
3. File systems do not handle complex transactions, whereas SQL makes it simple to build complex transactions in DBMS systems.
4. The file system has less security, but database management system supports more security mechanisms

G3

Question: List five responsibilities (functionality provided) of a database-management system. For each responsibility, explain the potential problems that would occur without the functionality.

Answer:

1. Concurrency: Even with effective integrity enforcement in each transaction, consistency restrictions may still be broken
2. Security: Unauthorized users may access the portion of database which they do not have permission
3. Backup and recovery: Data could be lost permanently and not able to recover to the state that existed prior to a failure
4. Integrity: constraints on consistency might not be met, e.g. account balances might fall below the minimum permitted
5. Interaction with the File Manager: without file manager interaction, nothing stored in the files can be retrieved

G4

Question: We all use SSOL to choose and register for classes. Another option would be to have a single Google sheet (shared spreadsheet) that we all use to register for classes. What are problems with using a shared spreadsheet?

Answer:

1. For shared spreadsheet, not multiple(too many) users can access and edit it simultaneously. When too many users are editing a shared spreadsheet simultaneously, the effective is decrease and likelihood of errors is increase.
2. Because we are editing a very large shared spreadsheet, a single Google sheet cannot prevent data redundancy.
3. A shared spreadsheet cannot store information as effectively as DBMS
4. A shared spreadshee has record limitation
5. It is not secure for us to use a single Google sheet to register for classes, because anyone who can access this spreadsheet can see personal information of other users of the spreadsheet

G5

Question: NoSQL databases have become increasingly popular for supporting applications. List 3 benefits of or reasons for using NoSQL databases versus SQL/relational databases. List 3 benefits of relational databases versus NoSQL databases.

Answer:

- **using NoSQL databases versus SQL/relational databases**: (1)NoSQL can work with large amounts of unstructured or semi-structured data that doesn't fit the relational model (2)NoSQL allow us to scale-out horizontally, which means we can add cheaper commodity servers whenever we need to (3)NoSQL support flexibility of a dynamic schema or want more choice over the data model
- **using relational databases versus NoSQL databases**: (1)SQL support transaction-oriented systems such as accounting or financial applications (2)SQL enable a high degree of data integrity and security (3)SQL can perform complex queries

Relational Model

R1

Question: A column in a relation (table) has a *type*. Consider implementing a `date` as `CHAR(10)` in the format `YYYY-MM-DD.` The lecture material states that attributes (column values) come from a *domain*. Using `date` explain the differenc between a *domain* and a *type*.

Answer:

`Domain` is a set of acceptable values that a column is allowed to contain. This is based on various properties and the data type for the column. `Domain` of `date` is character data in a length of 10 and in the format of YYYY-MM-DD. However data `type` determines what type of values can be stored in a column. `Type` of `date` is character data in a length of 10.

R2

Question: The domain for a relation (table) attribute (column) should be *atomic*. Why?

Answer:

When the domain for a relation attribute is atomic, columns of relational tables aren't repeating groups or arrays. The atomic value property's primary advantage is that it makes data processing logic simpler.

R3

Question: "In the US Postal System, a delivery point is a specific set of digits between 00 and 99 assigned to every address. When combined with the ZIP + 4 code, the delivery point provides a unique identifier for every deliverable address served by the United States Postal Service."

The lecture 2 slides provide a notation for representing a relation's schema. Assume we want to define a relation for US mailing addresses, and that the columns are:

- Zip code
- +4 code
- delivery_point
- address_line_1
- address_line_2
- city
- state

Use the notation to define the schema for an address. A simple example of an address's column values might be:

- Zip code: 10027
- +4 code: 6623
- delivery_point: 99
- address_line_1: 520 W 120th St
- address_line_2: Room 402

- city: New York
- state: NY

Answer:

ADDRESSES(<u>Zip_code</u>, <u>+4_code</u>, delivery_point, address_line_1, address_line_2, city, state)

R4

Note: Use the RelaX calculator and the schema associated with the recommended textbook to answer this question. Your answer should contain:

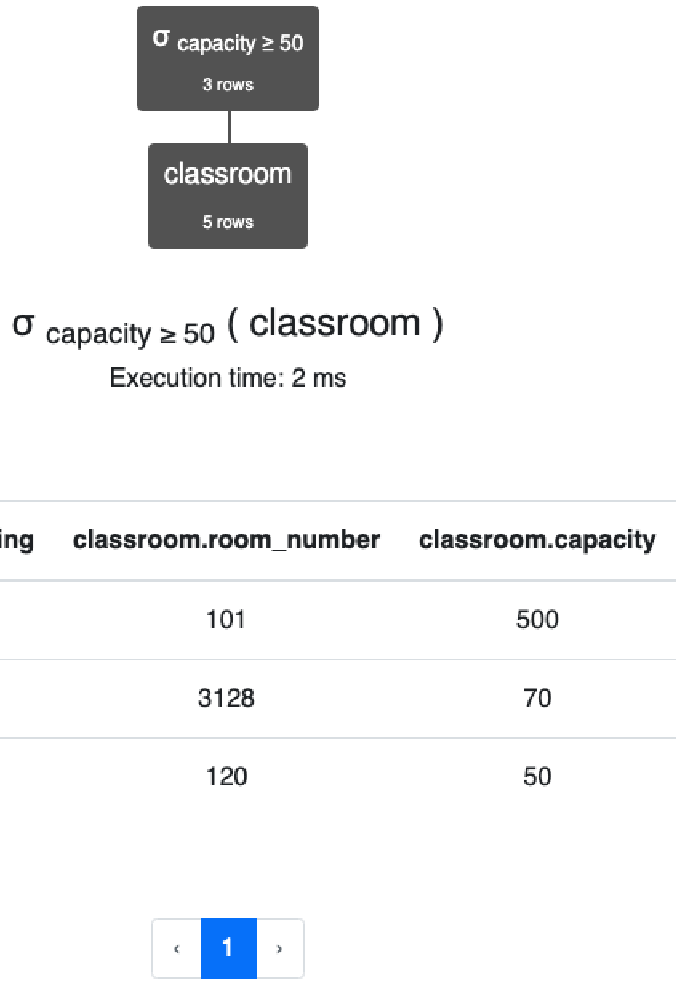
- The text for the query.
- An image showing the query execution and result.

An example of the format is:

Query

`σ capacity >= 50 (classroom)`

Execution



$\sigma_{\text{capacity} \geq 50}$ (classroom)

Execution time: 2 ms

classroom.building	classroom.room_number	classroom.capacity
'Packard'	101	500
'Taylor'	3128	70
'Watson'	120	50

Question: Translate the following SQL statement into an equivalent relational algebra statement.

```
select
    *
from
    instructor
```

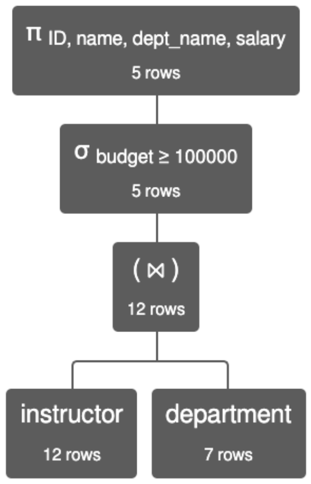
```
where
    dept_name in (select dept_name from department where budget >= 100000)
```

Answer:

Query

`π ID, name, dept_name, salary (σ budget>= 100000 (instructor ⋈ department))`

Execution



$\pi_{\text{ID, name, dept\_name, salary}} (\sigma_{\text{budget} \geq 100000} (\text{instructor} \bowtie \text{department}))$

Execution time: 2 ms

instructor.ID	instructor.name	instructor.dept_name	instructor.salary
10101	'Srinivasan'	'Comp. Sci.'	65000
12121	'Wu'	'Finance'	90000
45565	'Katz'	'Comp. Sci.'	75000
76543	'Singh'	'Finance'	80000
83821	'Brandt'	'Comp. Sci.'	92000

R5

Use the same format to answer this question.

Question

Use the following query to compute a new table.

```
section_and_time = π course_id, sec_id, semester, year, day, start_hr, start_min, end_hr,
end_min (section ⋈ time_slot)
```

Using only section_and_time, write a relational algebra expression that returns a relation of overlapping courses of the form

`(course_id_1, sec_id_1, semester_1, year_1, course_id_2, sec_id_2, semester_2, year_2)`.

Your table cannot container duplicates. For example, a result containing

```
(BIO-101, 1, fall, 2022, MATH-101, 2, fall, 2022)
(MATH-101, 2, fall, 2022, BIO-101, 1, fall, 2022)
```

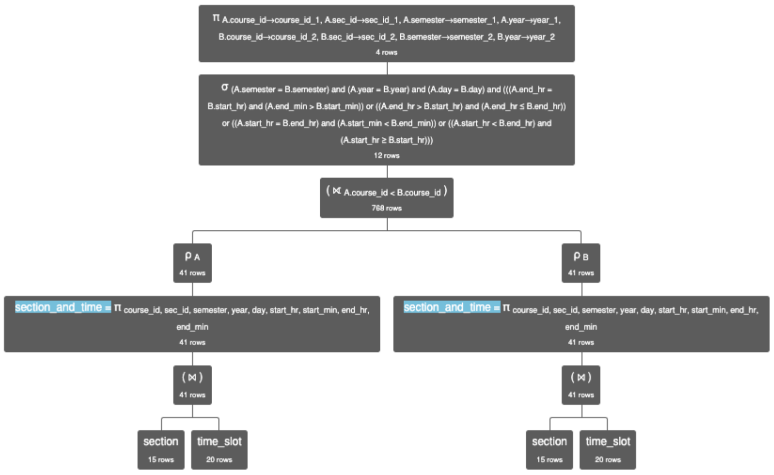
is incorrect.

Answer:

Query

```
π A.course_id -> course_id_1, A.sec_id -> sec_id_1, A.semester -> semester_1, A.year ->
year_1,
    B.course_id -> course_id_2, B.sec_id -> sec_id_2, B.semester -> semester_2, B.year ->
year_2
( σ (A.semester = B.semester) ∧ (A.year = B.year) ∧ (A.day = B.day) ∧
    (((A.end_hr = B.start_hr) ∧ (A.end_min > B.start_min)) ∨
        ((A.end_hr > B.start_hr) ∧ (A.end_hr ≤ B.end_hr)) ∨
        ((A.start_hr = B.end_hr) ∧ (A.start_min < B.end_min)) ∨
        ((A.start_hr < B.end_hr) ∧ (A.start_hr ≥ B.start_hr)))
((ρA(section_and_time)) ⋉ A.course_id < B.course_id (ρB(section_and_time))))
```

Execution


π A.course_id→course_id_1, A.sec_id→sec_id_1, A.semester→semester_1, A.year→year_1, B.course_id→course_id_2, B.sec_id→sec_id_2, B.semester→semester_2, B.year→year_2
4 rows
σ (A.semester = B.semester) and (A.year = B.year) and (A.day = B.day) and (((A.end_hr = B.start_hr) and (A.end_min > B.start_min)) or ((A.end_hr > B.start_hr) and (A.end_hr ≤ B.end_hr)) or ((A.start_hr = B.end_hr) and (A.start_min < B.end_min)) or ((A.start_hr < B.end_hr) and (A.start_hr ≥ B.start_hr)))
12 rows
(⋉ A.course_id < B.course_id)
768 rows
ρ A
41 rows
ρ B
41 rows
section_and_time = π course_id, sec_id, semester, year, day, start_hr, start_min, end_hr, end_min
41 rows
section_and_time = π course_id, sec_id, semester, year, day, start_hr, start_min, end_hr, end_min
41 rows
(⋈)
41 rows
(⋈)
41 rows
section
15 rows
time_slot
20 rows
section
15 rows
time_slot
20 rows

π A.course_id→course_id_1, A.sec_id→sec_id_1, A.semester→semester_1, A.year→year_1, B.course_id→course_id_2, B.sec_id→sec_id_2, B.semester→semester_2, B.year→year_2 (σ (A.semester = B.semester) and (A.year = B.year) and (A.day = B.day) and (((A.end_hr = B.start_hr) and (A.end_min > B.start_min)) or ((A.end_hr > B.start_hr) and (A.end_hr ≤ B.end_hr)) or ((A.start_hr = B.end_hr) and (A.start_min < B.end_min)) or ((A.start_hr < B.end_hr) and (A.start_hr ≥ B.start_hr))) ((ρ A π course_id, sec_id, semester, year, day, start_hr, start_min, end_hr, end_min (section ⋈ time_slot)) ⋉ A.course_id < B.course_id (ρ B π course_id, sec_id, semester, year, day, start_hr, start_min, end_hr, end_min (section ⋈ time_slot))))

Execution time: 4 ms

course_id_1	sec_id_1	semester_1	year_1	course_id_2	sec_id_2	semester_2	year_2
'CS-319'	1	'Spring'	2010	'FIN-201'	1	'Spring'	2010
'CS-319'	2	'Spring'	2010	'HIS-351'	1	'Spring'	2010
'CS-315'	1	'Spring'	2010	'MU-199'	1	'Spring'	2010
'CS-347'	1	'Fall'	2009	'PHY-101'	1	'Fall'	2009

‹ 1 ›

SQL

- You will use the Classic Models tutorial database, which you should have already loaded into MySQL.

S1

Question: Create a view `employee_customer_sales` with the following information:

- `employeeNumber`

- `employeeLastname`
- `employeeFirstName`
- `customerNumber`
- `customerName`
- `revenue`

- The employee information is for the employee that is the `customer.customerRepEmployeeNumber`.
- `revenue` is the total revenue over all of the customer's orders.
  - The revenue for an `order` is `priceEach*quantityOrdered` for each `orderdetails` in the order.

Answer:

In [12]:
```
%%sql

USE classicmodels;

CREATE OR REPLACE VIEW employee_customer_sales AS
    WITH order_revenue AS (
        SELECT customerNumber,
               orders.orderNumber,
               SUM(quantityOrdered * priceEach) AS order_revenue
        FROM orderdetails
        INNER JOIN orders
            ON orderdetails.orderNumber = orders.orderNumber
        GROUP BY orderNumber
    )
    SELECT employeeNumber AS employeeNumber,
           lastName AS employeeLastname,
           firstName AS employeeFirstName,
           order_revenue.customerNumber AS customerNumber,
           customerName AS customerName,
           SUM(order_revenue) AS revenue
    FROM order_revenue
    INNER JOIN customers
        ON order_revenue.customerNumber = customers.customerNumber
    INNER JOIN employees
        ON employees.employeeNumber = customers.salesRepEmployeeNumber
    GROUP BY customerNumber
    ORDER BY employeeLastname, revenue DESC;
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
0 rows affected.
```
Out[12]:
```
[]
```

Test Answer:

In [13]:
```
%sql select * from employee_customer_sales;
```

```
 * mysql+pymysql://root:***@localhost
98 rows affected.
```
Out[13]:

employeeNumber	employeeLastname	employeeFirstName	customerNumber	customerName	revenue
1337	Bondur	Loui	146	Saveley & Henriot, Co.	130305.35
1337	Bondur	Loui	353	Reims Collectables	126983.19
1337	Bondur	Loui	172	La Corne D'abondance, Co.	86553.52
1337	Bondur	Loui	406	Auto Canal+ Petit	86436.97
1337	Bondur	Loui	350	Marseille Mini Autos	71547.53
1337	Bondur	Loui	250	Lyon Souveniers	67659.19
1501	Bott	Larry	187	AV Stores, Co.	148410.09
1501	Bott	Larry	201	UK Collectables, Ltd.	106610.72
1501	Bott	Larry	334	Suominen Souveniers	103896.74
1501	Bott	Larry	311	Oulu Toy Supplies, Inc.	95706.15

1501	Bott	Larry	186	Toys of Finland, Co.	95546.46
1501	Bott	Larry	324	Stylish Desk Decors, Co.	80556.73
1501	Bott	Larry	240	giftsbymail.co.uk	71783.75
1501	Bott	Larry	489	Double Decker Gift Stores, Ltd	29586.15
1401	Castillo	Pamela	382	Salzburg Collectables	137480.07
1401	Castillo	Pamela	145	Danish Wholesale Imports	129085.12
1401	Castillo	Pamela	278	Rovelli Gifts	127529.69
1401	Castillo	Pamela	386	L'ordine Souveniers	125505.57
1401	Castillo	Pamela	227	Heintze Collectables	89909.80
1401	Castillo	Pamela	249	Amica Models & Co.	82223.23
1401	Castillo	Pamela	314	Petit Auto	70851.58
1401	Castillo	Pamela	452	Mini Auto Werke	51059.99
1401	Castillo	Pamela	381	Royale Belge	29217.18
1401	Castillo	Pamela	473	Frau da Collezione	25358.32
1188	Firrelli	Julie	320	Mini Creations Ltd.	101872.52
1188	Firrelli	Julie	379	Collectables For Less Inc.	73533.65
1188	Firrelli	Julie	495	Diecast Collectables	65541.74
1188	Firrelli	Julie	339	Classic Gift Ideas, Inc	57939.34
1188	Firrelli	Julie	204	Online Mini Collectables	55577.26
1188	Firrelli	Julie	173	Cambridge Collectables Co.	32198.69
1611	Fixter	Andy	114	Australian Collectors, Co.	180585.07
1611	Fixter	Andy	276	Anna's Decorations, Ltd	137034.22
1611	Fixter	Andy	282	Souveniers And Things Co.	133907.12
1611	Fixter	Andy	471	Australian Collectables, Ltd	55866.02
1611	Fixter	Andy	333	Australian Gift Network, Co	55190.16
1702	Gerard	Martin	458	Corrida Auto Replicas, Ltd	112440.09
1702	Gerard	Martin	298	Vida Sport, Ltd	108777.92
1702	Gerard	Martin	216	Enaco Distributors	68520.47
1702	Gerard	Martin	484	Iberia Gift Imports, Corp.	50987.85
1702	Gerard	Martin	344	CAF Imports	46751.14
1370	Hernandez	Gerard	141	Euro+ Shopping Channel	820689.54
1370	Hernandez	Gerard	119	La Rochelle Gifts	158573.12
1370	Hernandez	Gerard	209	Mini Caravy	75859.32
1370	Hernandez	Gerard	171	Daedalus Designs Imports	61781.70
1370	Hernandez	Gerard	242	Alpha Cognac	60483.36
1370	Hernandez	Gerard	256	Auto Associés & Cie.	58876.41
1370	Hernandez	Gerard	103	Atelier graphique	22314.36
1165	Jennings	Leslie	124	Mini Gifts Distributors Ltd.	591827.34
1165	Jennings	Leslie	450	The Sharp Gifts Warehouse	143536.27
1165	Jennings	Leslie	321	Corporate Gift Ideas Co.	132340.78
1165	Jennings	Leslie	161	Technics Stores Inc.	104545.22
1165	Jennings	Leslie	129	Mini Wheels Co.	66710.56
1165	Jennings	Leslie	487	Signal Collectibles Ltd.	42570.37
1504	Jones	Barry	448	Scandinavian Gift Ideas	120943.53
1504	Jones	Barry	121	Baane Mini Imports	104224.79

1504	Jones	Barry	167	Herkku Gifts	97562.47
1504	Jones	Barry	259	Toms Spezialitäten, Ltd	89223.14
1504	Jones	Barry	128	Blauer See Auto, Co.	75937.76
1504	Jones	Barry	299	Norway Gifts By Mail, Co.	69059.04
1504	Jones	Barry	144	Volvo Model Replicas, Co	66694.82
1504	Jones	Barry	189	Clover Collections, Co.	49898.27
1504	Jones	Barry	415	Bavarian Collectables Imports, Co.	31310.09
1612	Marsh	Peter	323	Down Under Souveniers, Inc	154622.08
1612	Marsh	Peter	496	Kelly's Gift Shop	137460.79
1612	Marsh	Peter	166	Handji Gifts& Co	107746.75
1612	Marsh	Peter	357	GiftsForHim.com	94431.76
1612	Marsh	Peter	412	Extreme Desk Decorations, Ltd	90332.38
1621	Nishi	Mami	148	Dragon Souveniers, Ltd.	156251.03
1621	Nishi	Mami	398	Tokyo Collectables, Ltd	105548.73
1621	Nishi	Mami	385	Cruz & Sons Co.	87468.30
1621	Nishi	Mami	177	Osaka Souveniers Co.	62361.22
1621	Nishi	Mami	211	King Kong Collectables, Co.	45480.79
1216	Patterson	Steve	363	Online Diecast Creations Co.	116449.29
1216	Patterson	Steve	157	Diecast Classics Inc.	104358.69
1216	Patterson	Steve	286	Marta's Replicas Co.	90545.37
1216	Patterson	Steve	462	FunGiftIdeas.com	88627.49
1216	Patterson	Steve	362	Gifts4AllAges.com	84340.32
1216	Patterson	Steve	198	Auto-Moto Classics Inc.	21554.26
1166	Thompson	Leslie	205	Toys4GrownUps.com	93803.30
1166	Thompson	Leslie	239	Collectable Mini Designs Co.	80375.24
1166	Thompson	Leslie	112	Signal Gift Stores	80180.98
1166	Thompson	Leslie	475	West Coast Collectables Co.	43748.72
1166	Thompson	Leslie	347	Men 'R' US Retailers, Ltd.	41506.19
1166	Thompson	Leslie	219	Boards & Toys Co.	7918.60
1286	Tseng	Foon Yue	151	Muscle Machine Inc	177913.95
1286	Tseng	Foon Yue	181	Vitachrome Inc.	72497.64
1286	Tseng	Foon Yue	455	Super Scale Inc.	70378.65
1286	Tseng	Foon Yue	424	Classic Legends Inc.	69214.33
1286	Tseng	Foon Yue	233	Québec Home Shopping Network	68977.67
1286	Tseng	Foon Yue	456	Microscale Inc.	29230.43
1323	Vanauf	George	131	Land of Toys Inc.	149085.15
1323	Vanauf	George	175	Gift Depot Inc.	95424.63
1323	Vanauf	George	328	Tekni Collectables Inc.	81806.55
1323	Vanauf	George	319	Mini Classics	78432.16
1323	Vanauf	George	486	Motor Mint Distributors Inc.	77726.59
1323	Vanauf	George	202	Canadian Gift Exchange Network	70122.19
1323	Vanauf	George	260	Royal Canadian Collectables, Ltd.	66812.00
1323	Vanauf	George	447	Gift Ideas Corp.	49967.78

S2

Question:

- Below, there is a query that creates a view. Run the query.
- Using the view, write a query that produces a table of the form `(productCode, productName)` for products that no customer in Asia has ordered.
- For this questions purposes, the Asian countries are:
  - Japan
  - Singapore
  - Philipines
  - Hong King
- You must not use a JOIN.

In [14]:
```
#
# Create the view
#
%sql create or replace view orders_all as \
    select * from orders join orderdetails using(orderNumber)
```

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

Out[14]:
```
[]
```

Answer:

In [15]:
```
%%sql

WITH asian_code AS (
    SELECT productCode
    FROM orders_all
    WHERE customerNumber IN (
        SELECT customerNumber
        FROM customers
        WHERE country = 'Japan' OR
              country = 'Singapore' OR
              country = 'Philippines' OR
              country = 'Hong Kong'
        )
)

SELECT productCode,
       productName
FROM products
WHERE productCode NOT IN (SELECT productCode FROM asian_code)
```

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

Out[15]:

productCode	productName
S10_1678	1969 Harley Davidson Ultimate Chopper
S10_4757	1972 Alfa Romeo GTA
S12_2823	2002 Suzuki XREO
S18_1342	1937 Lincoln Berline
S18_1367	1936 Mercedes-Benz 500K Special Roadster
S18_2795	1928 Mercedes-Benz SSK
S18_2870	1999 Indy 500 Monte Carlo SS
S18_3029	1999 Yamaha Speed Boat
S18_3233	1985 Toyota Supra
S18_3320	1917 Maxwell Touring Car
S18_3856	1941 Chevrolet Special Deluxe Cabriolet
S24_2022	1938 Cadillac V-16 Presidential Limousine
S24_2972	1982 Lamborghini Diablo

S24_4258	1936 Chrysler Airflow
S700_3505	The Titanic

S3

Question:

- Use the `customers` and `orders` for this query.
- Shipping days is the number of days between `orderDate` and `shippedDate.`
- Product a table of the form:
  - `customerNumber`
  - `customerName`
  - `noOfOrders` is the number of orders the customer placed.
  - `averageShippingDays`, which is the average shipping days.
  - `minimumShippingDays`, which is the minimum shipping days.
  - `maximumShippingDays`, which is the maximum shipping days.
- The table should only contain entries where:
  - `noOfOrders >= 3`
  - `averageShippingDays >= 5` or `maximumShippingDays >= 10.`

Answer:

In [16]:

```
%%sql

SELECT customers.customerNumber,
       customerName,
       COUNT(*) AS noOfOrders,
       AVG(datediff(shippedDate, orderDate)) AS averageShippingDays,
       MIN(datediff(shippedDate, orderDate)) AS minimumShippingDays,
       MAX(datediff(shippedDate, orderDate)) AS maximumShippingDays
FROM customers
INNER JOIN orders
    ON customers.customerNumber = orders.customerNumber
GROUP BY customers.customerNumber
HAVING noOfOrders >= 3 AND
       (averageShippingDays >= 5 OR maximumShippingDays >= 10)
```

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

Out[16]:

customerNumber	customerName	noOfOrders	averageShippingDays	minimumShippingDays	maximumShippingDays
363	Online Diecast Creations Co.	3	5.0000	4	6
385	Cruz & Sons Co.	3	5.3333	5	6
148	Dragon Souveniers, Ltd.	5	14.6000	1	65
198	Auto-Moto Classics Inc.	3	5.6667	5	6
161	Technics Stores Inc.	4	5.2500	4	6
205	Toys4GrownUps.com	3	5.3333	4	6
276	Anna's Decorations, Ltd	4	5.0000	4	6
462	FunGiftIdeas.com	3	5.0000	3	6
448	Scandinavian Gift Ideas	3	5.5000	5	6
328	Tekni Collectables Inc.	3	5.0000	4	6
209	Mini Caravy	3	5.6667	5	6
398	Tokyo Collectables, Ltd	4	5.5000	2	8

Graph Database — Neo4j

- You will use your online/cloud Neo4j database for these problems.
- You must have loaded the Movie sample data.

N1

Question:

- The relationship `REVIEWED` connects a `Person` and `Movie`, and has the properties `rating` and `summary`.
- Write Python code using `py2neo` that produces the following table.

In [17]:
```
import pandas as pd
```

Answer:

In [18]:
```
graph = Graph(neo4j_url, auth=(neo4j_user, neo4j_password))
q = "MATCH p=(n:Person)-[r:REVIEWED]->(m:Movie) return n.name , r.rating, r.summary, m.title"
result = graph.run(q)
df = pd.DataFrame(result,columns=['reviewer_name', 'rating', 'rating_summary', 'movie_title'])
df.sort_values(by=['reviewer_name'],ignore_index=True)
```

Out[18]:

	reviewer_name	rating	rating_summary	movie_title
0	Angela Scope	62	Pretty funny at times	The Replacements
1	James Thompson	100	The coolest football movie ever	The Replacements
2	James Thompson	65	Fun, but a little far fetched	The Da Vinci Code
3	Jessica Thompson	92	You had me at Jerry	Jerry Maguire
4	Jessica Thompson	65	Silly, but fun	The Replacements
5	Jessica Thompson	45	Slapstick redeemed only by the Robin Williams ...	The Birdcage
6	Jessica Thompson	85	Dark, but compelling	Unforgiven
7	Jessica Thompson	95	An amazing journey	Cloud Atlas
8	Jessica Thompson	68	A solid romp	The Da Vinci Code

N2

Question:

- There are relationships `ACTED_IN` and `DIRECTED` between `Person` and `Movie`.
- Write Python code that produces the following table that shows people or both acted in and directed a movie.

In [19]:
```
graph = Graph(neo4j_url, auth=(neo4j_user, neo4j_password))
q = "MATCH p=(a:Person)-[:ACTED_IN]->(m:Movie)<-[:DIRECTED]-(b:Person) WHERE a=b return a.name, m.title"
result = graph.run(q)
df = pd.DataFrame(result,columns=['Name', 'Title'])
df
```

Out[19]:

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

MongoDB

- Run the following code using your Atlas MongoDB.

In [20]:
```
import json

client = pymongo.MongoClient(
    mongodb_url
)

with open("episodes.json") as e_file:
    episodes = json.load(e_file)

episodes = episodes['episodes']

for e in episodes:
    client['w4111_final']['episodes'].insert_one(e)
```

In [21]:
```
ratings_df = pd.read_csv("got_title_ratings.csv")
ratings_info = ratings_df[['tconst', 'averageRating', 'numVotes']]
r_dict = ratings_info.to_dict("records")

for r in r_dict:
    client['w4111_final']['ratings'].insert_one(r)
```

Question:

Write Python code that uses an aggregation pipeline and operations to produce the following table.

In [22]:
```
# Requires the PyMongo package.
# https://api.mongodb.com/python/current
#
# Write the query/aggregation that produces result
```

In [23]:
```
collection_episodes = client['w4111_final']['episodes']

result = collection_episodes.aggregate( [
        {
            '$set': {
                'episodeLink': { '$substr': ['$episodeLink', 7, 9] }
            }
        },
        {
            '$lookup':
            {
                'from': 'ratings',
                'localField': 'episodeLink',
                'foreignField': 'tconst',
                'as': 'rating_info'
            }
        },
        {
            '$addFields': {
                'avgRating': {'$arrayElemAt':['$rating_info.averageRating',0]},
                'numVotes': {'$arrayElemAt':['$rating_info.numVotes',0]}
            }
        },
        {
            '$project': {
            'seasonNum': 1,
            'episodeNum': 1,
            'episodeLink': 1,
            'episodeTitle': 1,
            'avgRating': 1,
            'numVotes': 1
            }
        }

])
info_df = pd.DataFrame(list(result))
info_df = info_df[['seasonNum', 'episodeNum', 'episodeLink', 'episodeTitle', 'avgRating', 'numVotes']]
info_df
```

Out[23]:

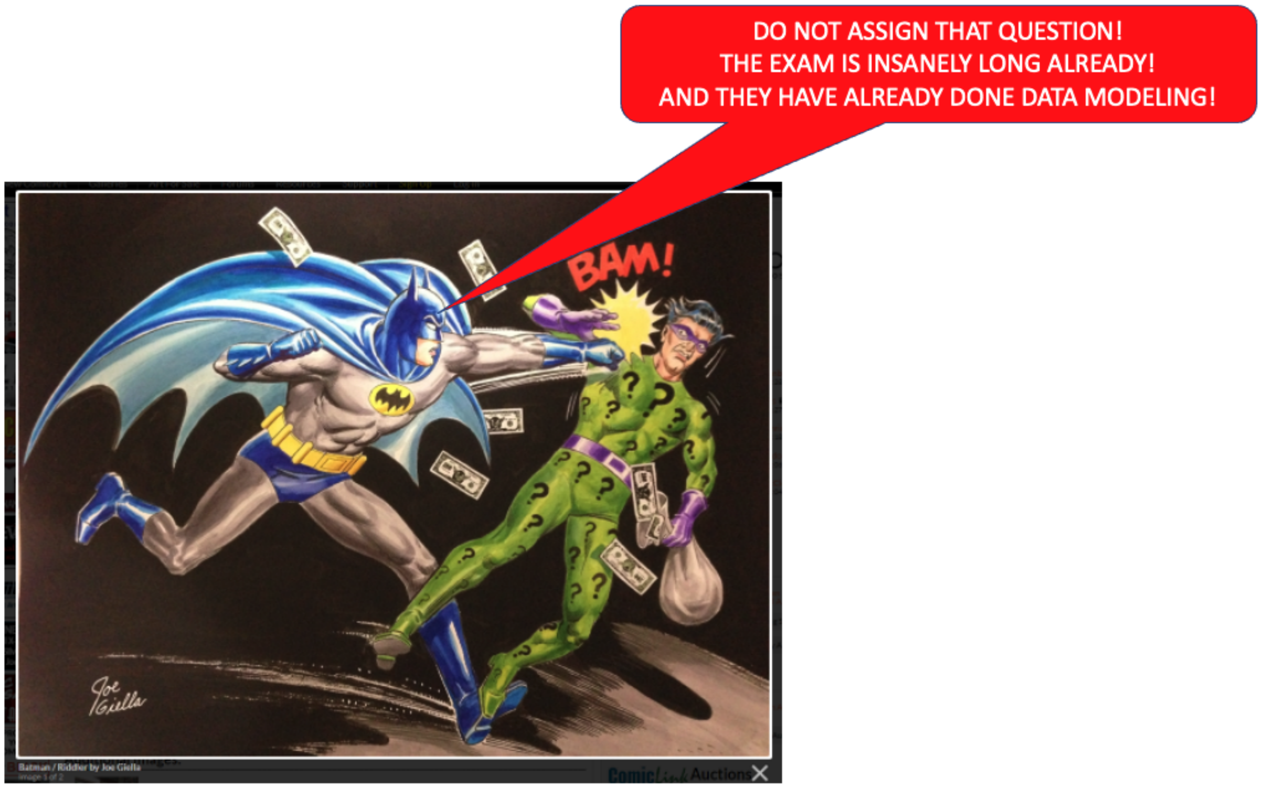
	seasonNum	episodeNum	episodeLink	episodeTitle	avgRating	numVotes
0	1	1	tt1480055	Winter Is Coming	8.9	48686
1	1	2	tt1668746	The Kingsroad	8.6	36837
2	1	3	tt1829962	Lord Snow	8.5	34863
3	1	4	tt1829963	Cripples, Bastards, and Broken Things	8.6	33136
4	1	5	tt1829964	The Wolf and the Lion	9.0	34436
...	...	...	...	...	...	...
68	8	2	tt6027908	A Knight of the Seven Kingdoms	7.9	130844
69	8	3	tt6027912	The Long Night	7.5	215995
70	8	4	tt6027914	The Last of the Starks	5.5	165067
71	8	5	tt6027916	The Bells	6.0	192449
72	8	6	tt6027920	The Iron Throne	4.0	248318

73 rows × 6 columns

Data Modeling and Schema Definition

- This is an exciting, interesting problem that involves:
  - Using Crow's Foot Notation
  - Relational approaches to implementing specialization, aggregation, quaternary relations, composite attributes and multi-valued attributes.
  - Foreign keys, check constraints and triggers.
- I did the answer and it took 3 hours to do all the work. My normal rule of thumb is that students require about 15 times as much time as I need to produce an answer.

- I giggled like the Riddler in Batman about how much fun we were going to have working on this question, and then the following happened.


DO NOT ASSIGN THAT QUESTION!
THE EXAM IS INSANELY LONG ALREADY!
AND THEY HAVE ALREADY DONE DATA MODELING!
BAM!

- So, there will not be any data modeling question on the exam. Darn!

Module II Questions

- The questions require brief, written answers.

Q1

Question:

Briefly explain:

- Functional Dependency
- Lossy Decomposition
- Normalization

Answer:

1. Functional Dependencies: It is a constraint in a DBMS that establishes the relationship between one attribute and another attribute.
2. Lossy Decomposition: It is the process of a relation be divided into several relational schemas, which results in information being lost when the original relation is retrieved.
3. Normalization: In order to reduce redundancy from a relation or group of relations, normalization splits the larger table into smaller ones and connects them using relationships.

Q2

Question:

Briefly explain:

- Serializability
- Conflict Serializability
- Deadlock
- Cascading Abort
- Two Phase Locking

Answer:

1. Serializability: The primary accuracy test for concurrent transaction executions is serializability. It is regarded as the highest level of transaction isolation and is crucial to concurrency control.
2. Conflict Serializability: a schedule is conflict serializable if it is conflict equivalent to a serial schedule
3. Deadlock: It is a case that two or more transactions are waiting indefinitely for one another to give up locks
4. Cascading Abort: In order to stop the second transaction from accessing invalid data, one transaction must terminate in order for the other to continue.
5. Two Phase Locking: Whenever Locking and Unlocking may be completed in two phases: 1)Growing Phase: Although no locks on data items can be released, new ones may be obtained. 2)Shrinking Phase: Only existing locks may be released during this time; no new locks may be purchased.

Q3

Question:

Briefly explain:

- Logical block addressing, CHS addressing
- RAID-0, RAID-1, RAID-5

- Fixed length records, variable length records.

Answer:

1. Logical block addressing: It is a method for identifying the position of data blocks stored on computer storage devices, typically secondary storage systems like hard disk drives.
2. CHS addressing: It is the method of locating specific disk sectors based on where they are located within a track, where the track is identified by the head and cylinder numbers.
3. RAID-0: it requires at least two drives, we can combine the drives and write data to both of them at once or sequentially, depending on the system.
4. RAID-1: RAID-1 will duplicate data and save a copy on each drive if we have at least two drives. Mirroring prevents file loss in the event of a drive failure.
5. RAID-5: Using RAID-5 will divide data into segments and save those parts across drives if we have at least three hard drives.
6. Fixed length records: the length of the fields in each record has been set to be a certain maximum number of characters long.
7. Variable length records: the length of a field can change to allow data of any size to fit.

Q4

Question:

Briefly explain:

- Clustered Index
- Sparse Index
- Covering Index

Answer:

1. Clustered Index: the index whose search key specifies the sequential order of the file.
2. Sparse Index: contains index records for only some search-key values.
3. Covering Index: an index covers the query if all the columns specified in the query are part of the index

Q5

Question:

Briefly explain:

- Equivalent queries
- Hash Join
- Materialization, Pipelining

Answer:

1. Equivalent queries: queries that will produce the exactly the same answer.
2. Hash join: It is a way of executing a join where a hash table is used to find matching rows between one or more tables
3. Materialization: generate results of an expression whose inputs are relations or are already computed, store it on disk, then repeat
4. Pipelining: pass on tuples to parent operations even the operation is still in progress