Problem 1: In this problem, you have to write SQL and RA queries for an online book store database.

Here are the tables:

Customers: (Customer\_id, Customer\_Last\_Name, Customer\_First\_Name, City, State, Zip)

Publisher: (Publisher\_id, Pub\_Name, Headquarter, Phone)

Author: (Author\_id, Last\_Name, First\_Name)

Category: (Category\_id, Category\_name)

Books: (ISBN, Author\_id, Title, PublishDate, Publisher\_id, Price, Category\_id)  
Publisher\_id references Publisher(Publisher\_id)  
Category\_id references Category(Category\_id)  
Author\_id references Author(Author\_id)

OrderItems: (Order\_id, Item\_id, ISBN, Quantity, Price\_Per\_Item)  
ISBN referencesBooks(ISBN)

Orders: (Order\_id, Customer\_id, OrderDate, ShipDate, ShipStreet, ShipCity, ShipState, ShipZip, ShipCost)  
Customer\_id references Customers(Customer\_id)

This schema provides data on an online book store’s sales. There are customers, each with a unique id, who can buy books from this store. Each book uses an ISBN as a unique identifier and we may assume each book only has one author. The customers can buy multiple books within one order, and if an order has five distinct books, we would use Item\_id values of 1 to 5 for these. A customer may also buy multiple copies of the same book, which is stored in the Quantity attribute in OrderItems. The entity Category models different types or genres of books, say Comics, Horror, Sports etc.

1. Draw an ER diagram that models this relational schema. Identify any weak entities and the cardinalities of all the relationships.(8 marks)

Diagram

Description automatically generated

Weak Entity set: Order

* Category wasn’t considered a weak entity set because it’s assumed that a category can exist and not have any books

1. Create the above schema in a database system, choose appropriate attributes types, and define primary keys, foreign keys and other constraints. You may use any relational database system, as long as it supports basic SQL, views, and some sort of triggers. Write the create table statements.(5 marks)

CREATE TABLE customers (  
 Customer\_id INT NOT NULL,  
 Customer\_Last\_Name VARCHAR(40) NOT NULL,  
 Customer\_First\_Name VARCHAR(40) NOT NULL,  
 City VARCHAR(30),  
 State VARCHAR(15),  
 Zip VARCHAR(15),  
 PRIMARY KEY (Customer\_id));

CREATE TABLE publisher (  
 Publisher\_id INT NOT NULL,  
 Pub\_Name VARCHAR(45) NOT NULL,  
 Headquarter VARCHAR(45),  
 Phone VARCHAR(15),  
 PRIMARY KEY (`Publisher\_id`));

CREATE TABLE author (  
 Author\_id INT NOT NULL,  
 Last\_Name VARCHAR(40),  
 First\_Name VARCHAR(40),  
 PRIMARY KEY (Author\_id));

CREATE TABLE category (  
 Category\_id INT NOT NULL,  
 Category\_name VARCHAR(20),  
 PRIMARY KEY (Category\_id));

CREATE TABLE books (  
 ISBN varchar(17),  
 Author\_id int references author(author\_id),  
 Title VARCHAR(45),  
 PublishDate date,  
 Publisher\_Id int references publisher(publisher\_id),  
 Price int NULL,  
 Category\_id int references category(category\_id),  
 PRIMARY KEY (ISBN));

CREATE TABLE orderitems (  
order\_id INT NOT NULL,  
item\_id VARCHAR(45) NOT NULL,  
ISBN VARCHAR(17) references books(ISBN),  
quantity INT NOT NULL,  
price\_per\_item INT NOT NULL,  
PRIMARY KEY (order\_id, item\_id));

CREATE TABLE orders (  
 Order\_id INT AUTO\_INCREMENT NOT NULL references orderitems(order\_id),  
 customer\_id INT references customers(customer\_id),  
 orderdate date NOT NULL,  
 shipdate date,  
 shipstreet VARCHAR(45) NOT NULL,  
 shipcity VARCHAR(45) NOT NULL,  
 shipstate VARCHAR(45) NOT NULL,  
 shipzip INT NOT NULL,  
 shipcost INT NOT NULL,  
 PRIMARY KEY (Order\_id));

(c) Write the following SQL queries and execute them on your database. Show the queries and the screenshot of the results: (22 marks)

I. For each book category, find the book with the longest name.

* For this problem, a subquery had to be used because group by would take precedence otherwise.

select title, category\_name, max  
from (select title, category\_name, max(length(title)) as max  
 from books, category  
 where books.category\_id = category.category\_id  
 group by title  
 order by length(title) desc  
 ) test  
group by category\_name;

II. Output the 10 books that had the most copies purchased in 2021, along with the number of copies purchased.

select title, quantity  
from books, orders, orderitems  
where orderitems.order\_id = orders.order\_id  
and orderitems.ISBN = books.ISBN  
and Year(orderdate) = 2021  
group by books.ISBN  
order by quantity desc  
limit 10

III. Find those orders that have not been shipped yet, and the name of the city the books are being shipped to. Display the name of the customers as well.

select distinct orders.order\_id, customer\_last\_name, customer\_first\_name, shipcity, shipdate  
from orders, customers  
where (shipdate is Null  
or shipdate > current\_date())  
and orders.customer\_id = customers.customer\_id;

IV. Show a list of authors who have written a book in the “Science Fiction” category that has never sold even a single copy in the store.

Select title, last\_name, first\_name  
from author, books, category, orderitems  
where books.author\_id = author.author\_id  
and category.category\_id = books.category\_id  
and category\_name = "Science Fiction"  
and books.isbn not in (select orderitems.isbn  
 from orderitems)

group by author.author\_id;

V. Display all distinct books purchased by a customer named “Robert Lane”.

select distinct title  
from books, customers, orders, orderitems  
where customers.Customer\_Last\_Name = "Lane"  
and customers.Customer\_First\_name = "Robert"  
and customers.customer\_id = orders.customer\_id  
and orderitems.order\_id = orders.order\_id  
and books.ISBN = orderitems.isbn;

VI. Show all books in the “Entertainment” category that were written by an author whose last name is “Kardashian”

select distinct title  
from books, author, category  
where last\_name = "Kardashian"  
and author.author\_id = books.Author\_id  
and category.category\_id = books.category\_id  
and category\_name = "Entertainment";

VII. For each year from 2010 to 2020, output the category with the highest total sales (sum of purchase prices) in that year.

select year(orderdate), category\_name, sum(price\_per\_item \* quantity) as sum\_purchase  
from category, books, orderitems, orders  
where orders.order\_id = orderitems.order\_id  
and category.category\_id = books.category\_id  
and orderitems.isbn = books.isbn  
and year(orderdate) > 2010  
and year(orderdate) < 2020  
group by year(orders.orderdate)  
order by orders.orderdate asc;

VIII. Delete any publishers from the database that have never published a book.

DELETE FROM publisher

where publisher\_id not in (select publisher\_id

from books);

IX. The company decides that any customer who bought books for more than $200 in 2021 should get a free copy of the new Harry Potter book, “Harry Potter and the Castle of Doom”. (You should assume that the book has already been inserted into the database.)

Write an update query that creates records in Orders and OrderItems for this book for each such customer, with price $0.

* Orders and OrderItems have auto\_increment specified for their primary keys
* Because an order is being placed, the shipdate will be null to indicate it hasn’t been shipped yet.
* Because a query can’t be inserted into two tables at once, two insert queries are used

insert into orders(Order\_id, customer\_id, OrderDate, shipdate, shipstreet, shipcity, shipstate, shipzip, shipcost)

(select   
CASE orders.order\_id  
 WHEN orders.order\_id < (select max(order\_id) from orders) then orders.order\_id = (select max(order\_id from orders) + 1  
END  
, customer\_id, current\_date(), null, shipstreet, shipcity, shipstate, shipzip, shipcost  
from orders, orderitems  
where orderitems.order\_id = orders.order\_id  
and year(orderdate) = 2021  
group by customer\_id  
having sum(quantity\*price\_per\_item) > 200  
);

* ‘HARRY POTTER’ is a placeholder ISBN for what the actual ISBN for the corresponding ISBN for Harry Potter and the Castle of Doom book.
* Because the item\_id is labelled 1-5 for the set of books we do have, 6 will be used to identify the free harry potter book

insert into orderitems(order\_id, item\_id, isbn, quantity, price\_per\_item)  
(select orderitems.order\_id, 6, 'HARRY POTTER', 1, 1  
from orderitems, orders  
where orderitems.order\_id = orders.order\_id  
group by customer\_id  
having sum(quantity\*price\_per\_item) > 200  
)  
;

X. Increase the price of each book in the category “Physics” by 10%. (Only increase the current price of the books, not the amount paid in past or current orders.)

update books, category  
 set price = price \* 1.1  
where category.category\_id = books.category\_id  
and category\_name = "Physics"

1. Write expressions in Relational Algebra for queries I - VII.(14 marks)
2. For each book category, find the book with the longest name.

S1<- (category\_name, len(title)) len(title) as total\_length(Books ⋈books.category\_id=category.category\_id Category)

S2 <- (category\_name, max(total\_length)) max(total\_length) as max\_length(S1)

πtitle,category\_name,max\_length(Category⋈category.category\_name=S2.category\_name S2)

1. Output the 10 books that had the most copies purchased in 2021, along with the number of copies purchased.

S1<- as total\_sold (σYear(OrderDate)=2021 (Order ⋈orders.order\_id=orderitems.order\_id (Books⋈orderitems.isbn=books.isbn Orderitems))

S2<- as max\_sold (S1

πtitle, sum(quantity), total\_sold (σorders.quantity≥S2.max\_sold (Orders⋈orders.order\_id=S2.order\_id S2)))

1. Find those orders that have not been shipped yet, and the name of the city the books are being shipped to. Display the name of the customers as well.

Πcustomer\_last\_name,customer\_first\_name,shipcity (σshipdate=null ∨shipdate>Date(Current\_Date) (Customers ⋈customer.customer\_id=orders.customer\_id orders))

1. Show a list of authors who have written a book in the “Science Fiction” category that has never sold even a single copy in the store.

Πtitle,last\_name, first\_name (σcategory\_name=”Science Fiction” (Author ⋈books.author\_id = author.author\_id (Category ⋈category.category\_id=books.category\_id books)))

-Πtitle,last\_name, first\_name (orderitems⋈books.isbn =orderitems.isbn (σcategory\_name=”Science Fiction” (Author ⋈books.author\_id = author.author\_id (Category ⋈category.category\_id=books.category\_id books)))

V. Display all distinct books purchased by a customer named “Robert Lane”

Πbooks.isbn,title (σCustomer\_First\_name=”Robert ∧Customer\_Last\_Name=”Lane” (customer ⋈customer.customer\_id=orders.customer\_id (orders⋈orders.order\_id=orderitems.order\_id (orderitems⋈books.isbn=orderitems.isbn Books))))

VI. Show all books in the “Entertainment” category that were written by an author whose last name is “Kardashian”

Πisbn,title, last\_name, first\_name (σlast\_name=”Kardashian” (author⋈author.author\_id=books.author\_id (σcategory\_name=”Entertainment” (books⋈books.category\_id=category.category\_id category))))

VII. For each year from 2010 to 2020, output the category with the highest total sales (sum of purchase prices) in that year.

(category\_name, len(title)) len(title) as total\_length

Y1 <-(orderitems.order\_id, category\_name,(price\_per\_item\*quantity) σorderdate≥2010 ∨orderdate ≤ 2020 (Category⋈category.category\_id=Books.category\_id (Books ⋈books.isbn=orderitems.isbn (Orderitems ⋈order.order\_id=orderitems.order\_id orders)))))

( (σY1.order\_id=orderitems.order\_id (Y1 x orderitems))

Problem 2: In this problem, you need to create views and triggers given the following relational schema about a chocolate-tasting experiment. Based on the rating from users, you should help the producer decide if the production of a particular chocolate needs to be increased or decreased, and which city has an effect on demand and supply.

Chocolate (choc\_id, choc\_name, brand, category, price, weight, calorie)

User(uid, uname, preferred\_category, city, country)

Rating (uid, choc\_id, timestamp, rating, comment)

For each chocolate, we have a unique choc\_id, the choc\_name, the brand of the chocolate (e.g., Reese’s, Hershey, Lindt ...), category of the chocolate product (e.g. truffles, bar, candy, ...), the price, the net weight, and the calorie count. Whenever a user tastes a chocolate and adds his/her rating for a chocolate, its timestamp is recorded. Ratings are from 1 to 5, and the user can also add a comment with the rating. For each user, we have a unique uid, a user name, the user’s preferred category of chocolate, and the city and country.

1. Define a view that stores, for each user, the choc\_id, choc\_name, brand, weight, calories, and rating, of all chocolates in the user’s preferred category that the user has rated.(5 marks)

* Because user is a keyword in MySQL, the table user was renamed to users

Create view v as (select chocolate.choc\_id, choc\_name, brand, weight, calorie, rating  
 from chocolate, users, rating  
 where chocolate.choc\_id = rating.choc\_id  
 and rating.uid = users.uid  
 and users.preferred\_category = chocolate.category  
 and rating is not NULL)

1. Using this view, output the choc\_name and brand that has the lowest calories per weight, among those that the user gave a rating of 5. (5 marks)

* Assuming this question was asking for the lowest calories per weight for every choc\_name and brand found

select choc\_name, brand, rating, (calorie/weight) as calories\_per\_weight  
from v  
where rating = 5  
group by choc\_name, brand  
order by calories\_per\_weight asc

1. Users can rate the same chocolate several times, but we want to limit how often they can rate it. Write a trigger that rejects any insertion of a new rating of a particular chocolate (identified by choc\_id) in a specific month by a user(uid) who had already rated the same chocolate three times in that month.(8 marks)

* Delimiter was needed for these SQL queries to work for me.

DELIMITER $$

create trigger rate\_limit before insert on rating  
for each row  
begin  
 declare test INT;  
 set test = (select count(choc\_Id)  
 from rating  
 where month(rating.`timestamp`) = month(new.`timestamp`)  
 and new.choc\_id = rating.choc\_id  
 group by choc\_id  
 );  
 if (test > 3)  
 then signal sqlstate '45000';  
 end if  
end;  
DELIMITER;

Problem 3: Suppose you are helping company Max.Inc build up a new database for tracking job candidates. The database should record the information about job candidates, interviewers, and recruiters.

The database has to store basic personal information such as first name, last name, resume (a piece of text), phone number, email address, position applied for, and application date for each candidate. Each candidate can only have one current application for a job, and can only apply for three positions within 12 months. Open positions for which candidates may apply are identified by a position ID, a short description, a department in which the position is located, and a set of tags describing the expertise needed for the position (see below for more on tags).

After a candidate applied for a certain position, the status of this candidate should be marked as “applied”. If a recruiter decides to move the candidate to the next step, the status should be changed to “in progress”, or otherwise the status can be changed to “rejected”. We also need to store some basic information about the recruiters, say their name, phone number, and maybe an ID. For each application, there is one recruiter who is responsible for it.

In addition to recruiters, there are interviewers, who are company employees that are available to interview applicants. Basic information about interviewers is stored in the database, including their names, phone numbers, department, and expertise. An interviewer’s expertise is stored as a set of tags chosen from a fixed dictionary of tags (e.g., “production”, “databases”, “sales”). The same dictionary of tags is also used to describe the expertise needed for a position (see above). After an application is moved to “in progress”, the recruiter will select a set of interviewers, and schedule a phone interview between the applicant and each of the chosen interviewers. The actual communication to find a convenient time for each interview is done outside the database, say via email, so you do not have to model that. However, once an interview has been scheduled, the information about the interview (who takes part in the interview and when it takes place) has to be stored in the database. After the interview completes, each interviewer can upload a short evaluation of the candidate and a hiring recommendation from 0 to 3 (3=“definite hire”, 2=“possible hire”, 1=“probably not”, or 0=“definitely do not hire”).

Finally, the recruiter will change the status of the application to either “make offer” or “reject”.

1. Design a database for the above scenario using the ER model. Draw the ER diagram, show the cardinalities of all relationships, and identify primary keys and any weak entities. List any assumptions you have as well. (10 marks)

This assumes that every interview meeting is a one-on-one interview.

Diagram

Description automatically generated

Weak Entity set: interview

1. Convert your ER diagram into a relational schema. Identify all tables, attributes, primary keys, and foreign keys. (5 marks)

**Interviewers**: (**i\_id**(Primary Key), Last\_Name, First\_Name, Department, expertise)

* I\_id: int, primary key
* Last\_Name: varchar(20)
* First\_Name: varchar(20)
* Department: varchar(20)
* Expertise: varchar(15)

**Candidate: (cid** (Primary Key), Last\_Name, First\_Name, resume, phone\_number, email\_address, desired\_position)

Assuming a resume can be optional for submission.

* cid: Int, primary key
* Last\_name: varchar (20)
* First\_Name: varchar(20)
* Resume: varchar(50)
* Phone\_number: varchar(15), not null
* Email\_address: varchar(30), not null
* Desired\_position: varchar(20), not null

**Recruiter: (rid** (Primary Key), Last\_Name, First\_Name, department)

* Rid: int, primary key
* Last\_Name: varchar(20), not null
* First\_Name: varchar(20), not null
* Department: varchar(20), not null

**Interview: (**cid, i\_id, date, recommendation)  
cid references Candidate(cid)  
i\_id references Interviewers(i\_id)cid: foreign key

* i\_id: foreign key, int
* cid: foreign key, int
* date: dateandtime, not null
* recommendation: int, not null

**Application:** (aid (Primary Key), cid, rid, app\_date, department, expertise, status, position)  
cid references candidate(cid)  
rid references recruiter(rid)

* aid: int, primary key
* cid: int, foreign key
* rid: int, foreign key
* app\_date: dateandtime
* department: varchar(20), not null
* expertise: varchar(15), not null
* status: varchar(20), not null
* position: varchar(20), not null

1. Write SQL statements for the following questions or updates. If you cannot answer a query using your schema, then you have to modify your solutions in (a) and (b) appropriately.(8 marks)
2. For a particular “in progress” application, say identified by an applicant’s name or application ID, list all interviewers that have at least one expertise that is required for the position.

select distinct interviewer.last\_name, interviewer.first\_name  
from application, interviewer  
where status='in progress'  
and interviewer.expertise = application.expertise

1. List the three positions posted during 2021 that received the most applications.

select position, count(distinct application.aid) as app\_count  
from application  
where year(appdate) = 2021  
group by application.position  
order by app\_count desc  
limit 3

1. For each reviewer, output their name, the number of interviews they did in 2021, and the average rating they gave in 2021.

* Assuming that reviewer actually means interviewer

select interviewer.last\_name, interviewer.first\_name, count(interview.i\_id), avg(recommendation)  
from interviewer, interview  
where interview.i\_id = interviewer.i\_id  
and year(interview.`date`) = 2021  
group by interview.i\_id

1. Output the names of any applicants, and the job they applied for, where an application was “in progress” for more than 100 days before being decided. (This includes both cases where current applications are still undecided after 100 days, and past applications where it took more than 100 days to decide an application.)

select application.aid, last\_name, first\_name, position  
from candidate, application  
where application.cid = candidate.cid  
and status='in progress'  
and datediff(now(), appdate) > 100  
group by application.aid

1. Create tables in the database system, and insert some sample data (5-10 tuples per table, but choose an interesting and meaningful data set, so that queries do not output empty results). Submit screenshots of what your tables look like after you inserted the data. Then execute the queries in (c) and submit the screenshots of the queries and outputs.(5 marks)

**Candidate table**

Graphical user interface, text, application, table

Description automatically generated

**Recruiter table**

Table

Description automatically generated

**Application table**

Graphical user interface, text, application, email

Description automatically generated

**Interviewer table**

Table

Description automatically generated

**Interview table**

Graphical user interface, text, application, chat or text message

Description automatically generated

**c) queries**

1. For a particular “in progress” application, say identified by an applicant’s name or application ID, list all interviewers that have at least one expertise that is required for the position.

Graphical user interface, text, application

Description automatically generated

1. List the three positions posted during 2021 that received the most applications.

Graphical user interface, text, application

Description automatically generated

1. For each reviewer, output their name, the number of interviews they did in 2021, and the average rating they gave in 2021.

Graphical user interface, application

Description automatically generated

1. Output the names of any applicants, and the job they applied for, where an application was “in progress” for more than 100 days before being decided. (This includes both cases where current applications are still undecided after 100 days, and past applications where it took more than 100 days to decide an application.)

Graphical user interface, text, application

Description automatically generated

1. It was stated in the description of the problem that each applicant can only apply for one position at a time, and that applicants cannot apply for more than three positions in any 12-month period. Discuss how you would enforce such a requirement in your system. Would you do this via integrity constraints? Or using a trigger? Or have the recruiter manually check this and if needed reject additional applications? Justify your answer.(5 marks)

Integrity constraints only define what each attribute can or cannot be. While it is p possible to have a recruiter manually check and reject every additional application, that person cannot be reasonably expected to keep count of every application submitted and compare it to any other possible applications sent by the same candidate.

A trigger can be designed such that whenever a new application is inserted, the SQL server will initialize an SQL query that counts the number of applications sent by the candidate using the same candidate ID to see how many applications that person has sent. The SQL subquery will peruse through the list of applications sent by the matching applicant.

This subquery can use a loop that, for every tuple found, it will check for any other applications such that datediff(application\_1.appdate-application\_2.appdate) > 365. In addition, for every submitted application found, the trigger will check if application\_1.position = application\_2.position. If two positions from two different applications match, then the insert query is automatically rejected.

If this if condition is fulfilled, then both applications will be counted. If the total count > 3, then the application insert is automatically rejected. Otherwise, the application will compare the next set of applications and their dates to see if they fit within the 12 month time-span.

If the total counting of applications < 3 for all applications of the same candidate, then the application is inserted into the application table.