**HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**Department of Information & Communication Technology**

-------------------------------



**DATABASE PROJECT REPORT**

**Project:** Book Store Management

**Lecturer:** Dr. Tran Viet Trung

***Students:***

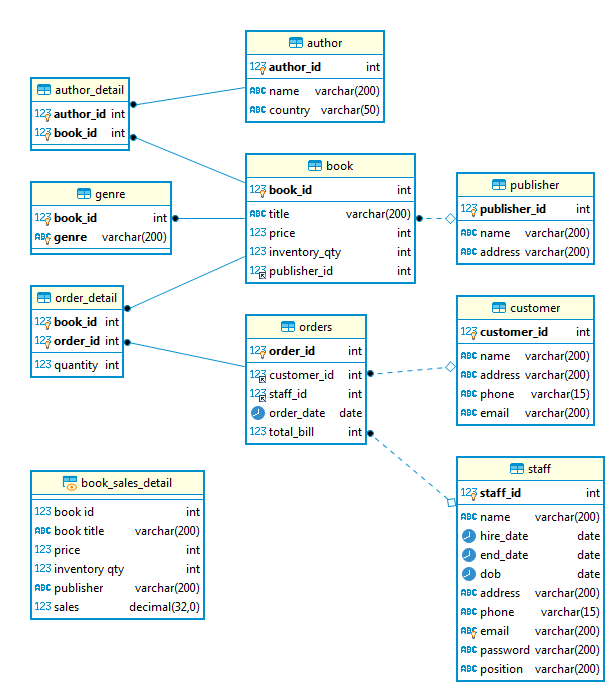
Trần Hải Sơn – 20176861

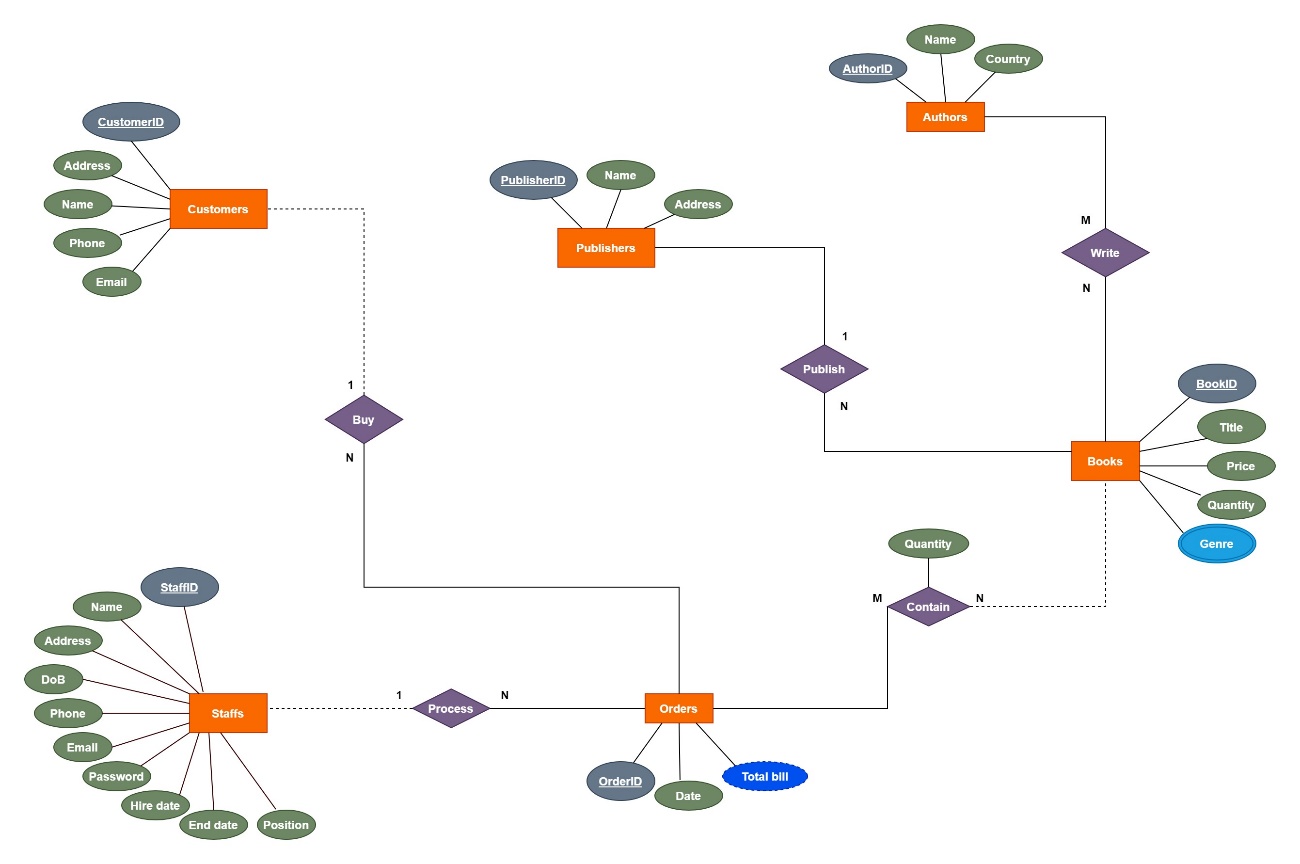
Trần Lê Hoàng – 20176764

Hoàng Tuấn Anh Văn – 20170224

-------------------------------

1. **Introduction:**
   1. **The database design:**
      * Most significantly this database is designed to manage books in the store, as well as information related to these books such as the total number of books available, authors, genres and publishers.
      * Additionally, the database system has the capability to manage books’ orders, total bills and the customers’ information, together with the information of all the staffs working in the bookstore.
   2. **Data requirements:**
      * **Book data**: information related to its title, author, publisher, genre and available inventory quantity
      * **Order data**: information related to books, order date, customer, total bill, and cashier
      * **Customer data**: information related to the customer’s name, address, phone, email
      * **Staff data**: information related to the staff’s name, hire date, end date, account and position
   3. **ER Diagram and relation detail:**
      * Table ***book***
        + *book\_id* which is used to identify books, represents primary key
        + *title* is used to represent book’s title
        + *price* stores the book’s price
        + *inventory****\_****qty* stores the quantity of available books in stock
        + *publisher*\_*id* is foreign key, references to *publisher*\_*id* in *publisher* table
        + 2 indexes are created in this table, one is on *book*\_*id*, the other is on *publisher*\_*id*
      * Table ***author***
        + *author\_id* which is used to identify authors, represents primary key
        + *name* stores the author’s name
        + *country* stores the author’ country
        + 1 index is created, which is on *author*\_*id*
      * Table ***author\_detail***
        + *author*\_*id* is foreign key, references to *author*\_*id* in *author* table
        + *book\_id* is foreign key, references to *book\_id* in *book* table
        + Both *author\_id* and *book\_id* form primary key in this table
        + 2 indexes are created in this table, one is on 2 fields (*book\_id, author\_id*), the other is on *book\_id*
      * Table ***genre***
        + *book\_id* is foreign key, references to *book\_id* in *book* table
        + *genre* stores the book’s genre
        + Both *genre* and *book\_id* form primary key in this table
        + 1 index are created in this table, which is on 2 fields (*book\_id, gerne*)
      * Table ***publisher***
        + *publisher\_id* which is used to identify publishers, represents primary key
        + *name* stores the publisher’s company name
        + *address* stores the address of the publisher
        + 1 index are created in this table, which is on *publisher\_id*
      * Table ***orders***
        + *order\_id* which is used to identify orders, represents primary key
        + *customer\_id* is foreign key, references to *customer\_id* in *customer* table
        + *staff\_id* is foreign key, references to *staff\_id* in *staff* table
        + *order*\_*date* stores the date which the order is created
        + *total*\_*bill* stores total money that customer have to pay for the order (when *order*\_*detail* is updated, *total*\_*bill* is increased automatically by trigger)
        + 3 indexes are created in this table, the first one is on *order*\_*id*, the second one is on *customer*\_*id* and the remaining one is on *staff*\_*id*
      * Table ***order\_detail***
        + *book*\_id is foreign key, references to *book*\_*id* in *book* table
        + *order*\_*id* is foreign key, references to *order*\_*id* in *order* table
        + *quantity* stores the quantity of selected book in order
        + Both *book*\_*id* and *order*\_*id* form primary key in this table
        + 2 indexes are created in this table, one is on 2 fields (*book*\_*id*, *order*\_*id*), the other is on *order*\_*id*
      * Table ***customer***
        + *customer*\_*id* which is used to identify customers, represents primary key
        + *name* stores the customer’s name
        + *address* stores the customer’s address
        + *phone* stores the customer’s phone
        + *email* stores the customer’s email
        + 1 index are created in this table, which is on *customer*\_*id*
      * Table ***staff***
        + *staff*\_*id* which is used to identify staffs, represents primary key
        + *name* stores the staff’s name
        + *hire*\_*date* stores the date that the staff was hired
        + *end*\_*date* stores the date that the staff ended his/her job
        + *dob* stores the staff’s date of birth
        + *address* stores the staff’s address
        + *phone* stores the staff’s phone
        + *email* stores the unique staff’s email, which is also the username to login
        + *password* stores the staff’s account password
        + *position* stores information about the staff’s position in the bookstore (such as manager, cashier, security, …)
        + 2 indexes are created in this table, one is on *staff*\_*id*, the other is on *email*





1. **Queries:**
   1. Retrieve total revenue of books by day

**SELECT** order\_date **date**, **sum**(total\_bill) **AS** `total revenue`

**FROM** orders

**GROUP** **BY** order\_date

**ORDER** **BY** order\_date **ASC**;

* 1. Retrieve infomation of customers who bought more than 3 Fantasy books

**SELECT** c.\*

**FROM** customer c, orders o, order\_detail od, genre g

**WHERE** c.customer\_id = o.customer\_id **AND** c.customer\_id = o.customer\_id

**AND** o.order\_id = od.order\_id **AND** g.book\_id = od.book\_id

**AND** g.genre = 'Fantasy'

**GROUP** **BY** c.customer\_id

**HAVING** **COUNT**(\*) >= 3;

* 1. Retrieve author(s) that co-operate with only one publisher

**SELECT** **DISTINCT** a.name 'author', p.name 'publisher'

**FROM** author a, author\_detail ad, book b, publisher p

**WHERE** a.author\_id = ad.author\_id **AND** ad.book\_id = b.book\_id **AND** p.publisher\_id = b.publisher\_id

**AND** 1 = (

**SELECT** **COUNT**(**DISTINCT** b1.publisher\_id)

**FROM** author a1, author\_detail ad1, book b1

**WHERE** a1.author\_id = ad1.author\_id **AND** ad1.book\_id = b1.book\_id

**AND** a1.author\_id = a.author\_id

);

* 1. Retrieve unsold books in the past 6 month and its last sold date (null if that book is unsold)

**SELECT** b.title, last\_sold.**date** **AS** 'last sold', b.book\_id

**FROM** (

**SELECT** **MAX**(o.order\_date) 'date', od.book\_id

**FROM** orders o, order\_detail od

**WHERE** o.order\_id = od.order\_id

**GROUP** **BY** od.book\_id

) last\_sold

**RIGHT** **JOIN** book b **ON** b.book\_id = last\_sold.book\_id

**WHERE** b.title **not** **in** (

**SELECT** **DISTINCT** b1.title

**FROM** order\_detail od

**JOIN** book b1 **ON** od.book\_id = b1.book\_id

**JOIN** orders o **ON** o.order\_id = od.order\_id

**WHERE** o.order\_date >= DATE\_SUB(**CURDATE**(), **INTERVAL** 180 **DAY**)

);

* 1. Retrieve information of customers who spent more than 2000000 in 2020, sort in descending order

**SELECT** c.\*, **sum**(o.total\_bill) **AS** 'total spending'

**FROM** customer c

**JOIN** orders o **ON** o.customer\_id = c.customer\_id

**WHERE** **YEAR**(o.order\_date) = 2020

**GROUP** **BY** c.customer\_id

**HAVING** **SUM**(o.total\_bill) > 2000000

**ORDER** **BY** **SUM**(o.total\_bill) **DESC**;

* 1. Retrieve all books written by J.K Rowling

**SELECT** b.title

**FROM** book b

**JOIN** author\_detail ad **ON** ad.book\_id = b.book\_id

**JOIN** author a **ON** a.author\_id = ad.author\_id

**WHERE** a.name = 'J. K. Rowling';

* 1. Retrieve books that have amount in stock smaller than 3 books and were sold more than 10 copies in last 3 month

**SELECT** b.\*, **SUM**(od.quantity) **AS** 'Copies sold in last 3 month'

**FROM** book b

**JOIN** order\_detail od **ON** b.book\_id = od.book\_id

**JOIN** orders o **ON** o.order\_id = od.order\_id

**WHERE** b.inventory\_qty < 3 **AND** o.order\_date >= DATE\_SUB(**CURDATE**(), **INTERVAL** 90 **DAY**)

**GROUP** **BY** b.book\_id

**HAVING** **SUM**(od.quantity) > 10;

* 1. Retrieve the best seller in top 3 publisher having most books sold

**SELECT** b.title 'book title', p.name 'publisher name', **SUM**(od.quantity)

**FROM** book b, order\_detail od, (

**SELECT** publisher.\*

**FROM** publisher

**JOIN** book **ON** publisher.publisher\_id = book.publisher\_id

**JOIN** order\_detail **ON** order\_detail.book\_id = book.book\_id

**GROUP** **BY** book.book\_id, book.publisher\_id

**ORDER** **BY** **SUM**(order\_detail.quantity) **DESC**

**LIMIT** 3

) p

**WHERE** b.book\_id = od.book\_id **AND** b.publisher\_id = p.publisher\_id

**GROUP** **BY** b.book\_id, b.publisher\_id

**HAVING** **SUM**(od.quantity) >= **ALL** (

**SELECT** **SUM**(od1.quantity)

**FROM** book b1, order\_detail od1

**WHERE** b1.book\_id = od1.book\_id **AND** b1.publisher\_id = b.publisher\_id

**GROUP** **BY** b1.book\_id

);

* 1. Retrieve the 2019 quarter revenue in descending order

**SELECT** **QUARTER**(o.order\_date) **AS** 'quarter', **SUM**(od.quantity \* b.price) **AS** 'revenue'

**FROM** orders o, order\_detail od, book b

**WHERE** o.order\_id = od.order\_id **AND** b.book\_id = od.book\_id **AND** **YEAR**(o.order\_date) = 2019

**GROUP** **BY** **QUARTER**(o.order\_date)

**ORDER** **BY** **SUM**(od.quantity \* b.price) **DESC**;

* 1. Create procedure to retrieve books in a price range

**DELIMITER $$**

**CREATE** **PROCEDURE** book\_in\_price\_range(**IN** low **int**, **IN** high **int**)

**BEGIN**

**SELECT** book.\*

**FROM** book

**WHERE** price >= low **AND** price <= HIGH;

**END**; $$

**DELIMITER ;**

**CALL** book\_in\_price\_range(100000, 300000);

* 1. Create view which retrieves the list of all books and the amount of each book sold, sorts by the amount of sold books in descending order

**CREATE** **VIEW** book\_sales\_detail **AS**

**SELECT** b.book\_id 'book id', b.title 'book title', b.price, b.inventory\_qty 'inventory qty', p.name 'publisher', **COALESCE**(**sum**(od.quantity), 0) 'sales'

**FROM** book b **LEFT** **JOIN** order\_detail od **ON** b.book\_id = od.book\_id, publisher p

**WHERE** p.publisher\_id = b.publisher\_id

**GROUP** **BY** b.book\_id

**ORDER** **BY** sales;

* 1. In the top 3 most favorite genres, retrieve top 4 books which were bought the most of each genre

**SELECT** \* **FROM** (

**SELECT** genre\_detail\_sales.book\_id, genre\_detail\_sales.title, genre\_detail\_sales.genre, genre\_detail\_sales.sales,

**DENSE\_RANK**() **OVER** ( **PARTITION** **BY** genre\_detail\_sales.genre **ORDER** **BY** genre\_detail\_sales.sales **DESC**) **AS** 'rank'

**FROM** (

**SELECT** b.book\_id, b.title, g.genre, **sum**(od.quantity) 'sales'

**FROM** genre g, book b, order\_detail od

**WHERE** g.book\_id = b.book\_id **AND** od.book\_id = b.book\_id

**AND** g.genre **IN** (

**SELECT** genre\_sales\_rank.genre **FROM** (

**SELECT** genre\_sales.\*, **DENSE\_RANK**() **OVER** (**ORDER** **BY** sales **DESC**) sales\_rank

**FROM** (

**SELECT** g.genre, **sum**(od.quantity) 'sales'

**FROM** genre g, book b, order\_detail od

**WHERE** g.book\_id = b.book\_id **AND** od.book\_id = b.book\_id

**GROUP** **BY** g.genre

**ORDER** **BY** sales **DESC**

) genre\_sales

) genre\_sales\_rank

**WHERE** sales\_rank <= 3

)

**GROUP** **BY** b.book\_id, g.genre

**ORDER** **BY** g.genre, sales **DESC**

) genre\_detail\_sales ) temp

**WHERE** temp.`rank` <= 4;

* 1. Retrieve the average revenue per months in 2019

**SELECT** **MONTH**(o.order\_date) 'month', **round**(**avg**(o.total\_bill)) 'average revenue'

**FROM** orders o

**WHERE** **YEAR**(o.order\_date) = 2019

**GROUP** **BY** **MONTH**(o.order\_date)

**ORDER** **BY** **MONTH**(o.order\_date);

* 1. Retrieve information of the customer who bought the most number of books and that amount of books

**SELECT** c.\*, **sum**(od.quantity) 'book amount'

**FROM** orders o, customer c, order\_detail od

**WHERE** o.order\_id = od.order\_id **AND** c.customer\_id = o.customer\_id

**GROUP** **BY** c.customer\_id

**HAVING** `book amount` >= **ALL** (

**SELECT** **sum**(od1.quantity)

**FROM** orders o1, customer c1, order\_detail od1

**WHERE** o1.order\_id = od1.order\_id

**AND** c1.customer\_id = o1.customer\_id

**GROUP** **BY** c1.customer\_id

);

* 1. Retrieve books information and authors of the top 7 most favorite books

**SELECT** \* **FROM** (

**SELECT** salesinfo.\*, **dense\_rank**() **over** (**order** **by** sales **desc**) sales\_rank

**FROM** (

**SELECT** b.title, b.price, b.inventory\_qty, a.name, a.country, **sum**(od.quantity) 'sales'

**FROM** author a, author\_detail ad, book b, order\_detail od

**WHERE** a.author\_id = ad.author\_id **AND**

ad.book\_id = b.book\_id **AND** od.book\_id = b.book\_id

**GROUP** **BY** b.book\_id, a.author\_id

**ORDER** **BY** sales **DESC**

) salesinfo

) salesinfo\_withrank

**WHERE** salesinfo\_withrank.sales\_rank <= 7;

* 1. Retrieve information of customer(s) who bought more than 3 times, the total number of books they bought, and the minimum time interval between their 2 consecutive orders

**SELECT** tt.customer\_id, tt.name, **sum**(tt.books) 'total books', **min**(datediff(tt.order\_date, tt.prev\_order)) 'min interval (days)'

**FROM** (

**SELECT** c.customer\_id, c.name, o.order\_id, **sum**(od.quantity) 'books', o.order\_date, lag(order\_date, 1) **OVER** (**PARTITION** **BY** customer\_id **ORDER** **BY** customer\_id, order\_date) **AS** 'prev\_order'

**FROM** customer c, order\_detail od, orders o

**WHERE** o.customer\_id = c.customer\_id **AND** o.order\_id = od.order\_id

**AND** 3 < (**SELECT** **count**(\*) **FROM** orders o1

**WHERE** o1.customer\_id = c.customer\_id)

**GROUP** **BY** customer\_id, order\_id

) tt

**GROUP** **BY** tt.customer\_id;

* 1. Retrieve information of the order which has the highest total bill in 2019 and the cashier that was in charge of that order

**SELECT** o.order\_id, o.order\_date, o.total\_bill, s.name, s.dob, s.phone, s.email

**FROM** orders o, staff s

**WHERE** o.staff\_id = s.staff\_id

**AND** **YEAR**(o.order\_date) = 2019

**AND** o.total\_bill >= **ALL** (

**SELECT** o.total\_bill

**FROM** orders o, staff s

**WHERE** o.staff\_id = s.staff\_id

**AND** **YEAR**(o.order\_date) = 2019

);

* 1. Retrieve information of customers who bought both "Adventure", "Mystery" and "Action" books

**SELECT** **DISTINCT** c.\*

**FROM** genre g, book b, order\_detail od, orders o, customer c

**WHERE** g.book\_id = b.book\_id **AND** b.book\_id = od.book\_id

**AND** od.order\_id = o.order\_id **AND** o.customer\_id = c.customer\_id

**AND** g.genre = 'Adventure'

**AND** c.customer\_id **IN** (

**SELECT** c1.customer\_id

**FROM** genre g1, book b1, order\_detail od1, orders o1, customer c1

**WHERE** g1.book\_id = b1.book\_id **AND** b1.book\_id = od1.book\_id

**AND** od1.order\_id = o1.order\_id **AND** o1.customer\_id = c1.customer\_id

**AND** g1.genre = 'Action'

)

**AND** c.customer\_id **IN** (

**SELECT** c2.customer\_id

**FROM** genre g2, book b2, order\_detail od2, orders o2, customer c2

**WHERE** g2.book\_id = b2.book\_id **AND** b2.book\_id = od2.book\_id

**AND** od2.order\_id = o2.order\_id **AND** o2.customer\_id = c2.customer\_id

**AND** g2.genre = 'Mystery'

);

* 1. Retrieve information of the publisher(s) selling all books written by 'George R. R. Martin'

**SELECT** p.publisher\_id, p.name 'Publisher', p.address

**FROM** publisher p, author a, book b, author\_detail ad

**WHERE** p.publisher\_id = b.publisher\_id **AND** b.book\_id = ad.book\_id **AND** ad.author\_id = a.author\_id

**AND** a.name = 'George R. R. Martin'

**GROUP** **BY** p.publisher\_id

**HAVING** **count**(\*) = (

**SELECT** **count**(\*)

**FROM** book b1, author a1, author\_detail ad1

**WHERE** b1.book\_id = ad1.book\_id **AND** ad1.author\_id = a1.author\_id

**AND** a1.name = 'George R. R. Martin'

);

* 1. During the Covid-19 pandemic, in order to maintain the business, the director of the bookstore decided to fire some non-manager staffs which were newly hired in 2020. Update and retrieve the list of these staffs who were fired.

**UPDATE** staff s

**SET** s.end\_date = **now**()

**WHERE** **YEAR**(s.hire\_date) = 2020 **AND** s.`position` <> 'manager';

* 1. Create trigger to update the quantity of books left after an order is made

**CREATE** **TRIGGER** update\_book\_quantity **BEFORE** **INSERT** **ON** order\_detail

**FOR** **EACH** **ROW**

**UPDATE** book

**SET** inventory\_qty = inventory\_qty - **NEW**.quantity

**WHERE** book\_id = **NEW**.book\_id;

* 1. Create trigger to get the total bill of an order when books are added into order

**CREATE** **TRIGGER** order\_total\_bill **BEFORE** **INSERT** **ON** order\_detail

**FOR** **EACH** **ROW**

**UPDATE** orders

**SET** total\_bill = total\_bill + (

**SELECT** (price \* **NEW**.quantity)

**FROM** book

**WHERE** book\_id = **NEW**.book\_id

)

**WHERE** order\_id = **NEW**.order\_id;

* 1. Give the name of exactly 2 publishers publishing the most number of books

**SELECT** p.name, **sum**(inventory\_qty) **AS** '# books sales'

**FROM** book b, publisher p

**WHERE** b.publisher\_id = p.publisher\_id

**GROUP** **BY** p.publisher\_id

**ORDER** **BY** **SUM**(inventory\_qty) **DESC**

**LIMIT** 2;

* 1. Give the information of all the authors whose books published by publisher 'Lao Động'

**SELECT** a.name, a.country

**FROM** author **AS** a, author\_detail **AS** ad, book **AS** b, publisher **AS** p

**WHERE** b.publisher\_id = p.publisher\_id

**AND** b.book\_id = ad.book\_id

**AND** ad.author\_id = a.author\_id

**AND** p.name **LIKE** N'%Lao Động%';

* 1. Give the name, hire date and the number of books sold by staffs who have been working less than 1 year

**SELECT** s.name, s.hire\_date, **SUM**(od.quantity) **AS** book\_sold

**FROM** staff s, orders o, order\_detail od

**WHERE** s.staff\_id = o.staff\_id **AND** o.order\_id = od.order\_id

**AND** (DATEDIFF(**now**(), s.hire\_date)/365) < 1

**GROUP** **BY** s.staff\_id

**ORDER** **BY** book\_sold **DESC**;

* 1. Retrieve the titles, authors of books written by American and English authors, whose books were published by more than 3 publishers

**SELECT** b.title 'Books', a.name

**FROM** book b, author\_detail ad, author a

**WHERE** b.book\_id = ad.book\_id **AND** ad.author\_id = a.author\_id

**AND** ad.author\_id **IN** (

**SELECT** a1.author\_id

**FROM** author **AS** a1, author\_detail **AS** ad1, book **AS** b1

**WHERE** a1.author\_id = ad1.author\_id **AND** b1.book\_id = ad1.book\_id

**AND** a1.country **IN** ('USA', 'England')

**GROUP** **BY** a1.author\_id

**HAVING** **COUNT**(**DISTINCT** b1.publisher\_id) >= 3

);

* 1. Function to get the number of books published by 1 publisher

**DELIMITER $$**

**CREATE** **FUNCTION** book\_count(publisherID **INT**) **RETURNS** **INT**

**LANGUAGE** **SQL** **DETERMINISTIC**

**BEGIN**

**DECLARE** number\_of\_book **INT**;

**SELECT** **COUNT**(book\_id)

**INTO** number\_of\_book

**FROM** book

**WHERE** publisher\_id = publisherID

**GROUP** **BY** publisher\_id;

**RETURN** number\_of\_book;

**END** $$

**DELIMITER ;**

* 1. Give the name of customers who buy books in only 1 genre

**SELECT** c.name

**FROM** customer **AS** c, orders **AS** o, order\_detail **AS** od, book **AS** b, genre **AS** g

**WHERE** c.customer\_id = o.customer\_id

**AND** o.order\_id = od.order\_id

**AND** od.book\_id = b.book\_id

**AND** b.book\_id = g.book\_id

**GROUP** **BY** c.customer\_id

**HAVING** **COUNT**(**DISTINCT** g.genre) <= 1;

* 1. Retrieve the title and price of books whose price is greater than the sum of 5 books that are the cheapest

**SELECT** b.title, b.price

**FROM** book **AS** b

**WHERE** b.price > (

**SELECT** **SUM**(cheapest.price)

**FROM** (

**SELECT** price

**FROM** book

**ORDER** **BY** price

**LIMIT** 5) **AS** cheapest

);

* 1. Give the name and phone number of customers who used to buy at least 1 book written by author 'J. K. Rowling' and the total number of books he/she's already bought till now is less than 5

**SELECT** c.name, c.phone

**FROM** customer **AS** c, orders **AS** o, order\_detail **AS** od

**WHERE** c.customer\_id = o.customer\_id

**AND** o.order\_id = od.order\_id

**AND** od.book\_id **IN** (

**SELECT** b.book\_id

**FROM** book **AS** b, author\_detail **AS** ad, author **AS** a

**WHERE** b.book\_id = ad.book\_id

**AND** a.author\_id = ad.author\_id

**AND** a.name = 'J. K. Rowling'

)

**GROUP** **BY** c.customer\_id

**HAVING** **SUM**(od.quantity) < 5;