Computer Science 204:

Database Programming

Joshua McCausey

Fulfillment of “prompt A” will require us to start off with a sample database. To do this, I right clicked on the left “schemas” section and created a new schema (database). For some of the following steps I have included images to further demonstrate the process.

Graphical user interface, text, application, email

Description automatically generated

After successfully creating our test database we now need to put together the SQL query that will create the proper tables that were provided and the correct insert statements to add the proper data along with the constraints. To do this I used the syntax as shown in the following images to properly create and fill the tables.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

For some of the table inserts I filled the syntax out by hand but then decided that the Client table and the Borrower table with 80 and 300 entries respectively would just simply take too long. I made the decision to create a program that would allow me to copy and paste the text from the provided database image into my program and it would return the properly written SQL syntax to re-paste back into my database insert statement. The piece of JavaScript code I wrote was the following and the image after that was the output, properly formatted to be easily copy and pasted back into my database window.

A screenshot of a computer screen

Description automatically generated

Text

Description automatically generated

Now that our tables are all properly created and filled, it’s time to run some queries to answer the questions from C 1-14.

1: Display all contents of the client table

SELECT \* FROM client

2: First names, last names, ages and occupations of all clients

SELECT ClientFirstName, ClientLastName, ClientDoB, Occupation

FROM client

3: First and last names of clients that borrowed books in March 2018

SELECT ClientFirstName, ClientLastName

FROM client

JOIN borrower

ON client.ClientId = borrower.ClientId

WHERE BorrowDate BETWEEN ‘2018-03-01’ AND ‘2018-03-30’

4: First and last names of the top 5 authors clients borrowed in 2017

The answer I came up with for this prompt WILL return a third column which was not asked for called “BookTotals” which is basically a sum of how many times BookId showed up in the borrower field. For me this option makes the most sense for clarity even with the third column visible.

SELECT AuthorFirstName, AuthorLastName, COUNT(br.BookId) AS BookTotals

FROM author a

JOIN book b ON a.AuthorId = b.BookAuthor

JOIN borrower br ON b.BookId – br.BookId

GROUP BY br.BookId

ORDER BY BookTotals DESC LIMIT 5

5: Nationalities of the least 5 authors that clients borrowed during the years 2015-2017

The answer to this question is very similar to the previous query. All we need to do is remove the DESC function and add a WHERE clause on the BorrowDate field in the borrower table.

SELECT AuthorFirstName, AuthorLastName, COUNT(br.BookId) AS BookTotals

FROM author a

JOIN book b ON a.AuthorId = b.BookAuthor

JOIN borrower br ON b.BookId – br.BookId

WHERE BorrowDate BETWEEN ‘2015-01-01’ AND ‘2017-01-01’

GROUP BY br.BookId

ORDER BY BookTotals LIMIT 5

6: The book that was most borrowed during the years 2015- 2017

This query yet again functions similar to the previous. It uses a count on the sold books using BookId and then uses that count with a GROUP BY to get the data all summed up. The answer in this instance is “Positive Figures” lending a total of 10 copies.

SELECT BookTitle, COUNT(br.BookId) AS BookTotal

FROM book b

JOIN borrower br

ON b.BookId = br.BookId

WHERE BorrowDate Between ‘2015-01-01’ AND ‘2017-01-01’

Group BY br.BookId ORDER BY BookTotal DESC LIMIT 1

7: Top borrowed genres for clients born in years 1970-1980

Once again the two keys to this query operating successfully is the COUNT() function and the GROUP BY keyword. The top genre for that age bracket is the Science genre scoring a total of 24 borrows.

SELECT Genre, COUNT(br.BookId) AS BookTotal

FROM book b

JOIN borrower br

ON b.BookID = br.BookId

JOIN client c

ON br.ClientId = c.ClientId

WHERE c.ClientDoB BETWEEN '1970-01-01' AND '1980-01-01'

GROUP BY Genre

ORDER BY BookTotal DESC

8: Top 5 occupations that borrowed the most in 2016

I used the good old classic COUNT() to keep track of the total books sold and then used the GROUP BY operator on the occupations field. The most studious profession it seems in that year were students with a total of 32 books borrowed.

9: Average number of borrowed books by job title

This problem took a few minutes of thinking to grasp what was required but I knew that I would need to use the AVG function. Turns out that we just needed to swap the COUNT function out for the AVG function and then move a couple of the simpler things around.

SELECT Occupation, AVG(br.BookId) AS BookTotal

FROM client c

JOIN borrower br

ON c.ClientId = br.ClientId

GROUP BY Occupation

ORDER BY BookTotal DESC

10: Create a VIEW and display the titles that were borrowed by at least 20% of clients

I had issues getting my SQL to function properly on this problem. I understand that I would have to use some operators such as \* .02 to get the 20 percent and I am very comfortable using views. Unfortunately, the problem itself stumped me but the following code is what I got to.

CREATE VIEW practiceView AS

SELECT BookTitle, COUNT(br.BookId) AS BookTotal

FROM book b

JOIN borrower br

ON b.BookID = br.BookId

JOIN client c

ON br.ClientId = c.ClientId

WHERE BookTotal >= COUNT(c.ClientId) \* .2

GROUP BY BookTitle

ORDER BY BooKTotal

11: The top month of borrowers in 2017

Using the MONTH function, I was able to specifically select the months out of the BorrowDate table and then use the COUNT with the BookId to see which months got the most sales. In this example August, July, and October all had the highest number of books checked out at a total of 10.

SELECT MONTH(BorrowDate) AS Months, COUNT(br.BookId) AS BookTotals

FROM borrower br

WHERE BorrowDate BETWEEN '2017-01-01' AND '2018-01-01'

GROUP BY Months

ORDER BY BookTotals DESC

12: Average number of borrowers by age

This question is somewhat confusing to me but from what I understand I am being asked to find the total number of borrowers who are each of the same age. The following query does just that. I use simple mathematic operations to figure each of the clients names and then use an alias for that. Then I do the same thing and use a count so I can get the totals on each of the age groups. Then all I had to do was properly use my GROUP BY and ORDER BY commands.

SELECT '2022' - ClientDoB AS age, COUNT('2022' - ClientDoB) AS totalBorrowers

FROM client

GROUP BY age

ORDER BY totalBorrowers DESC

13: The oldest and the youngest clients of the library

This is fairly easy to figure out. The previous query I just displayed already got us the ages of each of the clients and it is easy to order them from youngest to oldest. All I needed to do was to use a UNION on the same table so I could combine the results from the lowest end of the table to the highest using LIMIT.

(SELECT '2022' - ClientDoB AS age

FROM client

ORDER BY age LIMIT 1)

UNION

(SELECT '2022' - ClientDoB AS age

FROM client

ORDER BY age DESC LIMIT 1)

14: First and last names of authors that wrote books in more than one genre

This one I really struggled to figure out. I think I got the first section of it and got the script to print all the authors with how many books they have published. Unfortunately, the provided tables seem to not have any authors with more than one genre so I cannot do much testing of my own.

SELECT AuthorFirstName, AuthorLastName, COUNT(BookAuthor) AS totalWorks

FROM author a

JOIN book b

ON a.AuthorId = b.BookAuthor

GROUP BY BookAuthor

ORDER BY totalWorks DESC