**CIS-310 Database Design**

**Small Group Activity #11**

**30 points**

Names of group members: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Get in touch with your group. See Groups folder on Blackboard.
2. Discuss and complete the assignment together via E-mail, Discussion Forum, Blackboard Collaborate Ultra, and/or MS Teams.
3. Choose a recorder to prepare the final copy (one per group) and submit it via the Blackboard Assignments/Small Group Activities folder to the instructor.
4. Be sure all group members' names are on final copy. Do not add names of your group classmates who did not participate in the assignment.

In this small group activity, you will be using the Henry Books database. Your task is to write two stored procedures with the cursor construct and three triggers. Discuss each of the five problems with your group, write a T-SQL (SQL Server) code for each problem, and run the code on the U of L installation of SQL server. Save the code in a single file in your account on J drive. Every group member should run the code for each problem and save the code in his/her account on J drive. Capture the T-SQL code from SQL Server and paste it after the description of each problem. Also capture the output generated by your code and paste the output which will be in the Messages tab and/or the Results tab.

After you paste the queries and the output they produced save this document as Word or pdf file named SGA11\_Groupxx, where xx stand for the group number and submit via Blackboard. See the Assignments/Small Group Activities/Small Group Activity 11 folder.

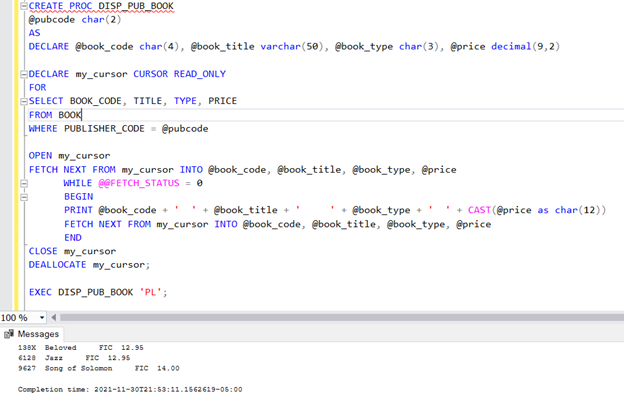
Watch the video placed in the Panopto & MS Teams Recordings content area for Week 15. It will be available on Mon, Nov 29. In the video I discuss several similar examples of the stored procedures with cursors and triggers based on the Premiere Products database. I also run the T-SQL code for them on SQL Server.

Once your compile stored procedures successfully, you should see them under objects in Programmability/Stored Procedures folder. Once you compile triggers successfully, you should see them as objects in the Tables/INVENTORY folder.

**Problem 1**

Write a stored procedure with the cursor construct named DISP\_PUB\_BOOK to retrieve and output the book code, book title, book type, price for every book whose publisher code is stored in variable @pubcode (an input formal parameter). Call the procedure for publisher ‘PL’.

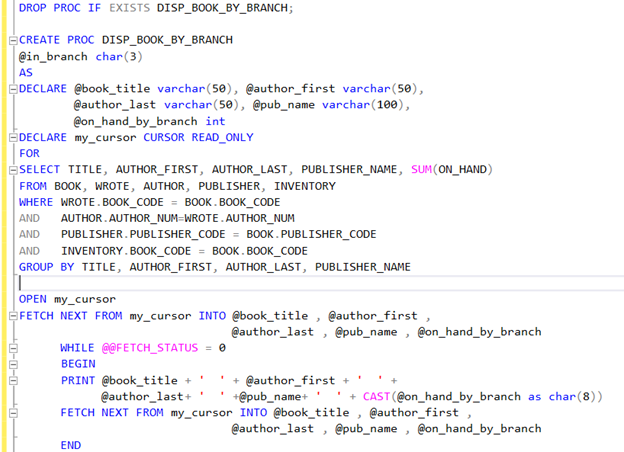
Copy/capture the T-SQL code for the procedure from SQL Server and the output it generated for publisher ‘PL’ and paste it below this line.



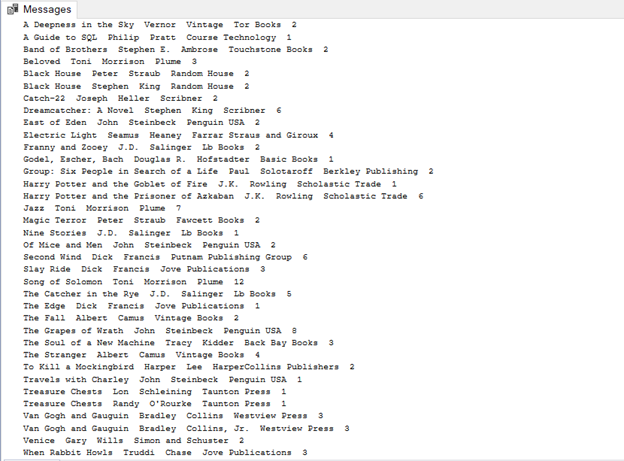
**Problem 2**

Write a stored procedure with the cursor construct named DISP\_BOOK\_BY\_BRANCH to retrieve and list the book title, the author first and last name, the publisher name, the amount on hand for all books stored in a specific branch number. The branch number, @in\_branch, is an input parameter through which the branch number will be supplied to the procedure. Call the procedure for branch number 1. (Note that in this problem you have to join several tables.)

Copy/capture the T-SQL code for the procedure from SQL Server and the output it generated for branch number 1 and paste it below this line.







**Problem 3**

Problem 3 contains three Parts: A, B, and C.

Assume the BOOK table contains a column called TOTAL\_ON\_HAND that represents the total units on hand in all branches for that book. Write the T-SQL code for the following three triggers described in Parts A, B, and C.

Before you start to write the code for the three triggers you **must** write and run the SQL code to add the TOTAL\_ON\_HAND column to the BOOK table, and then write and run the SQL code to populate the column with the data values. To reiterate, each row in the TOTAL\_ON\_HAND column in the BOOK table should contain the total units on hand in all branches for that book.

Insert the SQL code that you wrote to add column TOTAL\_ON\_HAND and populate it with the data values below this line.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Table

Description automatically generated with medium confidence

Graphical user interface

Description automatically generated

Note that the triggers will execute in background so you will not see their explicit execution like you do for stored procedures or views.

**Part A**. When inserting a row in the INVENTORY table, add the ON\_HAND value to the TOTAL\_ON\_HAND value for the appropriate book. To test the effect of the trigger step by step issue a series of the following simple commands.

--To see that the TOTAL\_ON\_HAND value for book 9931 is 2, execute

SELECT \*

FROM BOOK

WHERE BOOK\_CODE='9931';

--To see that the ON\_HAND value for book 9931 is 2 execute

SELECT \*

FROM INVENTORY

WHERE BOOK\_CODE='9931';

--Trigger is executed after this INSERT event in table INVENTORY

INSERT INTO INVENTORY

VALUES ('9931', 2, 5)

--To see that the row has been added for BOOK\_CODE='9931' and

--BRANCH\_NUM = 2, execute

SELECT \*

FROM INVENTORY

WHERE BOOK\_CODE='9931';

--The TOTAL\_ON\_HAND value is modified by on hand value = 5 inserted

--for book 9931 for branch 2. Now TOTAL\_ON\_HAND = 7. It was 2 before.

--To see it, execute

SELECT \*

FROM BOOK

WHERE BOOK\_CODE='9931';

Copy/capture the T-SQL code for the trigger from SQL Server and the output generated by all of the above SQL statements and paste it below this line.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Part B**. When updating a row in the INVENTORY table, add the difference between the new ON\_HAND vale and the old ON\_HAND value to the TOTAL\_ON\_HAND value for the appropriate book. To test the effect of the trigger step by step issue a series of the following simple commands.

--This update event causes execution of the trigger

UPDATE INVENTORY

SET ON\_HAND=8

WHERE BOOK\_CODE='9931' AND BRANCH\_NUM=2

--To see the ON HAND value updated for book 9931 in branch 2

SELECT \*

FROM INVENTORY

WHERE BOOK\_CODE='9931';

--To see the effect of the above UPDATE in table BOOK

SELECT \*

FROM BOOK

WHERE BOOK\_CODE='9931';

Copy/capture the T-SQL code for the trigger from SQL Server and the output generated by all of the above SQL statements and paste it below this line.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Part C**. When deleting a row in the INVENTORY table, subtract the add the difference between the new ON\_HAND vale and the old ON\_HAND value to the TOTAL\_ON\_HAND value for the appropriate book. To test the effect of the trigger step by step issue a series of the following simple commands.

--This delete event cause execution of the trigger

DELETE FROM INVENTORY

WHERE BOOK\_CODE='9931' AND BRANCH\_NUM=2;

--To see the row deleted. Now there is no row for book 9931 in branch 2.

SELECT \*

FROM INVENTORY

WHERE BOOK\_CODE='9931';

--To see the effect of the above delete in table BOOK.

--TOTAL\_ON\_HAND is 2 again.

SELECT \*

FROM BOOK

WHERE BOOK\_CODE='9931';

Copy/capture the T-SQL code for the trigger from SQL Server and the output generated by all of the above SQL statements and paste it below this line.

Graphical user interface, text, application

Description automatically generated

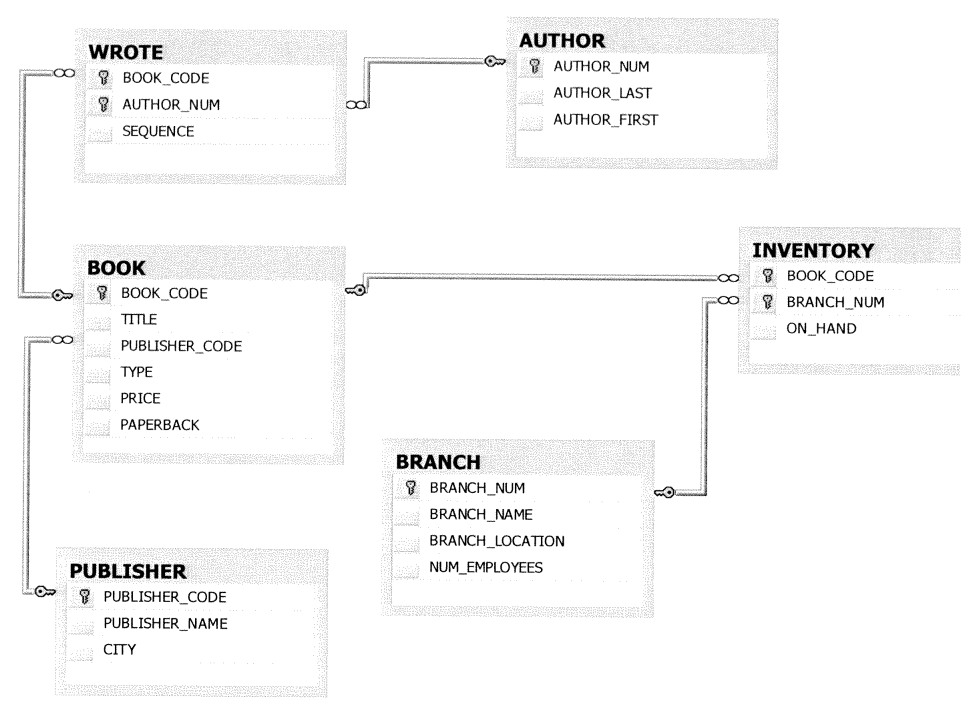
Graphical user interface, text, application, email

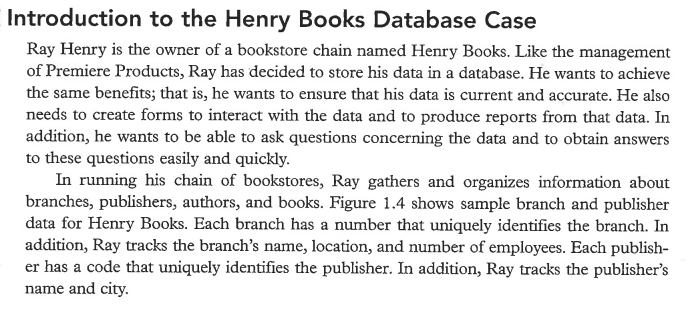
Description automatically generated

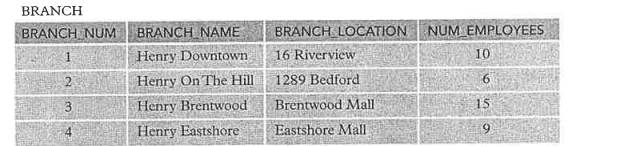
Graphical user interface, text, application, email

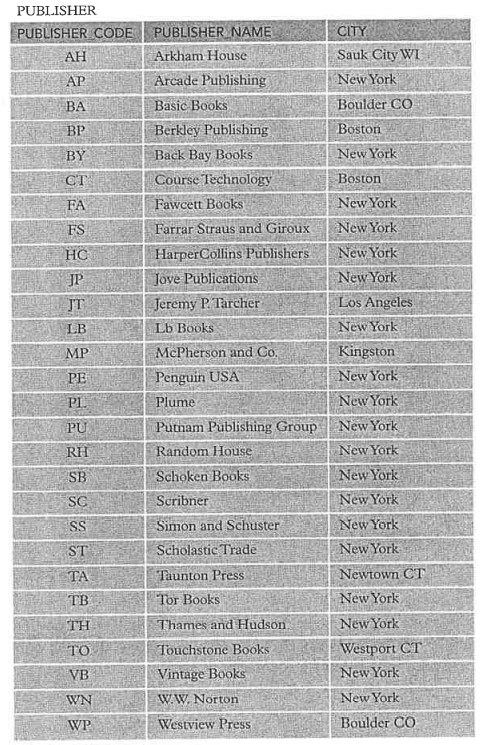
Description automatically generated

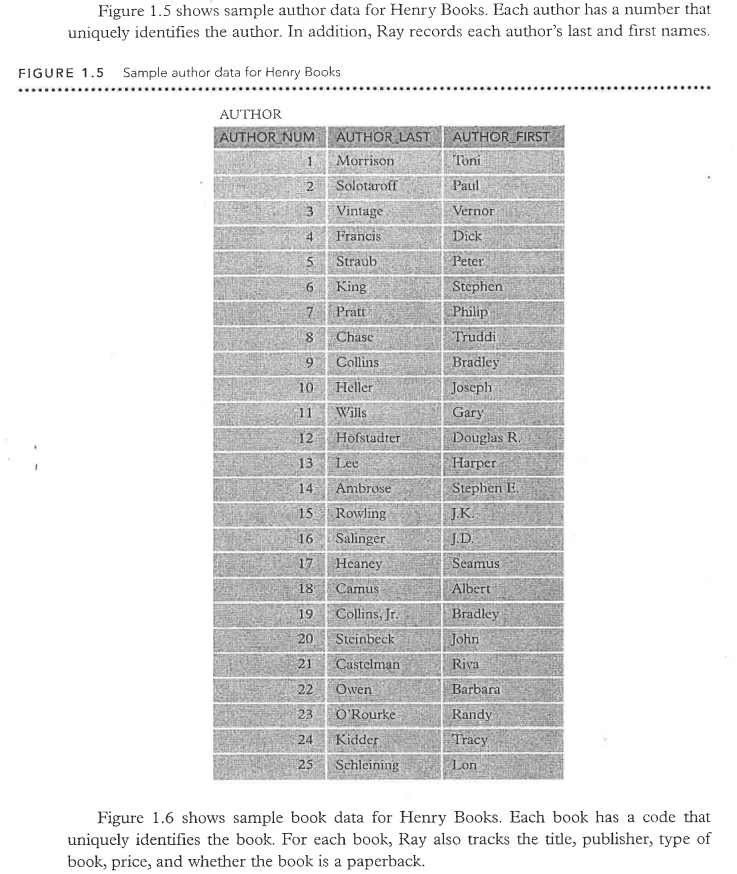
**The Henry Books Database ERD**

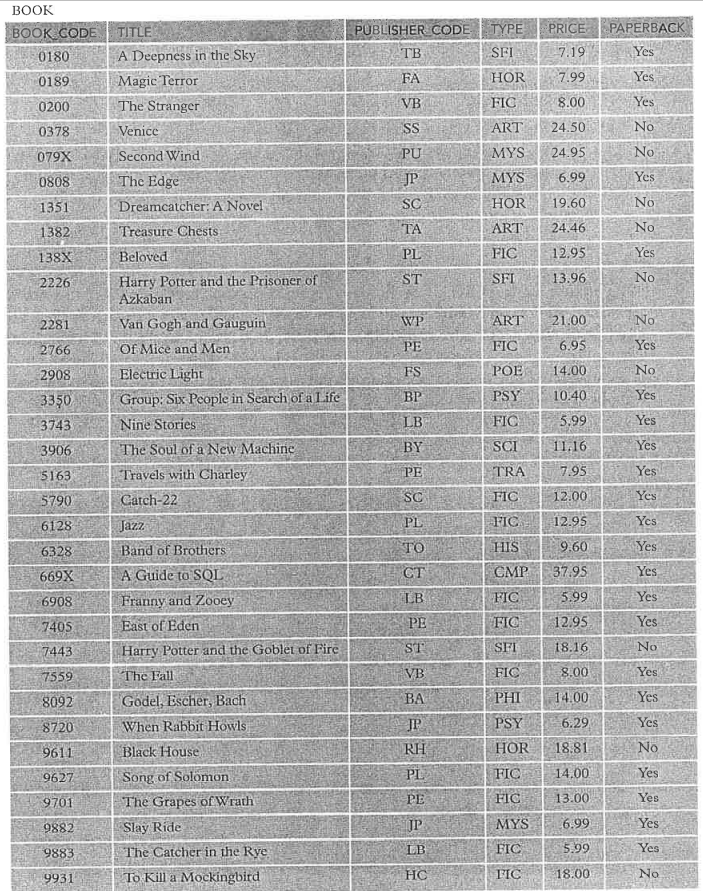


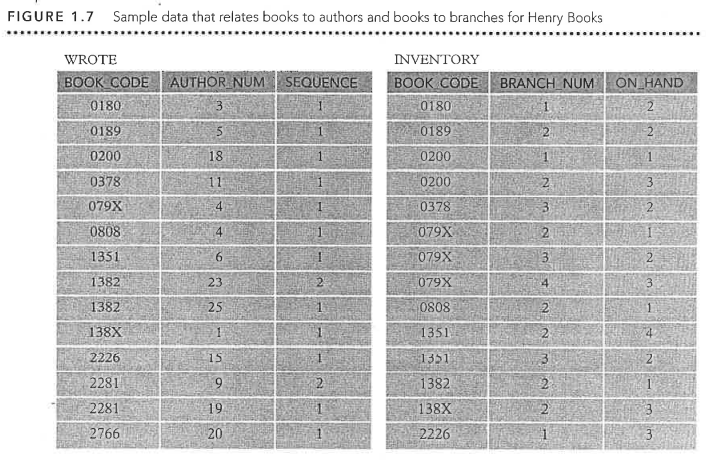










* 
* 