CIS-310 Database Design
Solution to Small Group Activity #10
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.

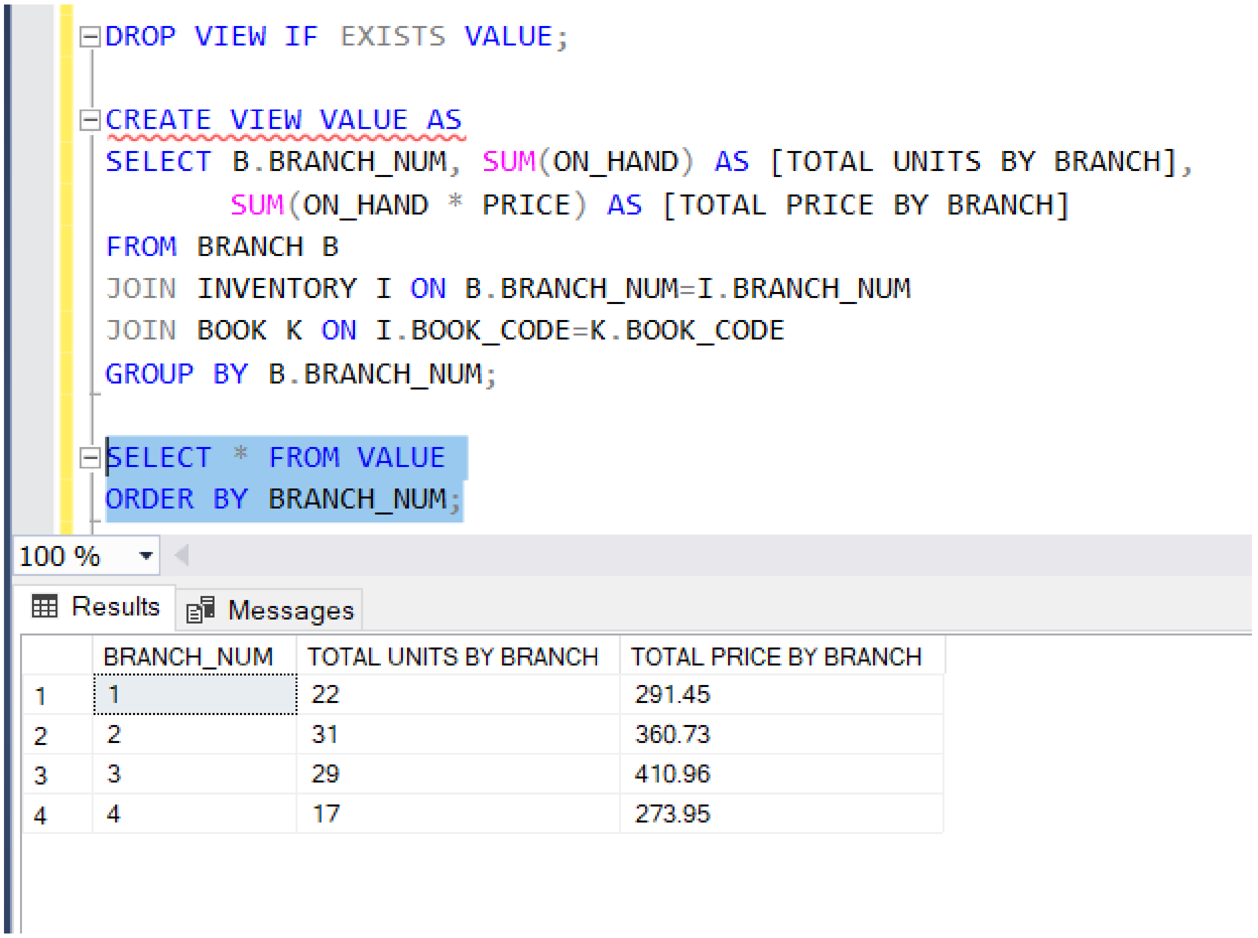
Discuss each of the six problems with your group, write an SQL code for each problem, and run the code on SQL server. Save the code in a single file in your account on J drive. Every group member should run the code for each problem and save the code in his/her account on J drive. Paste your code after the description of each problem and paste the output from the SELECT statement for Problems 1-4. For problems 5 & 6 show the output returned by SQL Server to the messages tab.

After you paste the queries and the output they produced save this document as Word or pdf file named SGA10_Groupxx, where xx stand for the group number and submit via Blackboard. See the Assignments/Small Group Activities/Small Group Activity 10 folder.

Watch the video(s) placed in the Panopto & MS Teams Recordings content area for the current week. In the videos I discuss similar examples based on the Premiere Products database run on SQL Server. The 6 problems below are based on the Henry Books database.

Problem 1

Create a view named VALUE. The view should display the branch number, the total number of books on hand for each branch and the total price of books for each branch. Display the total number of books on hand per branch as [TOTAL UNITS BY BRANCH] and the total price of books per branch as [TOTAL PRICE BY BRANCH]. Group and order the rows by branch number. Use the SELECT statement to show the output generated by the view. Copy/capture the SQL code from SQL Server and the output generated by the SELECT * FROM VALUE statement and paste it below this line.

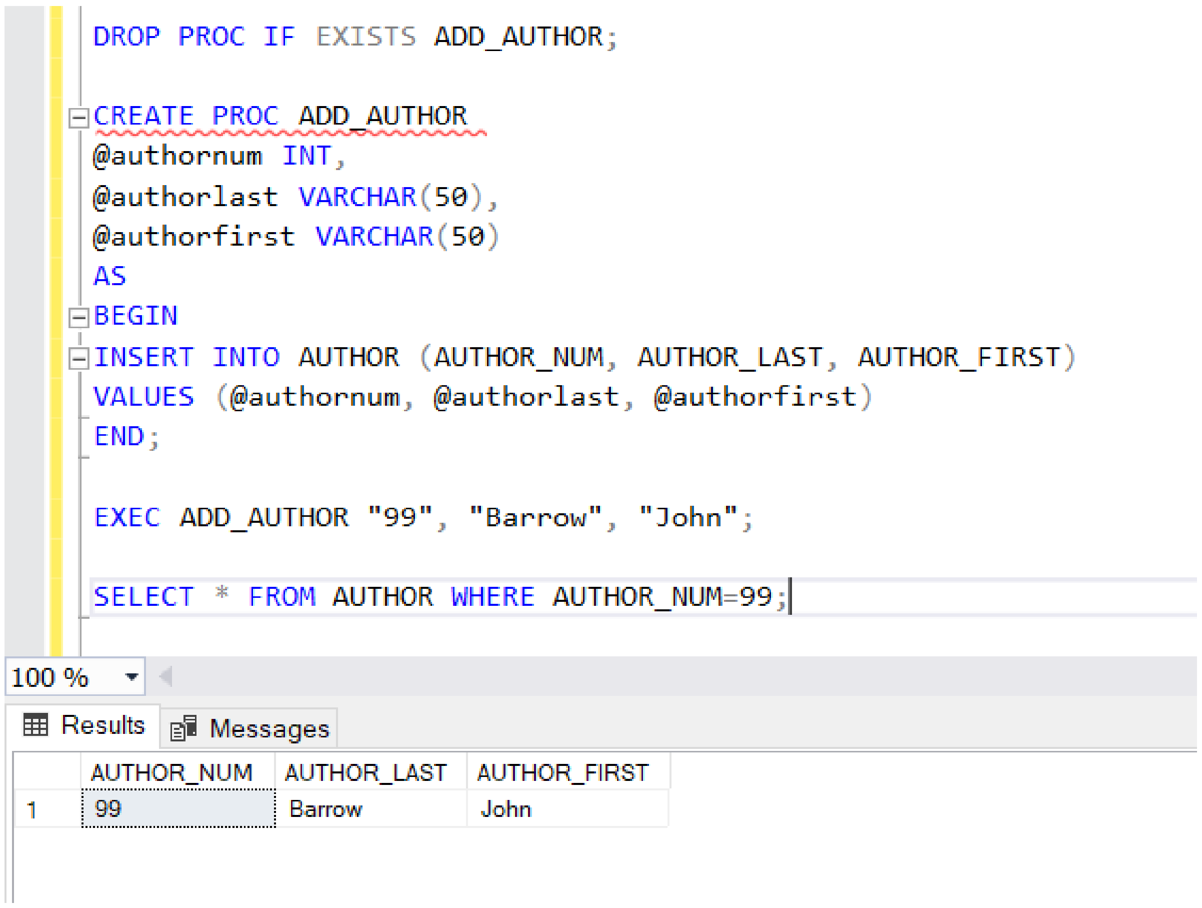


```
DROP VIEW IF EXISTS VALUE;

CREATE VIEW VALUE AS
SELECT B.BRANCH_NUM, SUM(ON_HAND) AS [TOTAL UNITS BY BRANCH],
       SUM(ON_HAND * PRICE) AS [TOTAL PRICE BY BRANCH]
FROM BRANCH B
JOIN INVENTORY I ON B.BRANCH_NUM=I.BRANCH_NUM
JOIN BOOK K ON I.BOOK_CODE=K.BOOK_CODE
GROUP BY B.BRANCH_NUM;

SELECT * FROM VALUE
ORDER BY BRANCH_NUM;
```

	BRANCH_NUM	TOTAL UNITS BY BRANCH	TOTAL PRICE BY BRANCH
1	1	22	291.45
2	2	31	360.73
3	3	29	410.96
4	4	17	273.95

Problem 2

Write a stored procedure to add a new author to the AUTHOR table. The procedure should accept 3 input parameters: the author number, author last name and author first name. Call/execute the procedure to add the author number 99, Barrow, John. Use the SELECT * FROM AUTHOR WHERE AUTHOR_NUM=99 statement to show that the author's name was added to the table. Copy/capture the SQL code for the stored procedure from SQL Server and the output generated by the above SELECT statement and paste it below this line.

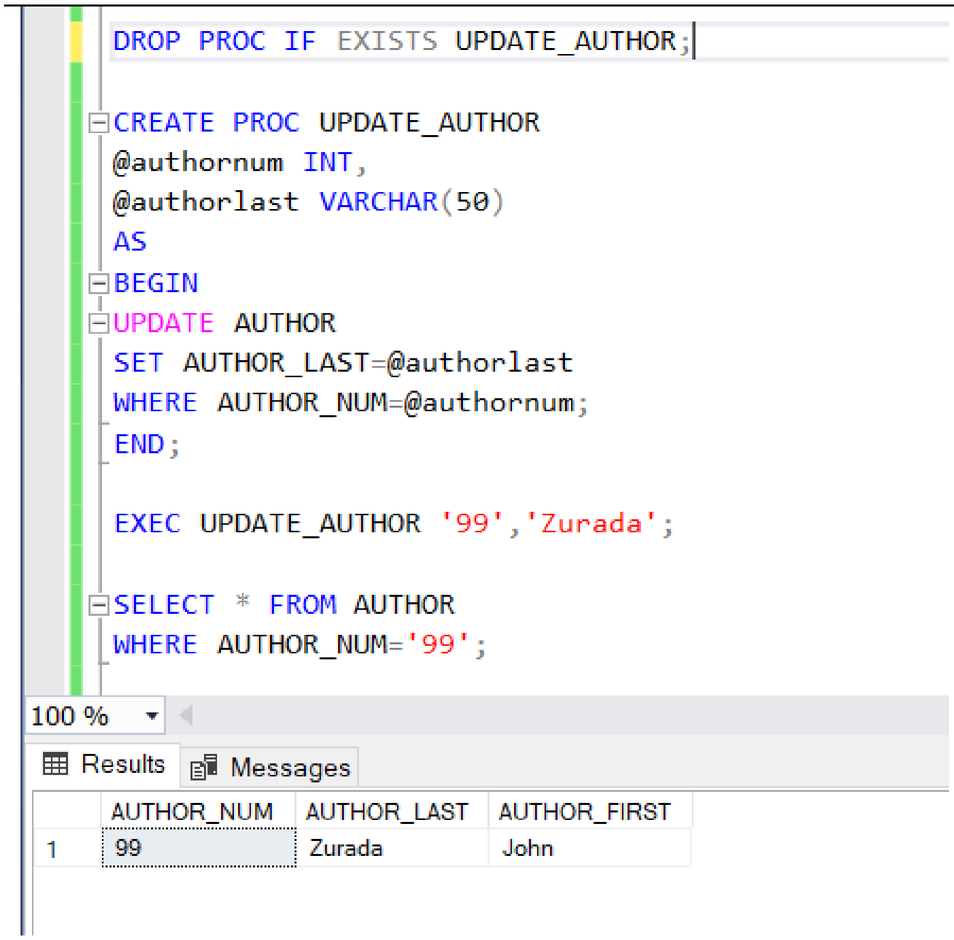


```sql
DROP PROC IF EXISTS ADD_AUTHOR;

CREATE PROC ADD_AUTHOR
@authornum INT,
@authorlast VARCHAR(50),
@authorfirst VARCHAR(50)
AS
BEGIN
INSERT INTO AUTHOR (AUTHOR_NUM, AUTHOR_LAST, AUTHOR_FIRST)
VALUES (@authornum, @authorlast, @authorfirst)
END;

EXEC ADD_AUTHOR "99", "Barrow", "John";

SELECT * FROM AUTHOR WHERE AUTHOR_NUM=99;
```

	AUTHOR_NUM	AUTHOR_LAST	AUTHOR_FIRST
1	99	Barrow	John

Problem 3

Write a stored procedure that will accept two input parameters: the author's number the author's last name. The procedure should change the author's last name to Zurada for that author number. Call/execute the procedure and pass to it the author number = 99 and the last name Zurada. Use the SELECT statement to show that that author's last name was changed in the table. Copy/capture the SQL code for the stored procedure from SQL Server and the output generated by the SELECT statement and paste it below this line.



```
DROP PROC IF EXISTS UPDATE_AUTHOR;

CREATE PROC UPDATE_AUTHOR
@authornum INT,
@authorlast VARCHAR(50)
AS
BEGIN
UPDATE AUTHOR
SET AUTHOR_LAST=@authorlast
WHERE AUTHOR_NUM=@authornum;
END;

EXEC UPDATE_AUTHOR '99','Zurada';

SELECT * FROM AUTHOR
WHERE AUTHOR_NUM='99';
```

100 %

Results | Messages

	AUTHOR_NUM	AUTHOR_LAST	AUTHOR_FIRST
1	99	Zurada	John

Problem 4

Write a stored procedure that will accept the author's number as an input parameter and will delete that author from the table. Call/execute the procedure for the author number 99 that you have recently added to the table. Use the SELECT statement to show that entire row representing that author was deleted from the table. Copy/capture the SQL code for the stored procedure from SQL Server and the output generated by the SELECT statement and paste it below this line.

Problem 5

Write a stored procedure that will calculate the total number of books on hand in the inventory. The procedure should store the total in the output parameter. Call/execute the procedure and print the total as “The total number of books is xx.”, where xx stands for the total. Use DECLARE, EXEC, and PRINT statements like in the examples posted and discussed in the recorded demo. Copy/capture the SQL code for the stored procedure from SQL Server and the output generated by SQL Server in the messages tab which shows the total.

Problem 6

Write a stored procedure that will calculate the total number of books in the inventory by the branch number. The procedure will accept the branch number as an input parameter. The procedure should have two output parameters to return the values for the total and the branch number. Call/execute the procedure and print the total for the selected branch number = 3. For example, your output may be “The total number of books for branch x is xx.”, where x stands for the branch number and xx for the total. Use DECLARE, EXEC, and PRINT statements like in the examples posted and discussed in the recorded demo. Copy/capture the SQL code for the stored procedure from SQL Server and the output generated by SQL Server in the messages tab which shows the total and the branch number.
