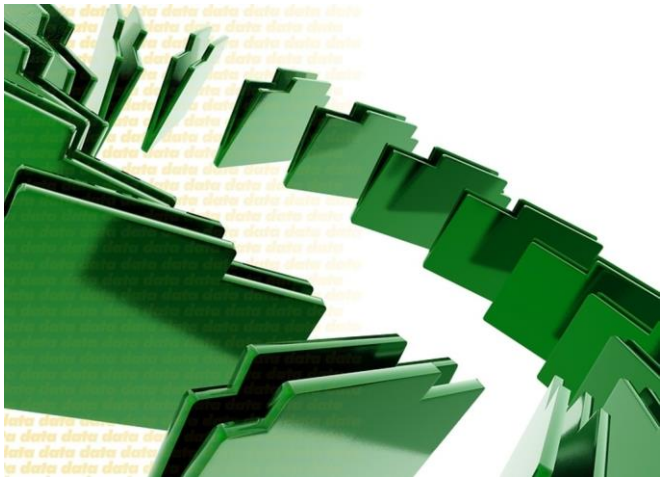

WEEK #9

ASSIGNMENT

2% Individual Assignment

Database with a View

Mark Morell

Database Management – Fall 2019

Assignment Type:

- Individual – Prepare and submit your results independently

Date Due:

- Thursday, November 7th by the end of the day

Instructions:

- Please submit your assignment electronically through eConestoga.
- Assignments should be submitted as Microsoft Word files using the course coversheet format. You **MUST** include your query as text in the Word document as well as a **FULL screenshot** of your SSMS screen (screen capture the entire application screen including the title bar through the bottom of the window). Multiple screenshots may be required.
- If you are using external sources (images, text, etc.) you must reference them as part of your assignment and not copy them as-is.
- Best practice is to research your answers and then write the response to the question in your own words.
- Please include the question number with your responses.

Late Assignment Penalty:

Days Late	Penalty %
1	5
2	10
3	20
4	40
5	60
6	80
7	100

Assignment Questions

Question #	Question	Score
1	Suppose that you have a large number of individual updates (i.e. that involve multiple SQL statements) that you need to make to a database in a short period of time. Which of the three transactions modes (auto-commit, explicit and implicit) would be best to use and why? You could use either explicit or implicit as both don't write the data to the database until the COMMIT is issued. Auto-commit will commit after each update which will take significantly more time to complete.	2
2	Write a series of SQL statements that demonstrate the concept of a "dirty read". Include comments in your SQL that describe why it demonstrates it. -- A dirty read involves retrieving data from the database that hasn't yet been -- committed so this would demonstrate what's required: BEGIN TRANSACTION UPDATE dbo.Person SET lastName = 'Whatever' WHERE number = '1252774' SELECT * from dbo.Person WHERE number = '1252774'	2
3	Using the SIS database, create a view named vStudentCourseList that produces the following information: • Student Number • Student's Full Name • Course Name • Course Credits *Hint: You will need to include 5 tables in your view including dbo.Student, dbo.CourseStudent, dbo.CourseOffering, dbo.Course and dbo.Person to get the information required here* CREATE VIEW dbo.vStudentCourseList AS SELECT p.number AS studentNumber, p.lastName + ', ' + p.firstName AS fullName, c.name, c.credits FROM dbo.Student AS s INNER JOIN dbo.CourseStudent AS cs ON cs.studentNumber = s.number INNER JOIN dbo.CourseOffering AS co ON co.id = cs.CourseOfferingId INNER JOIN dbo.Course AS c ON c.number = co.courseNumber INNER JOIN dbo.Person AS p ON p.number = s.number	4

4	Alter the vStudentCourseList created above to add the student's finalMark to the view and only include non-zero grades in the view. ALTER VIEW dbo.vStudentCourseList AS SELECT p.number AS studentNumber, p.lastName + ', ' + p.firstName AS fullName, c.name, c.credits, cs.finalMark FROM dbo.Student AS s INNER JOIN dbo.CourseStudent AS cs ON cs.studentNumber = s.number INNER JOIN dbo.CourseOffering AS co ON co.id = cs.CourseOfferingId INNER JOIN dbo.Course AS c ON c.number = co.courseNumber INNER JOIN dbo.Person AS p ON p.number = s.number WHERE cs.finalMark > 0	2
5	Using the view created above, write a query to retrieve the following information: • Student Number • Student's Full Name • Student's Average of all Courses Sort the information in the single query both by their average in descending order and then by student number in ascending order SELECT studentNumber, fullName, AVG(finalMark) AS finalAverage FROM dbo.vStudentCourseList GROUP BY studentNumber, fullName ORDER BY finalAverage DESC, studentNumber ASC -- Returns 90 rows	2

6	Using a Common Table Expression and <u>without</u> using any views, write a query to retrieve the exact same information as Question #5 above with the same columns and the same sort order. *Hint: Basically convert your view into a CTE and write a query around it.* WITH CTE_studentCourseList (studentNumber, fullName, finalMark) AS (SELECT p.number AS studentNumber, p.lastName + ', ' + p.firstName AS fullName, cs.finalMark FROM dbo.Student AS s INNER JOIN dbo.CourseStudent AS cs ON cs.studentNumber = s.number -- INNER JOIN dbo.CourseOffering AS co ON co.id = cs.CourseOfferingId -- INNER JOIN dbo.Course AS c ON c.number = co.courseNumber INNER JOIN dbo.Person AS p ON p.number = s.number WHERE cs.finalMark > 0) SELECT studentNumber, fullName, AVG(finalMark) AS avgFinalMark FROM CTE_studentCourseList GROUP BY studentNumber, fullName ORDER BY avgFinalMark DESC, studentNumber ASC -- Returns 90 rows	3
	Total	**15**