CSCI 3410 Database Final Project

For your final project you will build a database, populate it with data, and create two stored procedures to alter the database in a prescribed way. Please read the instructions below carefully. There is bonus points granted for additional tasks. The lab database will be reset so Lab 2 tables are cleared out for you to test your script against.

Important Information:

1. You cannot use a database design from class
2. Do not use a database design that was created by anyone other than yourself (this is plagiarism and violates the University's academic conduct)
3. Do not share your work with other students. You may seek the aid of another instructor at UNG for guidance. However, I'm here to help guide you along this project so come to me for help early and as often as you need.

Project Requirements:

1. Decide on an application, business, or game of your choice to design a database for. If you have trouble coming up with an idea you can reach out to me for some suggestions.
2. Assume the CREATE DATABASE command has already been performed.
3. Create 7-10 tables to store data for your chosen application, business, or game with the following specifications:
    a. Each table is at least in the Second Normal Form (2NF)
    b. Each table has a Primary Key of data type INT and IDENTITY set where the seed value is 0 or greater and increments by 1.
        i. I will allow varchar(x) or nvarchar(x) data types for primary keys in your "code" tables if you provide a valid reason in your SQL comments when creating the code table.
    c. All tables must relate to at least one other table (no island tables)
    d. There must be at least one many-to-many relationship (2 parent tables and 1 child table)
    e. All other relationships can be one-to-one or one-to-many as needed
    f. Foreign Key references do not need to cascade update or delete operations.
    g. All tables must use your schema in the labs database (example: JDOE1234.myTable)
    h. All tables must have at least 5 columns which includes any primary keys and foreign keys
    i. All columns must explicitly declare either NOT NULL or NULL
    j. At least one non-key column must use a DEFAULT constraint with a valid value
4. Using INSERT INTO statements, populate each table with at least 5-7 records
    a. All child tables must use SCOPE_IDENTITY() or @@IDENTITY to reference the parent's primary key when inserting data in to the table.
        i. Do not use explicit key values.
    b. Tip: Insert parent record then insert the child records before moving onto the next parent record. You will need to use a local variable to hold the parent's identity as any changes in the child tables will alter the function or global identity variables.
5. Create one stored procedure that will perform the following tasks:

   a. Updates the value of one column for a specific record in any of your tables. The update must perform a meaningful update and cannot simply set the value to NULL
   b. It must use at least one parameter to pass the identity of that table's record and it must only update one record.
   c. Example (increase the price of a product by 5%)
6. Create one stored procedure that will perform the following tasks:
   a. Deletes a record from a child table. The table you choose cannot have any dependent tables that reference it.
   b. It must use only one parameter to pass the identity of that table's record and it must only delete one record.
7. Entire project must be written in SQL and uses the template provide to you.
   a. Add your name to the Database Developer line, your Project Title, and Script Create Date in the header of the comments in the template.
   b. You must be able to run the entire script once to create all tables, procedures, insert data, and execute both procedures.
   c. Both procedures will only be ran once (so one update and one deletion)
   d. Create statements must be performed in the appropriate order (parents then children)
   e. Upload your SQL script to D2L before the deadline in the following formats {SQL, DOCX, PDF, TXT}
   f. NO LATE ASSIGNMENTS WILL BE ACCEPTED. NO EXCEPTIONS.
8. **BONUS POINTS**
   a. 1 bonus point will be granted for a CHECK constraint that is used on any non-key column that performs a valid check. It must not perform the work of any other constraint type such as but not limited to DEFAULT, NOT NULL, UNIQUE, etc…
   b. 3 bonus points will be granted for a database design with 10 tables
      i. If you're going to put in the effort to design 10 tables and put in at least 50 records you deserve some extra credit.
   c. 5 bonus points will be granted for a third stored procedure that performs the following tasks:
      i. Updates one record and one column's value
      ii. The update must also update on that same record the LastUpdated (datetime) and LastModifiedBy (varchar(20)). The table you update must allow for these two extra columns to keep track of the last person who updated the table.
      iii. Inserts a record into a logging table that records the date and time of entry, what record was altered from the system and allows you to record the username of the person that performed that operation. This table will keep a history of every update performed.
      iv. Each "logging" will insert only one record per operation
      v. The logging table will count towards your minimum required tables

Grading Rubric

Points	**Requirement**
20	Script runs once and creates all required objects, inserts all data, and executes all stored procedures (requirement 7)
20	All required number of tables created without errors (requirement 3)
5	All tables have primary keys (see requirement 3)
5	All integer based primary keys use the IDENTITY property (see requirement 3)
10	Includes one many-to-many relationship (requirement 3.d)
10	Inserted at least 5 records in each table (requirement 4)
10	Child tables use the identity function or global variable to reference the parent's identity (requirement 4)
10	Created a functioning Update Stored Procedure (requirement 5)
10	Created a functioning Deletion Stored Procedure (requirement 6)
+1	Bonus for a CHECK constraint (see requirement 8.a)
+3	Bonus for 10 tables (see requirement 8.b)
+5	Bonus for third Stored Procedure (see requirement 8.c)