**Homework 4 –** [***GitHub link SQL script***](https://github.com/MinhDung2209/homework_4_SQL-Programming-and-Relational-Data-Transformation/tree/master)

**Task 1 and 2:**

**1. Please create the Database structure from class**

**2. Create a Grades Table, that maps enrollment to letter grade, i.e., A+,A, B+ ...**

**A computer screen shot of a computer screen

AI-generated content may be incorrect.**

**A computer screen shot of a computer screen

AI-generated content may be incorrect.**

**Task 3: Populate the student table with 1000+ records using iteration. Please use the example from class as a reference but make sure that your data is well varied and not uniform**

**A computer screen shot of a computer screen

AI-generated content may be incorrect.**

**A computer screen shot of a computer screen

AI-generated content may be incorrect.**

**Task 4: Populate the classes table with 50 classes**

**A computer screen shot of a computer screen

AI-generated content may be incorrect.**

* SELECT \* FROM classes

**Task 5: Enroll every student in at least one class but make sure at least half the student population is enrolled in 2 or more classes**

**A computer screen shot of a computer

AI-generated content may be incorrect.**

* *STEP 1: Ensure every student has at least 1 class*
* *STEP 2: Enroll at least 60% of students with extra classes*
* *Show top 20 students with most classes*
* *Count number of classes per student*

**6. Assign grades for every class enrolled in**

**A computer screen shot of a computer screen

AI-generated content may be incorrect.**

* CountPerGrade
* SELECT \* FROM Grades Order by gradePoints desc;

**7. Write a stored procedure to compute the GPA**

**A computer screen shot of a computer screen

AI-generated content may be incorrect.**

**8. Write a stored procedure to compute the descriptive statistics of the student population, viz. Mean , Variance, Standard Deviation, Mode of the following Attributes**

**a.) G.P.A b.) HeightInCm c.) WeightInPounds**

**Hint: G.P.A is a derived attribute of every student, it depends on their grade in enrolled classes**

**A computer screen shot of a computer screen

AI-generated content may be incorrect.**

* EXEC ComputeDescriptiveStatistics@Gender = 'Male';
* EXEC ComputeDescriptiveStatistics;

**9. Create a VIEW to show how height distributes over Gender**

**A computer screen shot of a computer screen

AI-generated content may be incorrect.**

* SELECT \* FROM vHeightDistributionByGender;

**10. Write a store procedure to compute the distribution of G.P.A over Gender**

**A computer screen shot of a computer screen

AI-generated content may be incorrect.**

* EXEC ComputeGpaDistributionByGender;
* *STEP 1: Use a CTE to get the calculated GPA and gender for each student.*
* *STEP 2: Use another CTE to categorize each student's GPA into defined ranges (bins).*
* *STEP 3: Pivot the data using conditional aggregation to get counts for each gender*

**11. Describe what you learned from this homework**

This homework assignment provided a comprehensive, hands-on journey through the entire lifecycle of creating and managing a relational database system. Moving from foundational concepts to advanced analytical procedures, I have solidified my understanding of SQL not just as a query language, but as a powerful platform for data engineering and analysis.

First and foremost, I gained practical experience in database schema design and implementation. The initial tasks required me to translate a conceptual model into a physical one by creating a series of interconnected tables: students, classes, enrollments, and Grades. This process reinforced the importance of defining primary keys for uniqueness, foreign keys to enforce referential integrity, and appropriate data types to ensure data quality. Designing the Grades table with a UNIQUE constraint on enrollmentID was a key insight into building robust, error-resistant schemas.

The second major area of learning was in procedural SQL and large-scale data generation. Rather than performing trivial, manual INSERT statements, I wrote a script to programmatically generate over 1,000 student records. This required using WHILE loops, variables, and randomization functions like CHECKSUM(NEWID()). More importantly, I learned to create realistic, non-uniform data by implementing conditional logic, such as varying height and weight based on gender, and assigning specific departments based on a student's major. This elevated the task from simple data entry to a simulation of real-world data.

Subsequently, the assignment transitioned into advanced data analysis and reporting, which was the most challenging and rewarding part. I learned to encapsulate complex logic into reusable database objects. For instance, I developed a flexible ComputeGPA stored procedure with an optional parameter, allowing it to calculate GPA for either a single student or the entire population. The ComputeDescriptiveStatistics procedure was a deeper dive, compelling me to implement a custom solution for calculating the Mode—a function not natively available in SQL Server—in addition to using built-in aggregate functions like AVG, STDEV, and VAR.

Furthermore, I learned to differentiate between and apply two powerful reporting concepts: summary statistics and frequency distributions. For the latter, I created both a VIEW (vHeightDistributionByGender) and a Stored Procedure (ComputeGpaDistributionByGender). These tasks taught me the powerful technique of conditional aggregation (COUNT(CASE WHEN ...)), which allows for pivoting data to create clear, insightful comparison reports directly within the database engine. I also learned the practical use of a VIEW as a "virtual table" to simplify complex queries for end-users.

Finally, this entire process was wrapped in SQL best practices. I consistently used TRANSACTIONS to ensure atomicity, rolling back changes if errors occurred. I employed SET NOCOUNT ON for optimization and utilized Common Table Expressions (CTEs) and temporary tables to structure complex queries, making them more readable and maintainable.

So in conclusion, this homework has many lots of useful exercises that a little bit help me imagine how to bridge the gap between theoretical database concepts and practical, real-world applications. I have moved from simply querying data to engineering it, automating its processing, and deriving complex analytical insights directly from the database itself, though it still consumes errors, and optimization.

Thank you for reviewing the whole of my work!