Advance SQL Techniques

In this project, I have harnessed three distinct datasets from City of Chicago's Data Portal to create a comprehensive database in MySQL named "Chicago" (encoding as: utf8mb4_0900_ai_ci). The database consists of following three core datasets in ".sql" format:

- chicago_socioeconomic_data
- chicago_public_schools
- chicago_crime

Four exercises were included in this project:

1. Creating a Stored Procedure
    1) Query to list the school names, community names and average attendance for communities with a hardship index of 98
    2) Query to list all crimes that took place at a school. Include case number, crime type and community name.

2. Creating a View
   For privacy reasons, a view is needed to enable users to select just the school name and the icon fields from the CHICAGO_PUBLIC_SCHOOLS table. By providing a view, it ensures that users cannot see the actual scores given to a school, just the icon associated with their score. Defining new names for the view columns to obscure the use of scores and icons in the original table.

Column name in CHICAGO_PUBLIC_SCHOOLS	Column name in view
NAME_OF_SCHOOL	School_Name
Safety_Icon	Safety_Rating
Family_Involvement_Icon	Family_Rating
Environment_Icon	Environment_Rating
Instruction_Icon	Instruction_Rating
Leaders_Icon	Leaders_Rating
Teachers_Icon	Teachers_Rating

    1) Write and execute a SQL statement to create a view showing the columns listed in the above table, with new column names as shown in the second column.
    2) Write and execute a SQL statement that returns all of the columns from the view.

3) Write and execute a SQL statement that returns just the school name and leaders rating from the view.

3. Creating a Stored Procedure
   The icon fields are calculated based on the value in the corresponding score field. To make sure when a score field is updated; the icon field is updated too. To do this, a stored procedure is needed to receive the school id and a leaders score as input parameters, calculates the icon setting and updates the fields appropriately.
   1) Write the structure of a query to create or replace a stored procedure called UPDATE_LEADERS_SCORE that takes a in_School_ID parameter as an integer and a in_Leader_Score parameter as an integer.
   2) Inside the stored procedure, write a SQL statement to update the Leaders_Score field in the CHICAGO_PUBLIC_SCHOOLS table for the school identified by in_School_ID to the value in the in_Leader_Score parameter.
   3) Inside the stored procedure, write a SQL IF statement to update the Leaders_Icon field in the CHICAGO_PUBLIC_SCHOOLS table for the school identified by in_School_ID using the following information.

4. Using Transactions
   If someone calls the code with a score outside of the allowed range (0-99), the score would be updated with the invalid data and the icon would remain at its previous value. There are various ways to avoid this problem, and one is using a transaction.
   1) Update the stored procedure definition; add a generic ELSE clause to the IF statement that rolls back the current work if the score did not fit any of the preceding categories.
   2) Update the stored procedure definition; add a statement to commit the current unit of work at the end of the procedure.