**Question 1**

* Write and execute a SQL query to list the school names, community names and average attendance for communities with a hardship index of 98.

Texto

Descripción generada automáticamente con confianza media

Tabla

Descripción generada automáticamente

**Question 2**

* Write and execute a SQL query to list all crimes that took place at a school. Include case number, crime type and community name.

SELECT C.CASE\_NUMBER, C.PRIMARY\_TYPE, C.LOCATION\_DESCRIPTION, C.COMMUNITY\_AREA\_NUMBER, SO.COMMUNITY\_AREA\_NAME

FROM chicago\_crime AS C

LEFT JOIN chicago\_socioeconomic\_data AS SO

ON C.COMMUNITY\_AREA\_NUMBER= SO.COMMUNITY\_AREA\_NUMBER

WHERE C.LOCATION\_DESCRIPTION LIKE '%SCHOOL%'

Tabla

Descripción generada automáticamente

**Exercise 2: Creating a View**

For privacy reasons, you have been asked to create a view that enables users to select just the school name and the icon fields from the CHICAGO\_PUBLIC\_SCHOOLS table. By providing a view, you can ensure that users cannot see the actual scores given to a school, just the icon associated with their score. You should define new names for the view columns to obscure the use of scores and icons in the original table.

**Question 1**

* Write and execute a SQL statement to create a view showing the columns listed in the following table, with new column names as shown in the second column.

Texto

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

**Exercise 3: Creating a Stored Procedure**

The icon fields are calculated based on the value in the corresponding score field. You need to make sure that when a score field is updated, the icon field is updated too. To do this, you will write a stored procedure that receives the school id and a leaders score as input parameters, calculates the icon setting and updates the fields appropriately.

**Question 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.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**Question 2**

* Inside your 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.



Imagen que contiene Diagrama

Descripción generada automáticamente

**Question 3**

* Inside your 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

DELIMITER $$

CREATE PROCEDURE UPDATE\_Leaders\_Icon(IN in\_School\_ID INT , IN in\_Leader\_Score INT )

BEGIN

IF in\_Leader\_Score >= 0 AND in\_Leader\_Score <= 19 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = "Very weak"

WHERE School\_ID = in\_School\_ID ;

ELSEIF in\_Leader\_Score <= 39 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = "Weak"

WHERE School\_ID = in\_School\_ID ;

ELSEIF in\_Leader\_Score <= 59 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = "Average"

WHERE School\_ID = in\_School\_ID ;

ELSEIF in\_Leader\_Score <= 79 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = "Strong"

WHERE School\_ID = in\_School\_ID ;

ELSEIF in\_Leader\_Score <= 99 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = "Very weak"

WHERE School\_ID = in\_School\_ID ;

END IF;

END $$

DELIMITER ;

**Question 4**

* Run your code to create the stored procedure.

**Take a screenshot showing the SQL query and its results.**

* Write a query to call the stored procedure, passing a valid school ID and a leader score of 50, to check that the procedure works as expected.

**Exercise 4: Using Transactions**

You realise that if someone calls your code with a score outside of the allowed range (0-99), then the score will be updated with the invalid data and the icon will remain at its previous value. There are various ways to avoid this problem, one of which is using a transaction.

**Question 1**

* Update your 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.

**Question 2**

* Update your stored procedure definition again. Add a statement to commit the current unit of work at the end of the procedure

DELIMITER $$

CREATE PROCEDURE UPDATE\_Leaders\_Icon(

IN in\_School\_ID INT,

IN in\_Leader\_Score INT

)

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

RESIGNAL;

END;

START TRANSACTION;

IF in\_Leader\_Score >= 0 AND in\_Leader\_Score <= 19 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = 'Very weak'

WHERE School\_ID = in\_School\_ID;

ELSEIF in\_Leader\_Score >= 20 AND in\_Leader\_Score <= 39 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = 'Weak'

WHERE School\_ID = in\_School\_ID;

ELSEIF in\_Leader\_Score >= 40 AND in\_Leader\_Score <= 59 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = 'Average'

WHERE School\_ID = in\_School\_ID;

ELSEIF in\_Leader\_Score >= 60 AND in\_Leader\_Score <= 79 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = 'Strong'

WHERE School\_ID = in\_School\_ID;

ELSEIF in\_Leader\_Score >= 80 AND in\_Leader\_Score <= 99 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = 'Very strong'

WHERE School\_ID = in\_School\_ID;

ELSE

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Invalid value for in\_Leader\_Score. Value must be between 0 and 99.';

END IF;

COMMIT;

END $$

DELIMITER ;