**CHICAGO INSIGHTS: CRIME, EDUCATION & SOCIOECONOMICS ANALYSIS Using SQL**

**# Joins**

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

**Ans:-**

SELECT cps.NAME\_OF\_SCHOOL AS school\_name,

csd.COMMUNITY\_AREA\_NAME AS community\_name,

cps.AVERAGE\_STUDENT\_ATTENDANCE FROM chicago\_public\_schools

cps JOIN chicago\_socioeconomic\_data csd ON

cps.COMMUNITY\_AREA\_NUMBER = csd.COMMUNITY\_AREA\_NUMBER WHERE

csd.HARDSHIP\_INDEX = 98

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

**Ans:-**

SELECT

cc.CASE\_NUMBER,

cc.PRIMARY\_TYPE AS crime\_type,

csd.COMMUNITY\_AREA\_NAME AS community\_name

FROM

chicago\_crime cc

JOIN

chicago\_socioeconomic\_data csd

ON

cc.COMMUNITY\_AREA\_NUMBER = csd.COMMUNITY\_AREA\_NUMBER

WHERE

cc.LOCATION\_DESCRIPTION LIKE '%SCHOOL%';

**# View**

**Q.** 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.

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

**Ans:-**

CREATE VIEW School\_Survey\_View AS

SELECT

NAME\_OF\_SCHOOL AS School\_Name,

Safety\_Icon AS Safety\_Rating,

Family\_Involvement\_Icon AS Family\_Rating,

Environment\_Icon AS Environment\_Rating,

Instruction\_Icon AS Instruction\_Rating,

Leaders\_Icon AS Leaders\_Rating,

Teachers\_Icon AS Teachers\_Rating

FROM

chicago\_public\_schools;

**Q.** Write and execute a SQL statement that returns all of the columns from the view.

**Ans:-**

SELECT \*

FROM School\_Survey\_View;

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

**Ans:-**

SELECT

School\_Name,

Leaders\_Rating

FROM

School\_Survey\_View;

**# Creating a Stored Procedure**

**Q.** 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.

**Ans:-**

DELIMITER //

CREATE PROCEDURE UPDATE\_LEADERS\_SCORE(

IN in\_School\_ID INT,

IN in\_Leader\_Score INT

)

BEGIN

UPDATE chicago\_public\_schools

SET Leaders\_Score = in\_Leader\_Score

WHERE School\_ID = in\_School\_ID;

END //

DELIMITER ;

**Q.** 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.

**Ans:-**

DELIMITER //

CREATE PROCEDURE UPDATE\_LEADERS\_SCORE(

IN in\_School\_ID INT,

IN in\_Leader\_Score INT

)

BEGIN

UPDATE chicago\_public\_schools

SET Leaders\_Score = in\_Leader\_Score

WHERE School\_ID = in\_School\_ID;

END //

DELIMITER ;

**Q.** 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.

Score lower limit Score upper limit Icon

80 99 Very strong

60 79 Strong

40 59 Average

20 39 Weak

0 19 Very weak

**Ans:-**

DELIMITER //

CREATE PROCEDURE UPDATE\_LEADERS\_SCORE(

IN in\_School\_ID INT,

IN in\_Leader\_Score INT

)

BEGIN

IF in\_Leader\_Score BETWEEN 80 AND 99 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = 'Very strong'

WHERE School\_ID = in\_School\_ID;

ELSEIF in\_Leader\_Score BETWEEN 60 AND 79 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = 'Strong'

WHERE School\_ID = in\_School\_ID;

ELSEIF in\_Leader\_Score BETWEEN 40 AND 59 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = 'Average'

WHERE School\_ID = in\_School\_ID;

ELSEIF in\_Leader\_Score BETWEEN 20 AND 39 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = 'Weak'

WHERE School\_ID = in\_School\_ID;

ELSEIF in\_Leader\_Score BETWEEN 0 AND 19 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = 'Very weak'

WHERE School\_ID = in\_School\_ID;

END IF;

END //

DELIMITER ;

**Q.** Run your code to create the stored procedure.

**Ans:-**

DROP PROCEDURE IF EXISTS UPDATE\_LEADERS\_SCORE;

DELIMITER //

CREATE PROCEDURE UPDATE\_LEADERS\_SCORE(

IN in\_School\_ID INT,

IN in\_Leader\_Score INT

)

BEGIN

IF in\_Leader\_Score BETWEEN 80 AND 99 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = 'Very strong'

WHERE School\_ID = in\_School\_ID;

ELSEIF in\_Leader\_Score BETWEEN 60 AND 79 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = 'Strong'

WHERE School\_ID = in\_School\_ID;

ELSEIF in\_Leader\_Score BETWEEN 40 AND 59 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = 'Average'

WHERE School\_ID = in\_School\_ID;

ELSEIF in\_Leader\_Score BETWEEN 20 AND 39 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = 'Weak'

WHERE School\_ID = in\_School\_ID;

ELSEIF in\_Leader\_Score BETWEEN 0 AND 19 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = 'Very weak'

WHERE School\_ID = in\_School\_ID;

END IF;

END //

DELIMITER ;

**Q.** And to update a school:

**Ans:-**

CALL UPDATE\_LEADERS\_SCORE(1010, 73);

**Q.** 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.

**Ans:-**

CALL UPDATE\_LEADERS\_SCORE(1010, 50);

**# Using Transactions**

**Q.** 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.

**Ans:-**

DROP PROCEDURE IF EXISTS UPDATE\_LEADERS\_SCORE;

DELIMITER //

CREATE PROCEDURE UPDATE\_LEADERS\_SCORE(

IN in\_School\_ID INT,

IN in\_Leader\_Score INT

)

BEGIN

START TRANSACTION;

IF in\_Leader\_Score BETWEEN 80 AND 99 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = 'Very strong'

WHERE School\_ID = in\_School\_ID;

ELSEIF in\_Leader\_Score BETWEEN 60 AND 79 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = 'Strong'

WHERE School\_ID = in\_School\_ID;

ELSEIF in\_Leader\_Score BETWEEN 40 AND 59 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = 'Average'

WHERE School\_ID = in\_School\_ID;

ELSEIF in\_Leader\_Score BETWEEN 20 AND 39 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = 'Weak'

WHERE School\_ID = in\_School\_ID;

ELSEIF in\_Leader\_Score BETWEEN 0 AND 19 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = 'Very weak'

WHERE School\_ID = in\_School\_ID;

ELSE

-- If score is invalid, rollback

ROLLBACK;

END IF;

COMMIT;

END //

DELIMITER ;

CALL UPDATE\_LEADERS\_SCORE(1010, 75);

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

**Ans:-**

DROP PROCEDURE IF EXISTS UPDATE\_LEADERS\_SCORE;

DELIMITER //

CREATE PROCEDURE UPDATE\_LEADERS\_SCORE(

IN in\_School\_ID INT,

IN in\_Leader\_Score INT

)

BEGIN

START TRANSACTION;

IF in\_Leader\_Score BETWEEN 80 AND 99 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = 'Very strong'

WHERE School\_ID = in\_School\_ID;

COMMIT;

ELSEIF in\_Leader\_Score BETWEEN 60 AND 79 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = 'Strong'

WHERE School\_ID = in\_School\_ID;

COMMIT;

ELSEIF in\_Leader\_Score BETWEEN 40 AND 59 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = 'Average'

WHERE School\_ID = in\_School\_ID;

COMMIT;

ELSEIF in\_Leader\_Score BETWEEN 20 AND 39 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = 'Weak'

WHERE School\_ID = in\_School\_ID;

COMMIT;

ELSEIF in\_Leader\_Score BETWEEN 0 AND 19 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = 'Very weak'

WHERE School\_ID = in\_School\_ID;

COMMIT;

ELSE

ROLLBACK;

END IF;

END //

DELIMITER ;

CALL UPDATE\_LEADERS\_SCORE(1010, 55);

**Q.** Run your code to replace the stored procedure.

**Ans:-**

DROP PROCEDURE IF EXISTS UPDATE\_LEADERS\_SCORE;

DELIMITER //

CREATE PROCEDURE UPDATE\_LEADERS\_SCORE(

IN in\_School\_ID INT,

IN in\_Leader\_Score INT

)

BEGIN

START TRANSACTION;

IF in\_Leader\_Score BETWEEN 80 AND 99 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = 'Very strong'

WHERE School\_ID = in\_School\_ID;

COMMIT;

ELSEIF in\_Leader\_Score BETWEEN 60 AND 79 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = 'Strong'

WHERE School\_ID = in\_School\_ID;

COMMIT;

ELSEIF in\_Leader\_Score BETWEEN 40 AND 59 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = 'Average'

WHERE School\_ID = in\_School\_ID;

COMMIT;

ELSEIF in\_Leader\_Score BETWEEN 20 AND 39 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = 'Weak'

WHERE School\_ID = in\_School\_ID;

COMMIT;

ELSEIF in\_Leader\_Score BETWEEN 0 AND 19 THEN

UPDATE chicago\_public\_schools

SET Leaders\_Icon = 'Very weak'

WHERE School\_ID = in\_School\_ID;

COMMIT;

ELSE

ROLLBACK;

END IF;

END //

DELIMITER ;

**Q.** Write and run one query to check that the updated stored procedure works as expected when you use a valid score of 38.

**Ans:-**

CALL UPDATE\_LEADERS\_SCORE(1010, 38);

To check if it updated correctly, run:

SELECT

School\_ID,

Leaders\_Score,

Leaders\_Icon

FROM

chicago\_public\_schools

WHERE

School\_ID = 1010;

**Q.** Write and run another query to check that the updated stored procedure works as expected when you use an invalid score of 101.

**Ans:-**

CALL UPDATE\_LEADERS\_SCORE(1010, 101);

To Verify:

SELECT

School\_ID,

Leaders\_Score,

Leaders\_Icon

FROM

chicago\_public\_schools

WHERE

School\_ID = 1010;

**Full Flow:**

**-- Call procedure with invalid score**

CALL UPDATE\_LEADERS\_SCORE(1010, 101);

**-- Check if anything changed**

SELECT

School\_ID,

Leaders\_Score,

Leaders\_Icon

FROM

chicago\_public\_schools

WHERE

School\_ID = 1010;