Final Project: Advanced SQL Technique

Scenario

You have to analyse the following datasets for the city of Chicago, as available on the Chicago City data portal.

- Socioeconomic indicators in Chicago
- Chicago public schools
- Chicago crime data

Based on the information available in the different tables, you have to run specific queries using Advanced SQL techniques that generate the required result sets.

Objectives

After completing this lab, you will be able to:

Use joins to query data from multiple tables

Create and query views

Write and run stored procedures

Use transactions

Software Used in this Lab

In this lab, you will use MySQL. MySQL is a Relational Database Management System (RDBMS) designed to efficiently store, manipulate, and retrieve data.

Mysql_learners database has been used in this lab.

Exercise 1: Using Joins

You have been asked to produce some reports about the communities and crimes in the Chicago area. You will need to use SQL join queries to access the data stored across multiple tables.

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.

```sql
# The list of school names, community names and average attendance for communities with a hardship index of 98.

SELECT
    cps.NAME_OF_SCHOOL,
    csd.COMMUNITY_AREA_NAME,
    cps.AVERAGE_STUDENT_ATTENDANCE
FROM
    chicago_public_schools cps
LEFT JOIN
    chicago_socioeconomic_data csd
ON
    cps.COMMUNITY_AREA_NUMBER = csd.COMMUNITY_AREA_NUMBER
WHERE
    csd.HARDSHIP_INDEX = 98;
```

NAME_OF_SCHOOL	COMMUNITY_AREA_NAME	AVERAGE_STUDENT_ATTENDANCE
George Washington Carver Military Academy High Sch...	Riverdale	91.60%
George Washington Carver Primary School	Riverdale	90.90%
Ira F Aldridge Elementary School	Riverdale	92.90%
William E B Dubois Elementary School	Riverdale	93.30%

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.

```sql
# List of all crimes that took place at a school. This includes case number, crime type and community name.

SELECT
    cc.CASE_NUMBER,
    cc.PRIMARY_TYPE AS CRIME_TYPE,
    csd.COMMUNITY_AREA_NAME
FROM
    chicago_crime cc
LEFT JOIN
    chicago_socioeconomic_data csd
ON
    cc.COMMUNITY_AREA_NUMBER = csd.COMMUNITY_AREA_NUMBER
WHERE
    cc.LOCATION_DESCRIPTION LIKE '%School%';
```

CASE_NUMBER	CRIME_TYPE	COMMUNITY_AREA_NAME
HL353697	BATTERY	South Shore
HL725506	BATTERY	Lincoln Square
HP716225	BATTERY	Douglas
HH639427	BATTERY	Austin
JA460432	BATTERY	Ashburn
HS200939	CRIMINAL DAMAGE	Austin
HK577020	NARCOTICS	Rogers Park
HS305355	NARCOTICS	Brighton Park
HT315369	ASSAULT	East Garfield Park
HR585012	CRIMINAL TRESPASS	Ashburn
HH292682	PUBLIC PEACE VIOLATION	CHICAGO
G635735	PUBLIC PEACE VIOLATION	CHICAGO

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.

Column name in CHICAGO_PUBLIC_SCHOOLS	**Column name in Icon**
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

```
CREATE VIEW school_rating_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;
```

```
CREATE VIEW school_rating_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;
```

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

```sql
# All columns from the view

SELECT *
FROM school_rating_view;
```

School_Name	Safety_Rating	Family_Rating	Environment_Rating	Instruction_Rating	Leaders_Rating	Teachers_Rating
Abraham Lincoln Elementary School	Very Strong	Very Strong	Strong	Strong	Weak	Strong
Adam Clayton Powell Paideia Community Academy Elem...	Average	Strong	Strong	Very Strong	Weak	Strong
Adlai E Stevenson Elementary School	Strong	NDA	Average	Weak	Weak	NDA
Agustin Lara Elementary Academy	Average	Average	Average	Weak	Weak	Average
Air Force Academy High School	Average	Strong	Strong	Average	Weak	Average
Albany Park Multicultural Academy	Strong	Weak	Strong	Strong	Weak	Average

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

```
# The school name and leaders rating
from the view

SELECT School_Name, Leaders_Rating
FROM school_rating_view;
```

School_Name	Leaders_Rating
Abraham Lincoln Elementary School	Weak
Adam Clayton Powell Paideia Community Academy Elem...	Weak
Adlai E Stevenson Elementary School	Weak
Agustin Lara Elementary Academy	Weak
Air Force Academy High School	Weak
Albany Park Multicultural Academy	Weak
Albert G Lane Technical High School	Weak
Albert R Sabin Elementary Magnet School	Weak
Alcott High School for the Humanities	Weak
Alessandro Volta Elementary School	Weak
Alexander Graham Bell Elementary School	Weak
Alexander Graham Elementary School	Weak
Alexander Hamilton Elementary School	Weak
Alexander von Humboldt Elementary School	Weak
Alex Haley Elementary Academy	Weak
Alfred David Kohn Elementary School	Weak
Alfred Nobel Elementary School	Weak
Alice L Barnard Computer Math & Science Center Ele...	Weak

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.

```
# Creating or Replacing 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.

CREATE PROCEDURE UPDATE_LEADERS_SCORE(
    IN in_School_ID INT,
    IN in_Leader_Score INT
)
BEGIN

END;
```

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0144 seconds.)

```
# Creating or Replacing 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. CREATE
PROCEDURE UPDATE_LEADERS_SCORE( IN in_School_ID INT, IN
in_Leader_Score INT ) BEGIN END;
```

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.

```
# Updating 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.

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 $$
```

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0164 seconds.)

```
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;
```

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.

Score lower limit	Score upper limit	Icon
80	99	Very strong
60	79	Strong
40	59	Average
20	39	Weak
0	19	Very weak

```
DELIMITER $$

CREATE PROCEDURE UPDATE_LEADERS_SCORE(
    IN in_School_ID INT,
    IN in_Leader_Score INT
)
BEGIN
    DECLARE New_Icon VARCHAR(50);

    # Determine the Leaders_Icon based on the Leader Score range
    IF in_Leader_Score >= 80 AND in_Leader_Score <= 99 THEN
        SET New_Icon = 'Very strong';
    ELSEIF in_Leader_Score >= 60 AND in_Leader_Score <= 79 THEN
        SET New_Icon = 'Strong';
    ELSEIF in_Leader_Score >= 40 AND in_Leader_Score <= 59 THEN
        SET New_Icon = 'Average';
    ELSEIF in_Leader_Score >= 20 AND in_Leader_Score <= 39 THEN
        SET New_Icon = 'Weak';
    ELSEIF in_Leader_Score >= 0 AND in_Leader_Score <= 19 THEN
        SET New_Icon = 'Very weak';
    END IF;

    # Update the Leaders_Icon for the given School_ID
    UPDATE chicago_public_schools
    SET Leaders_Icon = New_Icon
    WHERE School_ID = in_School_ID;

END $$

DELIMITER ;
```

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0073 seconds.)

```
VARCHAR(50); # Determine the Leaders_Icon based on the Leader Score range IF in_Leader_Score >= 80 AND
in_Leader_Score <= 99 THEN SET New_Icon = 'Very strong'; ELSEIF in_Leader_Score >= 60 AND in_Leader_Score <= 79 THEN
SET New_Icon = 'Strong'; ELSEIF in_Leader_Score >= 40 AND in_Leader_Score <= 59 THEN SET New_Icon = 'Average'; ELSEIF
in_Leader_Score >= 20 AND in_Leader_Score <= 39 THEN SET New_Icon = 'Weak'; ELSEIF in_Leader_Score >= 0 AND
in_Leader_Score <= 19 THEN SET New_Icon = 'Very weak'; END IF; # Update the Leaders_Icon for the given School_ID
UPDATE chicago_public_schools SET Leaders_Icon = New_Icon WHERE School_ID = in_School_ID; END;
```

Question 4

Run your code to create the stored procedure.

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.

```
1 CALL UPDATE_LEADERS_SCORE(609927,
  50);
```

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

```
# Adding 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.

DELIMITER $$

CREATE PROCEDURE UPDATE_LEADERS_SCORE(
    IN in_School_ID INT,
    IN in_Leader_Score INT
)
BEGIN
    # Declare the new icon
    DECLARE New_Icon VARCHAR(50);

    # Start the transaction
    START TRANSACTION;

    # Check if the leader score is within the valid range (0-99)
    IF in_Leader_Score >= 0 AND in_Leader_Score <= 99 THEN

        # Determine the Leaders_Icon based on the Leader Score range
        IF in_Leader_Score >= 80 AND in_Leader_Score <= 99 THEN
            SET New_Icon = 'Very strong';
        ELSEIF in_Leader_Score >= 60 AND in_Leader_Score <= 79 THEN
            SET New_Icon = 'Strong';
        ELSEIF in_Leader_Score >= 40 AND in_Leader_Score <= 59 THEN
            SET New_Icon = 'Average';
        ELSEIF in_Leader_Score >= 20 AND in_Leader_Score <= 39 THEN
            SET New_Icon = 'Weak';
        ELSEIF in_Leader_Score >= 0 AND in_Leader_Score <= 19 THEN
            SET New_Icon = 'Very weak';
        END IF;

        # Update the Leaders_Score and Leaders_Icon for the given School_ID
        UPDATE chicago_public_schools
        SET Leaders_Score = in_Leader_Score, Leaders_Icon = New_Icon
        WHERE School_ID = in_School_ID;

        # Commit the transaction if all updates are successful
        COMMIT;

    ELSE
        # If the score is outside the valid range (0-99), roll back the transaction
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid Leader Score: Must be between 0 and 99';
    END IF;

```

```
END $$

DELIMITER ;
```

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0076 seconds.)

```
CREATE PROCEDURE UPDATE_LEADERS_SCORE( IN in_School_ID INT, IN in_Leader_Score INT ) BEGIN # Declare the new icon DECLARE New_Icon VARCHAR(50); # Start the transaction START TRANSACTION; # Check if the leader score is within the valid range (0-99) IF in_Leader_Score >= 0 AND in_Leader_Score <= 99 THEN # Determine the Leaders_Icon based on the Leader Score range IF in_Leader_Score >= 80 AND in_Leader_Score <= 99 THEN SET New_Icon = 'Very strong'; ELSEIF in_Leader_Score >= 60 AND in_Leader_Score <= 79 THEN SET New_Icon = 'Strong'; ELSEIF in_Leader_Score >= 40 AND in_Leader_Score <= 59 THEN SET New_Icon = 'Average'; ELSEIF in_Leader_Score >= 20 AND in_Leader_Score <= 39 THEN SET New_Icon = 'Weak'; ELSEIF
```

Question 2

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

Run your code to replace the stored procedure.

```
# Adding a statement to commit the current unit of work at the end of the procedure.

DELIMITER $$

CREATE PROCEDURE UPDATE_LEADERS_SCORE(
    IN in_School_ID INT,
    IN in_Leader_Score INT
)
BEGIN
    -- Declare the new icon
    DECLARE New_Icon VARCHAR(50);

    -- Start the transaction
    START TRANSACTION;

    -- Check if the leader score is within the valid range (0-99)
     IF in_Leader_Score >= 0 AND in_Leader_Score <= 99 THEN

        -- Determine the Leaders_Icon based on the Leader Score range
        IF in_Leader_Score >= 80 AND in_Leader_Score <= 99 THEN
            SET New_Icon = 'Very strong';
        ELSEIF in_Leader_Score >= 60 AND in_Leader_Score <= 79 THEN
            SET New_Icon = 'Strong';
        ELSEIF in_Leader_Score >= 40 AND in_Leader_Score <= 59 THEN
            SET New_Icon = 'Average';
        ELSEIF in_Leader_Score >= 20 AND in_Leader_Score <= 39 THEN
            SET New_Icon = 'Weak';
        ELSEIF in_Leader_Score >= 0 AND in_Leader_Score <= 19 THEN
            SET New_Icon = 'Very weak';
        END IF;

        -- Update the Leaders_Score and Leaders_Icon for the given School_ID
```

```
33          UPDATE chicago_public_schools
34          SET Leaders_Score = in_Leader_Score, Leaders_Icon = New_Icon
35          WHERE School_ID = in_School_ID;
36
37          -- Commit the transaction since score is valid and update was successful
38          COMMIT;
39
40      ELSE
41          -- If the score is outside the valid range (0-99), roll back the transaction
42          ROLLBACK;
43          SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid Leader Score: Must be between 0 and 99';
44      END IF;
45
46  END $$
47
48  DELIMITER ;
```

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

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0031 seconds.)

```
CALL UPDATE_LEADERS_SCORE(609709, 38);
```

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

```
1 CALL UPDATE_LEADERS_SCORE(609709, 101);
```

Error

SQL query: Copy

```
CALL UPDATE_LEADERS_SCORE(609709, 101);
```

MySQL said:

```
#1644 - Invalid Leader Score: Must be between 0 and 99
```