SQL ACID transaction & stored procedure

Database(server) query optimization and transaction control

About project

Objectives:

1. Use joins to query data from multiple tables
2. Create and query views
3. Write and run stored procedures
4. Use transactions

Scenario:

In this project, I worked with three datasets that are available on the City of Chicago’s Data Portal:

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

I created a table for each one, and load the appropriate dataset through the Db2 console. If you have already completed the Hands on Lab: Joins, you can reuse the tables you created for that hands-on lab.

Skills utilised

SQL

Database management

Cloud computing

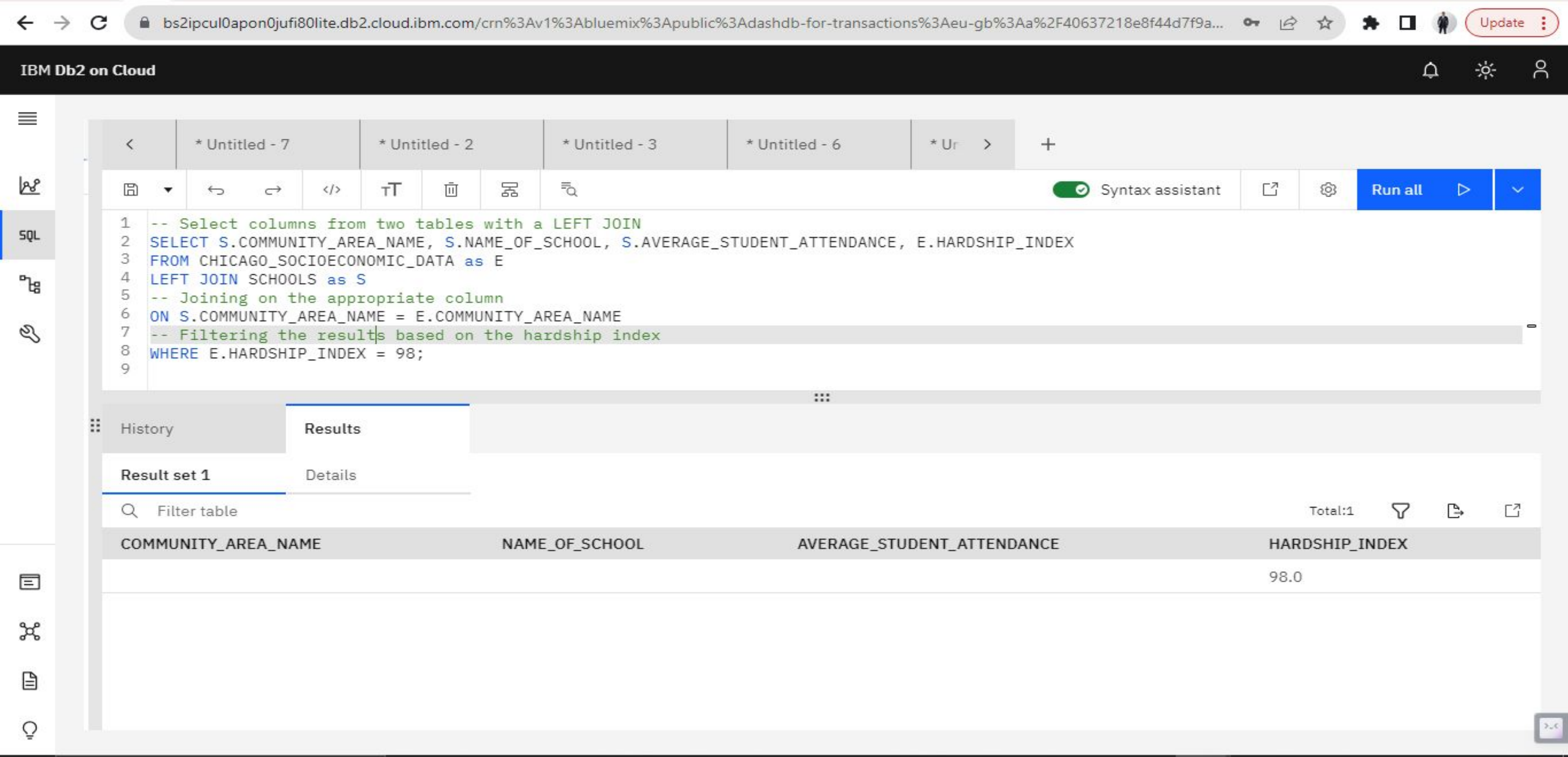
Software used: IBM Db2 cloud console

Project screenshots and explanations

Next seven slides

Join Tables

Wrote and executed a SQL query to list the school names, community names and average attendance for communities with a hardship index of 98. Combining data from both the SCHOOLS table and SOCIOECONOMIC table.

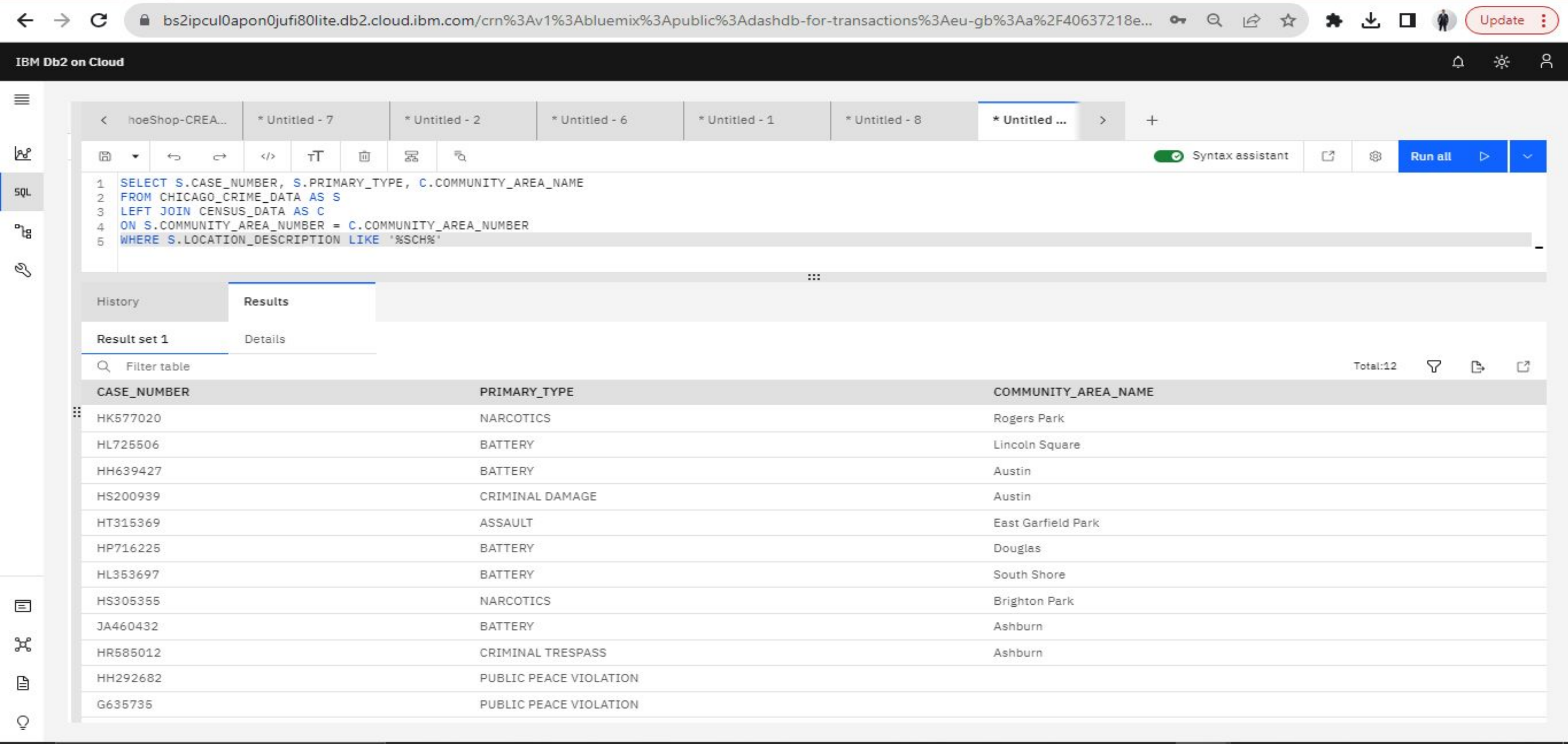


```
-- Select columns from two tables with a LEFT JOIN
SELECT S.COMMUNITY_AREA_NAME, S.NAME_OF_SCHOOL, S.AVERAGE_STUDENT_ATTENDANCE, E.HARDSHIP_INDEX
FROM CHICAGO_SOCIOECONOMIC_DATA as E
LEFT JOIN SCHOOLS as S
-- Joining on the appropriate column
ON S.COMMUNITY_AREA_NAME = E.COMMUNITY_AREA_NAME
-- Filtering the results based on the hardship index
WHERE E.HARDSHIP_INDEX = 98;
```

COMMUNITY_AREA_NAME	NAME_OF_SCHOOL	AVERAGE_STUDENT_ATTENDANCE	HARDSHIP_INDEX
			98.0

Sort Crimes in schools

Wrote and executed a SQL query to list all crimes that took place at a school. Include case number, crime type and community name columns.

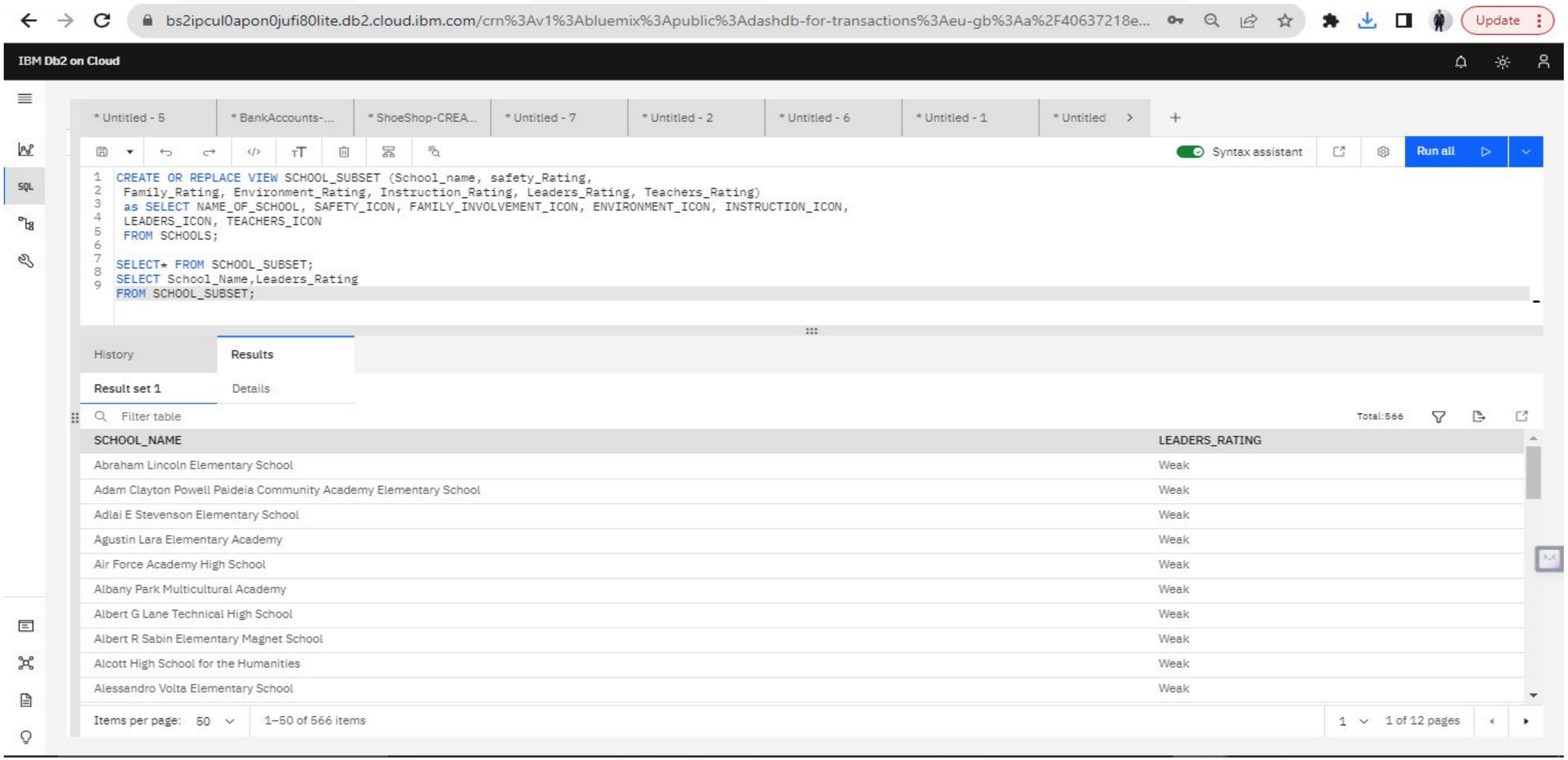


```
SELECT S.CASE_NUMBER, S.PRIMARY_TYPE, C.COMMUNITY_AREA_NAME
FROM CHICAGO_CRIME_DATA AS S
LEFT JOIN CENSUS_DATA AS C
ON S.COMMUNITY_AREA_NUMBER = C.COMMUNITY_AREA_NUMBER
WHERE S.LOCATION_DESCRIPTION LIKE '%SCH%'
```

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

Create a view to protect sensitive data

Wrote and executed a SQL statement to create a view showing the columns listed in the following table, with new column names, School_Name, Safety_Rating, Family_Rating, Environment_Rating, Instruction_Rating, Leaders_Rating, Teachers_Rating.



```
CREATE OR REPLACE VIEW SCHOOL_SUBSET (School_name, safety_Rating,
 Family_Rating, Environment_Rating, Instruction_Rating, Leaders_Rating, Teachers_Rating)
 as SELECT NAME_OF_SCHOOL, SAFETY_ICON, FAMILY_INVOLVEMENT_ICON, ENVIRONMENT_ICON, INSTRUCTION_ICON,
 LEADERS_ICON, TEACHERS_ICON
 FROM SCHOOLS;

SELECT* FROM SCHOOL_SUBSET;
SELECT School_Name,Leaders_Rating
FROM SCHOOL_SUBSET;
```

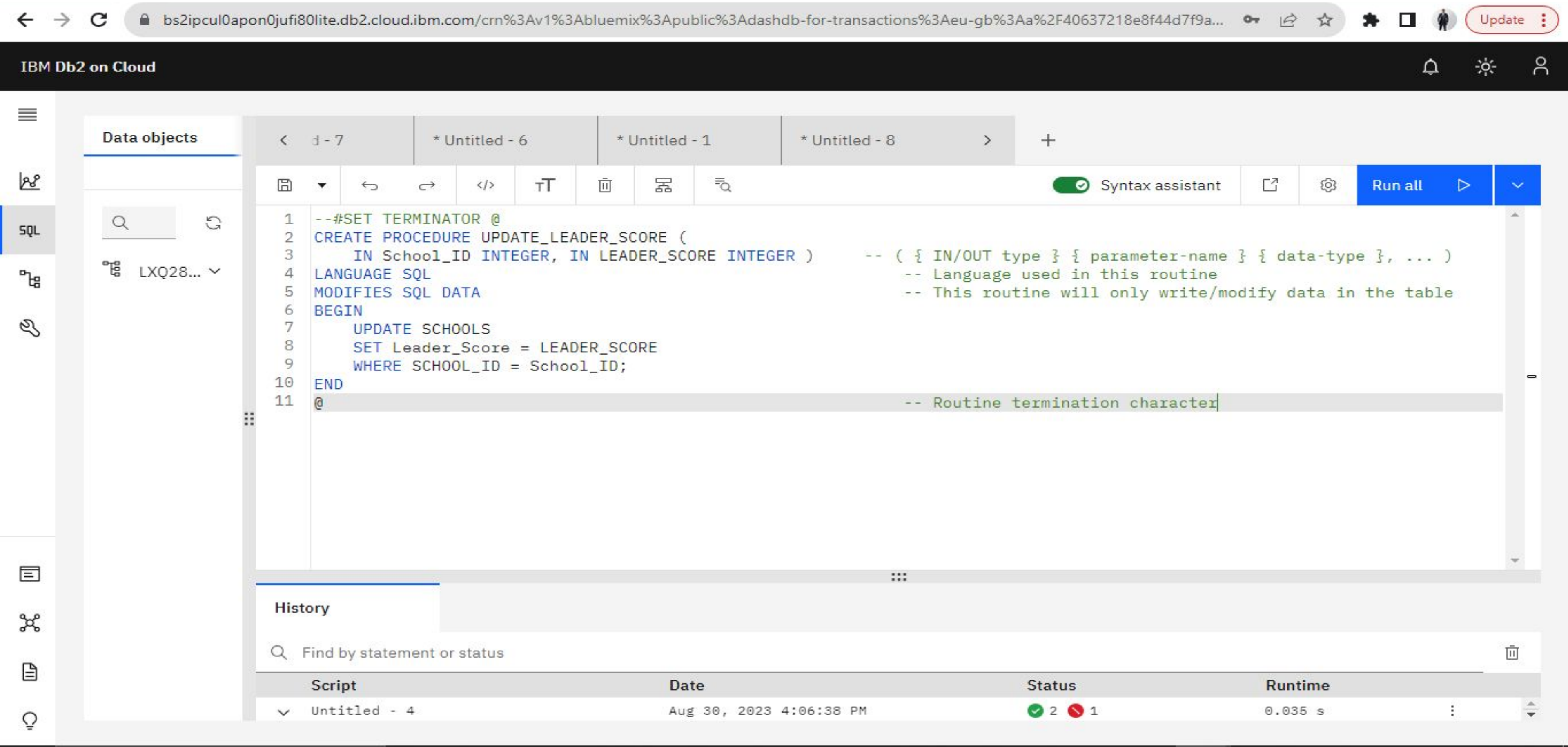
Total:566

SCHOOL_NAME	LEADERS_RATING
Abraham Lincoln Elementary School	Weak
Adam Clayton Powell Paideia Community Academy Elementary School	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

Items per page: 50 1–50 of 566 items 1 1 of 12 pages

Create Stored Procedure

Wrote the structure of a query to create or replace a stored procedure called UPDATE_LEADERS_SCORE that accepts two inputs: SCHOOL_ID and LEADERS_SCORE to automate database update when called from an external application.


bs2ipcul0apon0jufi80lite.db2.cloud.ibm.com/crn%3Av1%3Abluemix%3Apublic%3Adashdb-for-transactions%3Aeu-gb%3Aa%2F40637218e8f44d7f9a...
Update
IBM Db2 on Cloud
Data objects
LXQ28...
d - 7
* Untitled - 6
* Untitled - 1
* Untitled - 8
Syntax assistant
Run all

```
--#SET TERMINATOR @
CREATE PROCEDURE UPDATE_LEADER_SCORE (
    IN School_ID INTEGER, IN LEADER_SCORE INTEGER )     -- ( { IN/OUT type } { parameter-name } { data-type }, ... )
LANGUAGE SQL                                              -- Language used in this routine
MODIFIES SQL DATA                                         -- This routine will only write/modify data in the table
BEGIN
    UPDATE SCHOOLS
    SET Leader_Score = LEADER_SCORE
    WHERE SCHOOL_ID = School_ID;
END
@                                                         -- Routine termination character
```

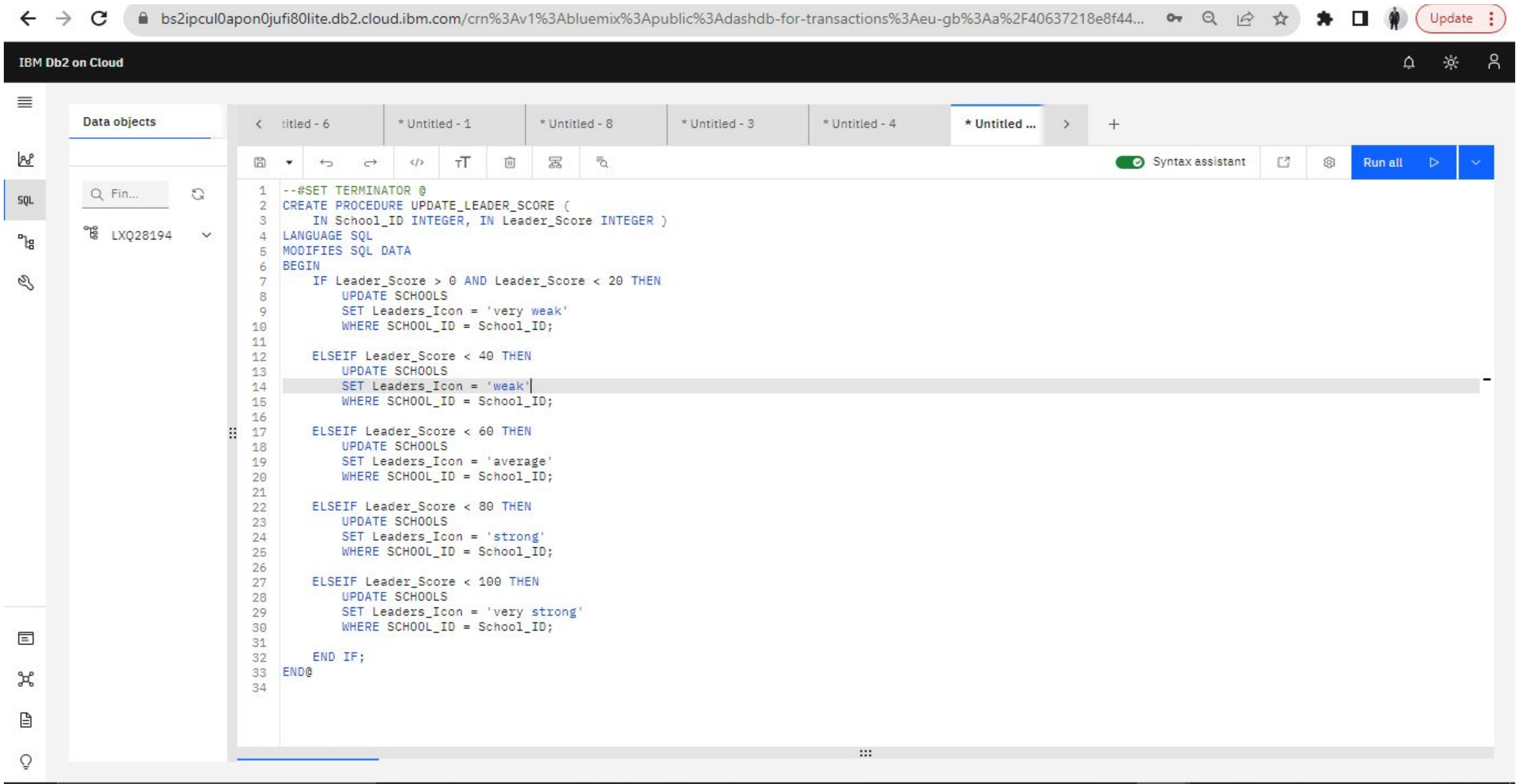
History

Find by statement or status

Script	Date	Status	Runtime
Untitled - 4	Aug 30, 2023 4:06:38 PM	✓ 2 ⊘ 1	0.035 s

Modify Stored Procedure to effect more database updates.

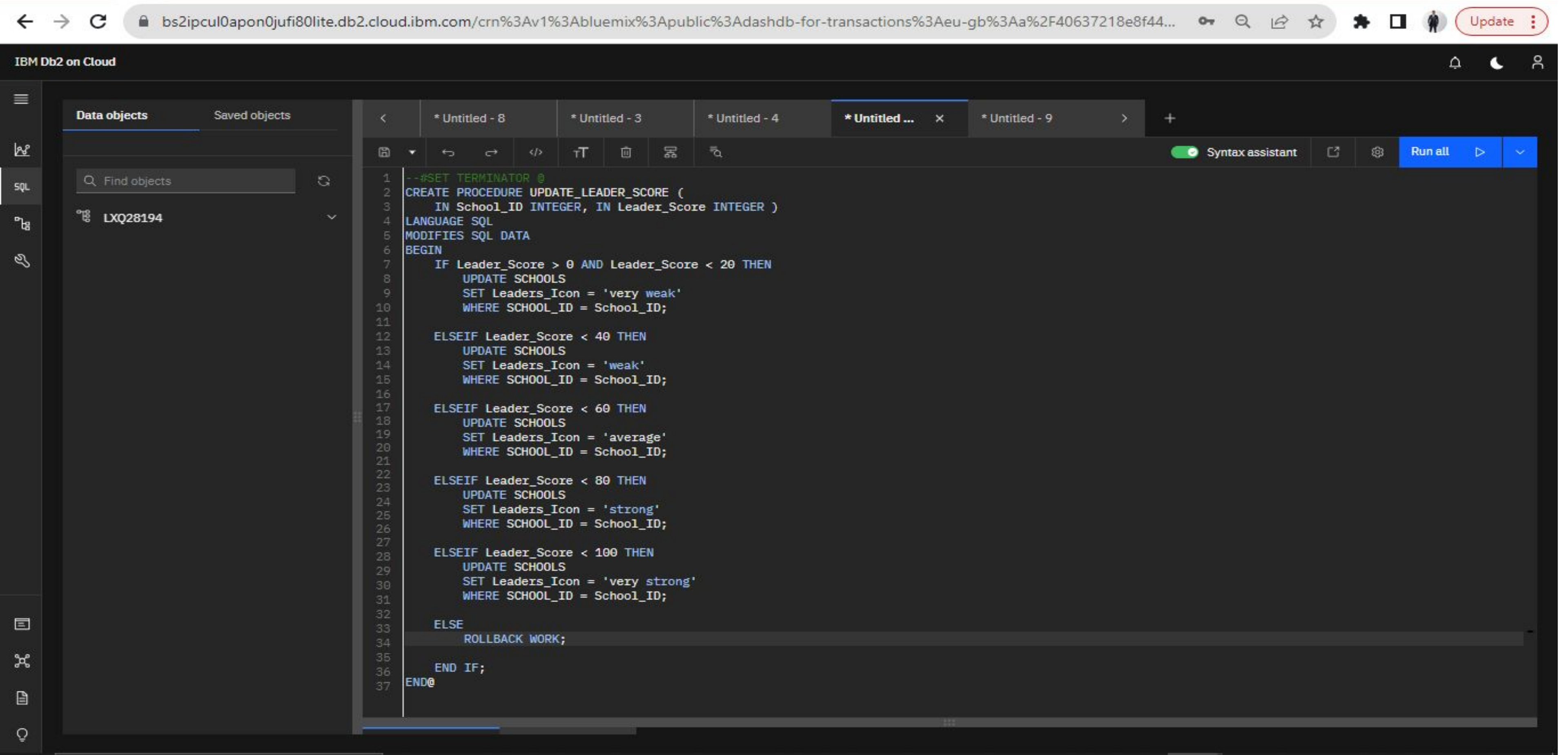
Inside the initial stored procedure, I wrote a SQL IF statement to update the Leaders_Icon field in the CHICAGO_PUBLIC_SCHOOLS table for the school identified by in_School_ID.



```
--#SET TERMINATOR @
CREATE PROCEDURE UPDATE_LEADER_SCORE (
    IN School_ID INTEGER, IN Leader_Score INTEGER )
LANGUAGE SQL
MODIFIES SQL DATA
BEGIN
    IF Leader_Score > 0 AND Leader_Score < 20 THEN
        UPDATE SCHOOLS
        SET Leaders_Icon = 'very weak'
        WHERE SCHOOL_ID = School_ID;

    ELSEIF Leader_Score < 40 THEN
        UPDATE SCHOOLS
        SET Leaders_Icon = 'weak'
        WHERE SCHOOL_ID = School_ID;

    ELSEIF Leader_Score < 60 THEN
        UPDATE SCHOOLS
        SET Leaders_Icon = 'average'
        WHERE SCHOOL_ID = School_ID;

    ELSEIF Leader_Score < 80 THEN
        UPDATE SCHOOLS
        SET Leaders_Icon = 'strong'
        WHERE SCHOOL_ID = School_ID;

    ELSEIF Leader_Score < 100 THEN
        UPDATE SCHOOLS
        SET Leaders_Icon = 'very strong'
        WHERE SCHOOL_ID = School_ID;

    END IF;
END@
```

Use ACID Transaction to protect data integrity in the database

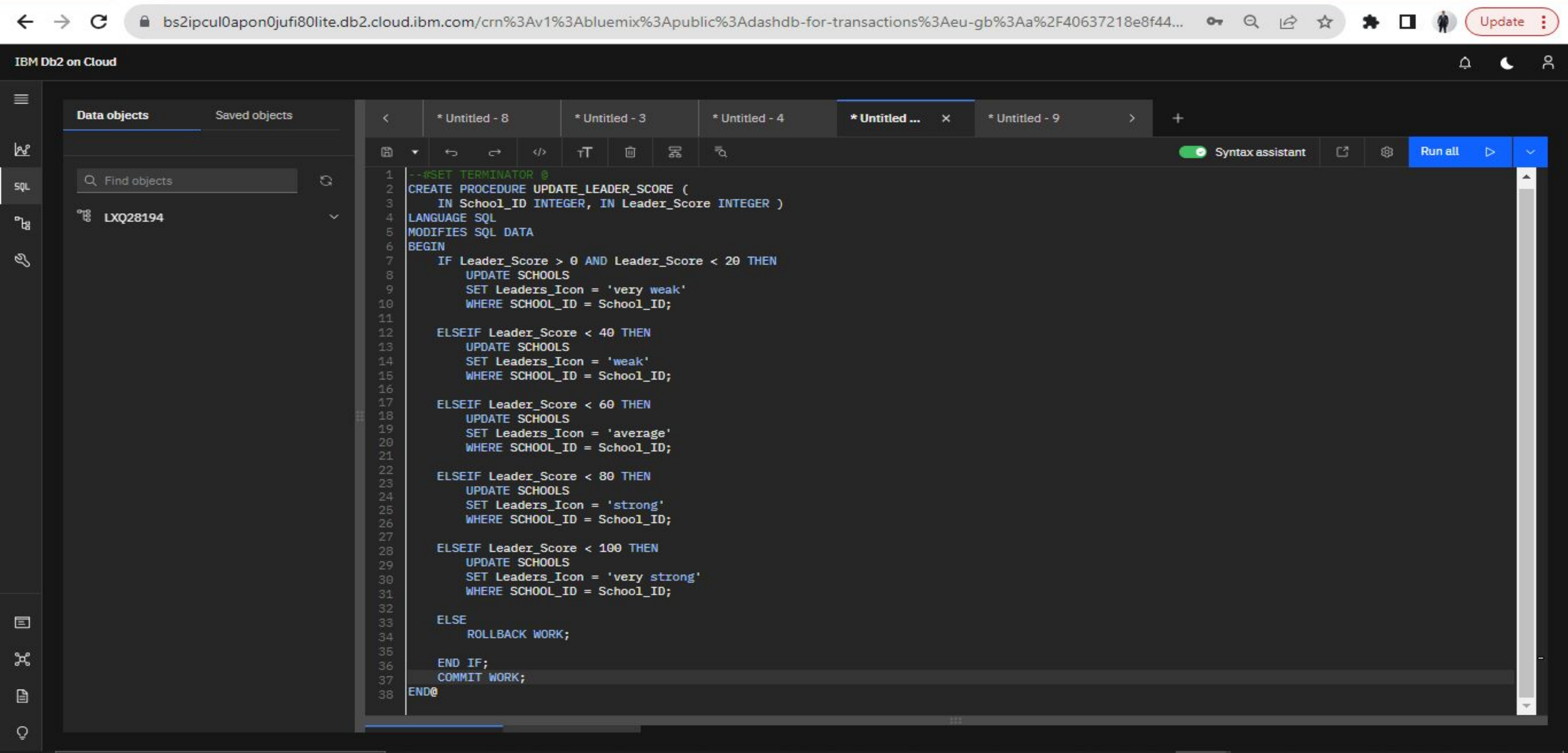
Updated the stored procedure definition in the last slide to add a generic ELSE clause to the IF statement that rolls back the current work if the score did not fit any of the preceding categories to ensure data consistency in the database.



```
--#SET TERMINATOR @
CREATE PROCEDURE UPDATE_LEADER_SCORE (
    IN School_ID INTEGER, IN Leader_Score INTEGER )
LANGUAGE SQL
MODIFIES SQL DATA
BEGIN
    IF Leader_Score > 0 AND Leader_Score < 20 THEN
        UPDATE SCHOOLS
        SET Leaders_Icon = 'very weak'
        WHERE SCHOOL_ID = School_ID;

    ELSEIF Leader_Score < 40 THEN
        UPDATE SCHOOLS
        SET Leaders_Icon = 'weak'
        WHERE SCHOOL_ID = School_ID;

    ELSEIF Leader_Score < 60 THEN
        UPDATE SCHOOLS
        SET Leaders_Icon = 'average'
        WHERE SCHOOL_ID = School_ID;

    ELSEIF Leader_Score < 80 THEN
        UPDATE SCHOOLS
        SET Leaders_Icon = 'strong'
        WHERE SCHOOL_ID = School_ID;

    ELSEIF Leader_Score < 100 THEN
        UPDATE SCHOOLS
        SET Leaders_Icon = 'very strong'
        WHERE SCHOOL_ID = School_ID;

    ELSE
        ROLLBACK WORK;

    END IF;
END@
```

ACID TRANSACTION-COMMIT SUCCESSFUL OPERATIONS IN DB

Updated the stored procedure definition again in the last slide to add a statement to commit the current unit of work at the end of the procedure if the DB operation is successful.



```
--#SET TERMINATOR @
CREATE PROCEDURE UPDATE_LEADER_SCORE (
    IN School_ID INTEGER, IN Leader_Score INTEGER )
LANGUAGE SQL
MODIFIES SQL DATA
BEGIN
    IF Leader_Score > 0 AND Leader_Score < 20 THEN
        UPDATE SCHOOLS
        SET Leaders_Icon = 'very weak'
        WHERE SCHOOL_ID = School_ID;

    ELSEIF Leader_Score < 40 THEN
        UPDATE SCHOOLS
        SET Leaders_Icon = 'weak'
        WHERE SCHOOL_ID = School_ID;

    ELSEIF Leader_Score < 60 THEN
        UPDATE SCHOOLS
        SET Leaders_Icon = 'average'
        WHERE SCHOOL_ID = School_ID;

    ELSEIF Leader_Score < 80 THEN
        UPDATE SCHOOLS
        SET Leaders_Icon = 'strong'
        WHERE SCHOOL_ID = School_ID;

    ELSEIF Leader_Score < 100 THEN
        UPDATE SCHOOLS
        SET Leaders_Icon = 'very strong'
        WHERE SCHOOL_ID = School_ID;

    ELSE
        ROLLBACK WORK;

    END IF;
    COMMIT WORK;
END@
```

Conclusion

ACID transactions play a crucial role in maintaining the reliability, integrity, and consistency of data, which directly benefits businesses and their customers by providing a solid foundation for accurate and trustworthy operations.

With database entities secured and automated as illustrated in this project, businesses can be sure their database is secured and data entries consistent, helping the customers and helping themselves at the same time

Contact

Ogbonna Ngwu

ngwuogbonnaprince@gmail.com

+2348165533706