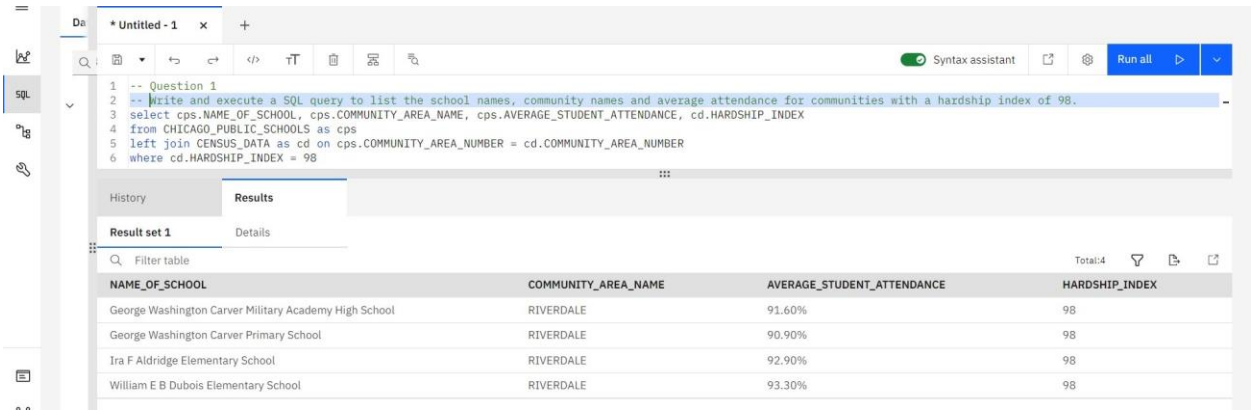
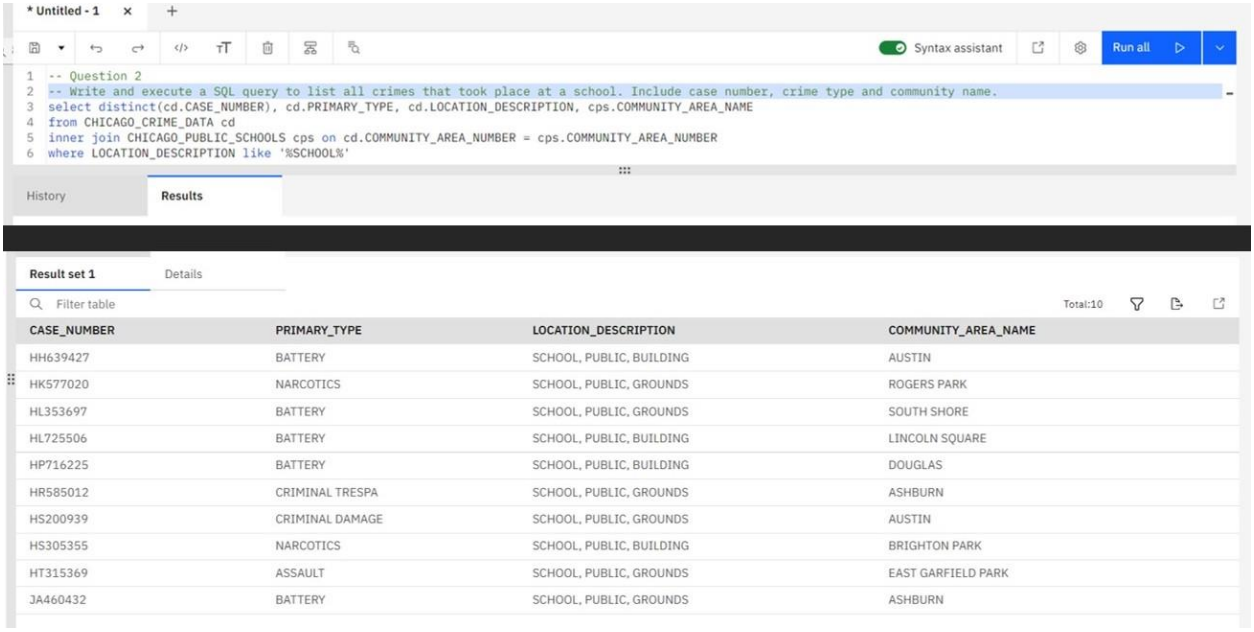
IBM Data Science: Advanced SQL Techniques Final

- Work with three datasets that are available on the City of Chicago's Data Portal:
    - Socioeconomic indicators in Chicago
        - six socioeconomic indicators of public health significance and a “hardship index,” for each Chicago community area, for the years 2008 – 2012.
        - https://data.cityofchicago.org/Health-Human-Services/Census-Data-Selected-socioeconomic-indicators-in-C/kn9c-c2s2
    - Chicago public schools
        - all school level performance data used to create CPS School Report Cards for the 2011-2012 school year.
        - https://data.cityofchicago.org/Education/Chicago-Public-Schools-Progress-Report-Cards-2011-/9xs2-f89t
    - Chicago crime data
        - reported incidents of crime (with the exception of murders where data exists for each victim) that occurred in the City of Chicago from 2001 to present, minus the most recent seven days.
        - https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2
- Store the datasets in database tables in IBM DB2
    - Download .CSV for each dataset
        - .CSVs are cleaned subsets (500 rows) for each dataset
    - Load into DB2
    - Create DB2 table for each .CSV

- Exercise 1: Using Joins - You have been asked to produce some reports about the communities and crimes in the Chicago area.
    - 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.



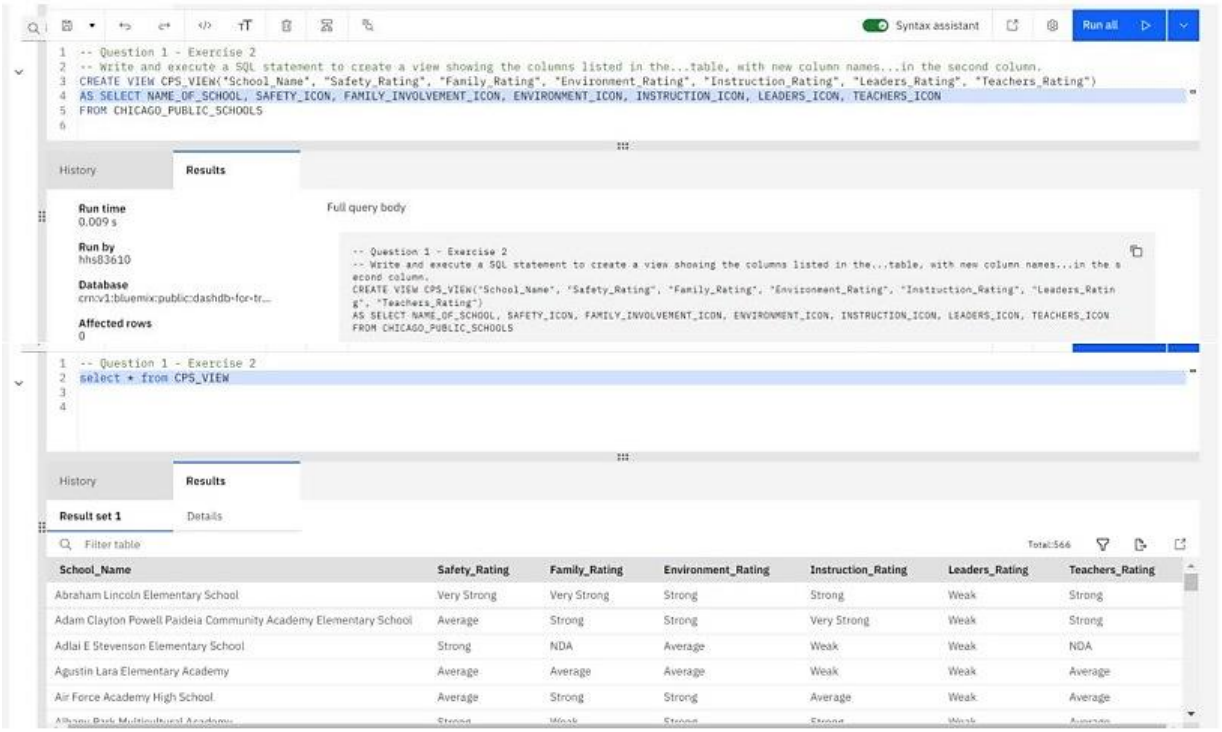
- Question 2: Write and execute a SQL query to list all crimes that took place at a school. Include case number, crime type and community name.



- 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.

CHICAGO_PUBLIC_SCHOOLS	**NAME for 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

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



```
-- Question 1 - Exercise 2
-- Write and execute a SQL statement to create a view showing the columns listed in the...table, with new column names...in the second column.
CREATE VIEW CPS_VIEW("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 CHICAGO_PUBLIC_SCHOOLS
```

Syntax assistant | Run all

History | Results

Run time
0.009 s

Run by
hhs83610

Database
crn:v1:bluemix:public:dashdb-for-tr...

Affected rows
0

Full query body

```
-- Question 1 - Exercise 2
-- Write and execute a SQL statement to create a view showing the columns listed in the...table, with new column names...in the s
econd column.
CREATE VIEW CPS_VIEW("School_Name", "Safety_Rating", "Family_Rating", "Environment_Rating", "Instruction_Rating", "Leaders_Ratin
g", "Teachers_Rating")
AS SELECT NAME_OF_SCHOOL, SAFETY_ICON, FAMILY_INVOLVEMENT_ICON, ENVIRONMENT_ICON, INSTRUCTION_ICON, LEADERS_ICON, TEACHERS_ICON
FROM CHICAGO_PUBLIC_SCHOOLS
```

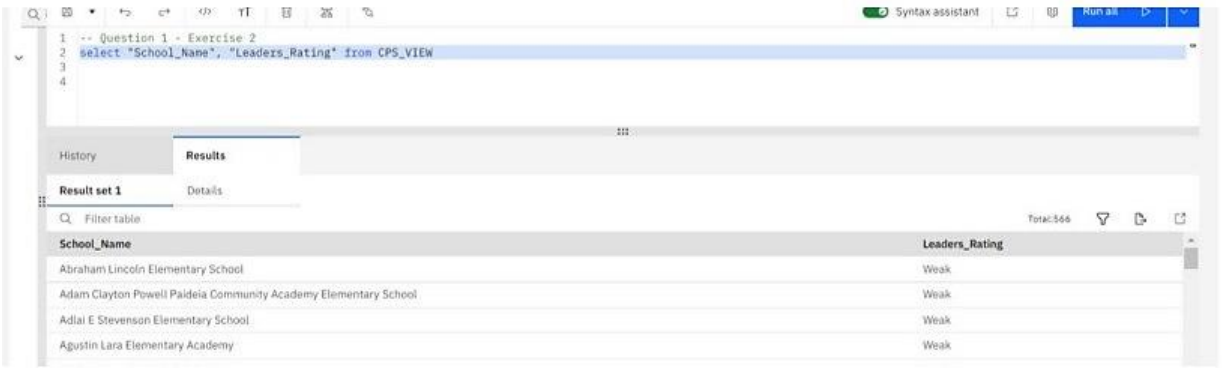
```
-- Question 1 - Exercise 2
select * from CPS_VIEW
```

History | Results

Result set 1 | Details

Filter table — Total:566

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 Elementary School	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



```
-- Question 1 - Exercise 2
select "School_Name", "Leaders_Rating" from CPS_VIEW
```

Syntax assistant | Run all

History | Results

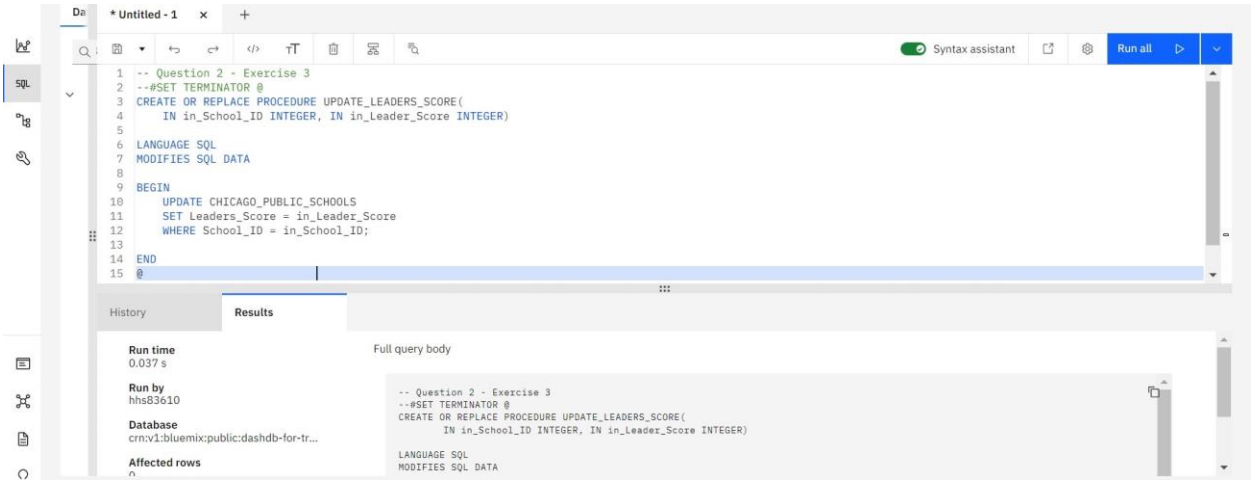
Result set 1 | Details

Filter table — 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

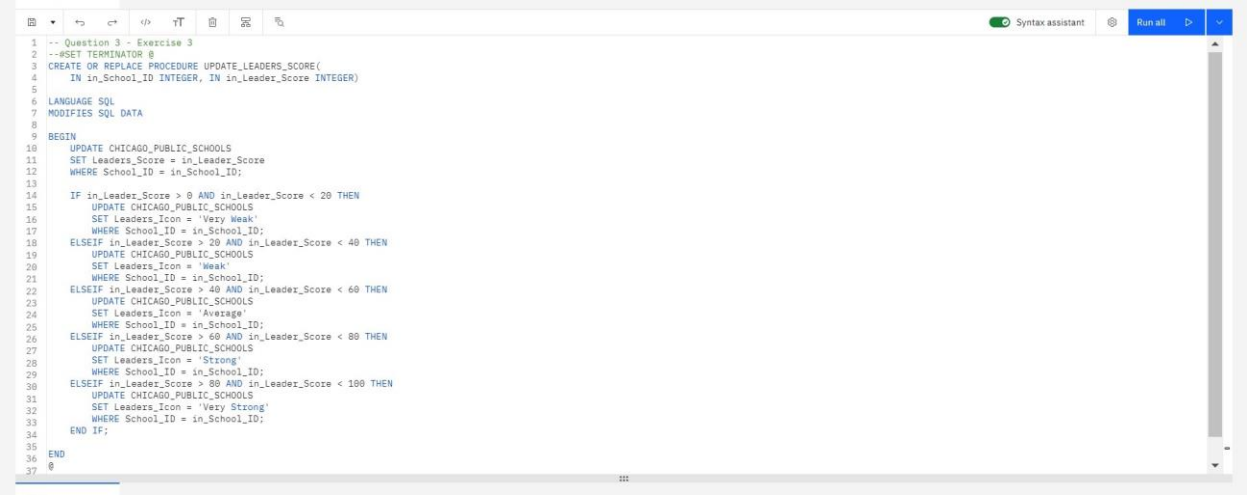
- Exercise 3: Creating a Stored Procedure - The icon fields are calculated based on the value in the corresponding score field. You need to make sure that when a score field is updated, the icon field is updated too. To do this, you will write a stored procedure that receives the school id and a leaders 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. Don't forget to use the #SET TERMINATOR statement to use the @ for the CREATE statement terminator.

    - 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.

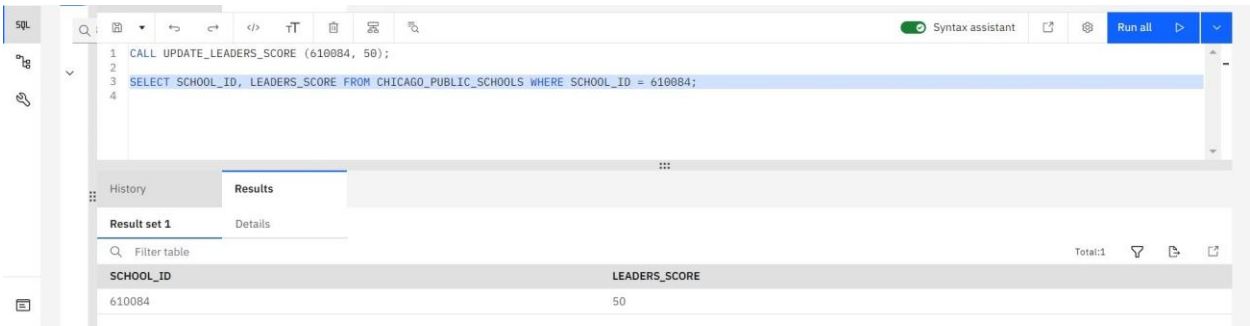


- Question 3: Inside your stored procedure, write a SQL IF statement to update the Leaders_Icon field in the CHICAGO_PUBLIC_SCHOOLS table for the school identified by in_School_ID using the following information.

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

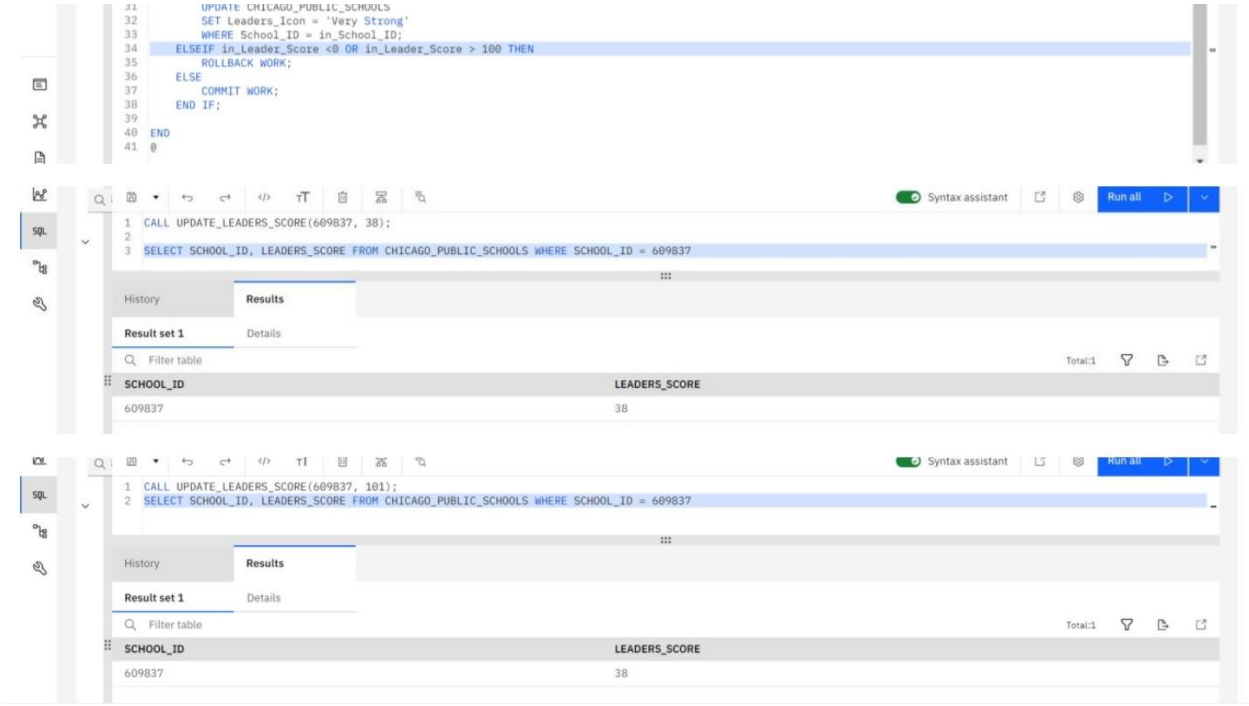


- Question 4: Run your code to create the stored procedure.
  - Write a query to call the stored procedure, passing a valid school ID and a leader score of 50, to check that the procedure works as expected.



- Exercise 4: Using Transactions - You realize 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.
  - Write and run one query to check that the updated stored procedure works as expected when you use a valid score of 38.
  - Write and run another query to check that the updated stored procedure works as expected when you use an invalid score of 101.



```
        UPDATE CHICAGO_PUBLIC_SCHOOLS
        SET Leaders_Icon = 'Very Strong'
        WHERE School_ID = in_School_ID;
    ELSEIF in_Leader_Score <0 OR in_Leader_Score > 100 THEN
        ROLLBACK WORK;
    ELSE
        COMMIT WORK;
    END IF;

END
@
```

Syntax assistant | Run all

```
CALL UPDATE_LEADERS_SCORE(609837, 38);

SELECT SCHOOL_ID, LEADERS_SCORE FROM CHICAGO_PUBLIC_SCHOOLS WHERE SCHOOL_ID = 609837
```

History | Results

Result set 1 | Details

Filter table — Total:1

SCHOOL_ID	LEADERS_SCORE
609837	38

Syntax assistant | Run all

```
CALL UPDATE_LEADERS_SCORE(609837, 101);
SELECT SCHOOL_ID, LEADERS_SCORE FROM CHICAGO_PUBLIC_SCHOOLS WHERE SCHOOL_ID = 609837
```

History | Results

Result set 1 | Details

Filter table — Total:1

SCHOOL_ID	LEADERS_SCORE
609837	38