

## 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.

The screenshot shows the IBM Data Studio interface. The top panel displays a SQL query for 'Question 1'. The query is designed to list school names, community names, and average attendance for communities with a hardship index of 98. It uses a left join between the CHICAGO\_PUBLIC\_SCHOOLS table (aliased as cps) and the CENSUS\_DATA table (aliased as cd) based on their community area numbers.

```
1 -- Question 1
2 -- Write and execute a SQL query to list the school names, community names and average attendance for communities with a hardship index of 98.
3 select cps.NAME_OF_SCHOOL, cps.COMMUNITY_AREA_NAME, cps.AVERAGE_STUDENT_ATTENDANCE, cd.HARDSHIP_INDEX
4 from CHICAGO_PUBLIC_SCHOOLS as cps
5 left join CENSUS_DATA as cd on cps.COMMUNITY_AREA_NUMBER = cd.COMMUNITY_AREA_NUMBER
6 where cd.HARDSHIP_INDEX = 98
```

The bottom panel shows the 'Results' tab with 'Result set 1'. It displays a table with 4 columns: NAME\_OF\_SCHOOL, COMMUNITY\_AREA\_NAME, AVERAGE\_STUDENT\_ATTENDANCE, and HARDSHIP\_INDEX. The table contains 4 rows of data, all from the RIVERDALE community area.

NAME_OF_SCHOOL	COMMUNITY_AREA_NAME	AVERAGE_STUDENT_ATTENDANCE	HARDSHIP_INDEX
George Washington Carver Military Academy High School	RIVERDALE	91.60%	98
George Washington Carver Primary School	RIVERDALE	90.90%	98
Ira F Aldridge Elementary School	RIVERDALE	92.90%	98
William E B Dubois Elementary School	RIVERDALE	93.30%	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.

The screenshot shows a SQL IDE with a query editor and a results pane. The query is as follows:

```

1 -- Question 2
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.
3 select distinct(cd.CASE_NUMBER), cd.PRIMARY_TYPE, cd.LOCATION_DESCRIPTION, cps.COMMUNITY_AREA_NAME
4 from CHICAGO_CRIME_DATA cd
5 inner join CHICAGO_PUBLIC_SCHOOLS cps on cd.COMMUNITY_AREA_NUMBER = cps.COMMUNITY_AREA_NUMBER
6 where LOCATION_DESCRIPTION like '%SCHOOL%'

```

The results pane shows the following data:

CASE_NUMBER	PRIMARY_TYPE	LOCATION_DESCRIPTION	COMMUNITY_AREA_NAME
HH639427	BATTERY	SCHOOL, PUBLIC, BUILDING	AUSTIN
HK577020	NARCOTICS	SCHOOL, PUBLIC, GROUNDS	ROGERS PARK
HL353697	BATTERY	SCHOOL, PUBLIC, GROUNDS	SOUTH SHORE
HL725506	BATTERY	SCHOOL, PUBLIC, BUILDING	LINCOLN SQUARE
HP716225	BATTERY	SCHOOL, PUBLIC, BUILDING	DOUGLAS
HR585012	CRIMINAL TRESPA	SCHOOL, PUBLIC, GROUNDS	ASHBURN
HS200939	CRIMINAL DAMAGE	SCHOOL, PUBLIC, GROUNDS	AUSTIN
HS305355	NARCOTICS	SCHOOL, PUBLIC, BUILDING	BRIGHTON PARK
HT315369	ASSAULT	SCHOOL, PUBLIC, GROUNDS	EAST GARFIELD PARK
JA460432	BATTERY	SCHOOL, PUBLIC, GROUNDS	ASHBURN

- 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.

1 -- Question 1 - Exercise 2  
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.  
3 CREATE VIEW CPS\_VIEW('School\_Name', 'Safety\_Rating', 'Family\_Rating', 'Environment\_Rating', 'Instruction\_Rating', 'Leaders\_Rating', 'Teachers\_Rating')  
4 AS SELECT NAME\_OF\_SCHOOL, SAFETY\_ICON, FAMILY\_INVOLVEMENT\_ICON, ENVIRONMENT\_ICON, INSTRUCTION\_ICON, LEADERS\_ICON, TEACHERS\_ICON  
5 FROM CHICAGO\_PUBLIC\_SCHOOLS  
6

History

Results

Run time  
0.009 s

Run by  
hhs83610

Database  
cmov1bluemix: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 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

1 -- Question 1 - Exercise 2  
2 select \* from CPS\_VIEW  
3  
4

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
Alfonso Bask Multicultural Academy	Strong	Weak	Strong	Strong	Weak	Average

1 -- Question 1 - Exercise 2  
2 select "School\_Name", "Leaders\_Rating" from CPS\_VIEW  
3  
4

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.

The screenshot shows a SQL IDE with a query editor and a results pane. The query editor contains the following SQL code:

```

1 -- Question 1 - Exercise 3
2
3 --#SET TERMINATOR @
4 CREATE PROCEDURE UPDATE_LEADERS_SCORE(
5     IN in_School_ID INTEGER, IN in_Leader_Score INTEGER)
6
7 LANGUAGE SQL
8 READS SQL DATA
9
10 BEGIN
11
12 END
13 @
14
15

```

The results pane shows the execution details:

- Run time: 0.024 s
- Run by: hhs83610
- Database: crn:v1:bluemix:public:dashdb-for-tr...
- Affected rows: 0

The full query body is displayed in the results pane, matching the code in the editor.

- 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.

The screenshot shows the same SQL IDE with the query editor updated to include an UPDATE statement within the stored procedure. The query editor contains the following SQL code:

```

1 -- Question 2 - Exercise 3
2 --#SET TERMINATOR @
3 CREATE OR REPLACE PROCEDURE UPDATE_LEADERS_SCORE(
4     IN in_School_ID INTEGER, IN in_Leader_Score INTEGER)
5
6 LANGUAGE SQL
7 MODIFIES SQL DATA
8
9 BEGIN
10     UPDATE CHICAGO_PUBLIC_SCHOOLS
11     SET Leaders_Score = in_Leader_Score
12     WHERE School_ID = in_School_ID;
13
14 END
15 @
16
17

```

The results pane shows the execution details:

- Run time: 0.037 s
- Run by: hhs83610
- Database: crn:v1:bluemix:public:dashdb-for-tr...
- Affected rows: 0

The full query body is displayed in the results pane, matching the code in the editor.

- 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

```

1 -- Question 3 - Exercise 3
2 --SET TERMINATOR @
3 CREATE OR REPLACE PROCEDURE UPDATE_LEADERS_SCORE(
4     IN in_School_ID INTEGER, IN in_Leader_Score INTEGER)
5
6 LANGUAGE SQL
7 MODIFIES SQL DATA
8
9 BEGIN
10     UPDATE CHICAGO_PUBLIC_SCHOOLS
11     SET Leaders_Score = in_Leader_Score
12     WHERE School_ID = in_School_ID;
13
14     IF in_Leader_Score > 0 AND in_Leader_Score < 20 THEN
15         UPDATE CHICAGO_PUBLIC_SCHOOLS
16         SET Leaders_Icon = 'Very Weak'
17         WHERE School_ID = in_School_ID;
18     ELSEIF in_Leader_Score > 20 AND in_Leader_Score < 40 THEN
19         UPDATE CHICAGO_PUBLIC_SCHOOLS
20         SET Leaders_Icon = 'Weak'
21         WHERE School_ID = in_School_ID;
22     ELSEIF in_Leader_Score > 40 AND in_Leader_Score < 60 THEN
23         UPDATE CHICAGO_PUBLIC_SCHOOLS
24         SET Leaders_Icon = 'Average'
25         WHERE School_ID = in_School_ID;
26     ELSEIF in_Leader_Score > 60 AND in_Leader_Score < 80 THEN
27         UPDATE CHICAGO_PUBLIC_SCHOOLS
28         SET Leaders_Icon = 'Strong'
29         WHERE School_ID = in_School_ID;
30     ELSEIF in_Leader_Score > 80 AND in_Leader_Score < 100 THEN
31         UPDATE CHICAGO_PUBLIC_SCHOOLS
32         SET Leaders_Icon = 'Very Strong'
33         WHERE School_ID = in_School_ID;
34     END IF;
35
36 END
37 @
  
```

- 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 (610084, 50);
2
3 SELECT SCHOOL_ID, LEADERS_SCORE FROM CHICAGO_PUBLIC_SCHOOLS WHERE SCHOOL_ID = 610084;
4
  
```

SCHOOL_ID	LEADERS_SCORE
610084	50

- 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.

The screenshot displays a SQL IDE interface with two panels. The top panel shows the definition of a stored procedure named `UPDATE_LEADERS_SCORE`. The procedure updates the `Leaders_Score` in the `CHICAGO_PUBLIC_SCHOOLS` table based on the `in_School_ID`. It includes an `IF` statement with conditions for scores less than 0 or greater than 100, a `ROLLBACK WORK` clause, and a `COMMIT WORK` clause. The bottom panel shows the execution of two queries. The first query calls the stored procedure with a valid score of 38, and the second query selects the updated score for the same school ID. Both queries return a single result row showing the school ID and the updated score.

```
31 UPDATE CHICAGO_PUBLIC_SCHOOLS
32 SET Leaders_Score = 'Very Strong'
33 WHERE School_ID = in_School_ID;
34 ELSEIF in_Leader_Score < 0 OR in_Leader_Score > 100 THEN
35 ROLLBACK WORK;
36 ELSE
37 COMMIT WORK;
38 END IF;
39
40 END
41 @
```

1 CALL UPDATE\_LEADERS\_SCORE(609837, 38);  
2  
3 SELECT SCHOOL\_ID, LEADERS\_SCORE FROM CHICAGO\_PUBLIC\_SCHOOLS WHERE SCHOOL\_ID = 609837

SCHOOL_ID	LEADERS_SCORE
609837	38

1 CALL UPDATE\_LEADERS\_SCORE(609837, 101);  
2 SELECT SCHOOL\_ID, LEADERS\_SCORE FROM CHICAGO\_PUBLIC\_SCHOOLS WHERE SCHOOL\_ID = 609837

SCHOOL_ID	LEADERS_SCORE
609837	38