Logo

Description automatically generated

Final Project: Advanced SQL Techniques

Estimated time needed: **60** minutes

Objectives

After completing this lab, you will be able to:

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, you will work with three datasets that are available on the City of Chicago's Data Portal:

* Socioeconomic indicators in Chicago

Chicago public schools

* Chicago crime data

You must download each dataset, create 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. However, you should not reuse similar tables with other names from other exercises or labs, as they may not create the correct results.

**Important note:**

If you have **not** yet downloaded the three datasets from the City of Chicago's Data Portal, created the required tables, and loaded the data, please follow the instructions in this section.

City of Chicago Datasets

1. **Socioeconomic indicators in Chicago**

This dataset contains a selection of six socioeconomic indicators of public health significance and a “hardship index,” for each Chicago community area, for the years 2008 – 2012. A detailed description of this dataset and the original dataset can be obtained from the Chicago Data Portal at: [https://data.cityofchicago.org/Health-Human-Services/Census-Data-Selected-socioeconomic-indicators-in-C/kn9c-c2s2](https://data.cityofchicago.org/Health-Human-Services/Census-Data-Selected-socioeconomic-indicators-in-C/kn9c-c2s2?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkDB0201ENSkillsNetwork20127838-2022-01-01)

1. **Chicago public schools**

This dataset shows all school level performance data used to create CPS School Report Cards for the 2011-2012 school year. A detailed description of this dataset and the original dataset can be obtained from the Chicago Data Portal at: [https://data.cityofchicago.org/Education/Chicago-Public-Schools-Progress-Report-Cards-2011-/9xs2-f89t](https://data.cityofchicago.org/Education/Chicago-Public-Schools-Progress-Report-Cards-2011-/9xs2-f89t?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkDB0201ENSkillsNetwork20127838-2022-01-01)

1. **Chicago crime data**

This dataset reflects 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. A detailed description of this dataset and the original dataset can be obtained from the Chicago Data Portal at: [https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2](https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkDB0201ENSkillsNetwork20127838-2022-01-01)

Store the datasets in database tables

The lab requires you to have these three tables populated with a subset of the whole datasets. Download the 'ChicagoCensusData.csv', 'ChicagoPublicSchools.csv', and 'ChicagoCrimeData.csv' datasets below and load the data into your Db2 On Cloud database.

[Chicago Census Data](https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DB0201EN-SkillsNetwork/labs/BonusModule_Coursera_v5/datasets/ChicagoCensusData.csv)

[Chicago Public Schools](https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DB0201EN-SkillsNetwork/labs/BonusModule_Coursera_v5/datasets/ChicagoPublicSchools.csv)

[Chicago Crime Data](https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DB0201EN-SkillsNetwork/labs/BonusModule_Coursera_v5/datasets/ChicagoCrimeData.csv)

You need to create a new table for each dataset. As you load each dataset, click on "(+) New Table", specify the name of the table you want to create, and then click "Next".

Graphical user interface, text, application, email

Description automatically generated

Name the new tables as follows:

1. **CENSUS\_DATA**
2. **CHICAGO\_PUBLIC\_SCHOOLS**
3. **CHICAGO\_CRIME\_DATA**

After you have created the tables, review the data in each table by using the View Data feature in the Db2 On Cloud console.

Database name = **Mysql\_Learners**

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.

[Select](https://rasaqh-8080.theiadocker-2-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) a.NAME\_OF\_SCHOOL , b.COMMUNITY\_AREA\_NAME , a.AVERAGE\_STUDENT\_ATTENDANCE,a.Average\_Teacher\_Attendance from chicago\_socioeconomic\_data b, chicago\_public\_schools a where a.COMMUNITY\_AREA\_NUMBER = b.COMMUNITY\_AREA\_NUMBER [and](https://rasaqh-8080.theiadocker-2-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/logical-operators.html#operator_and) b.HARDSHIP\_INDEX = 98

Hint 1: Use tables CHICAGO\_PUBLIC\_SCHOOLS and CENSUS\_DATA

Hint 2: Use a left join

**Take a screenshot showing the SQL query and its results.**

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.

Select a.CASE\_NUMBER ,a.PRIMARY\_TYPE, a. location\_description , b.COMMUNITY\_AREA\_NUMBER as b\_CANum, b.COMMUNITY\_AREA\_NAME from chicago\_socioeconomic\_data b , chicago\_crime a

where a.COMMUNITY\_AREA\_NUMBER = b.COMMUNITY\_AREA\_NUMBER and b.COMMUNITY\_AREA\_NUMBER is not null and a.LOCATION\_DESCRIPTION like '%School%';

go

Hint 1: Use tables CHICAGO\_CRIME\_DATA and CENSUS\_DATA

Hint 2: Use a left join

Hint 3: The column location\_description will help you find the crime location

**Take a screenshot showing the SQL query and its results.**

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.

Create view vw\_school\_list as

Select NAME\_OF\_SCHOOL 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

from chicago\_public\_schools;

go

| **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 |

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

describe vw\_school\_list;

go

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

select school\_name, Leaders\_Rating from vw\_school\_list;

**Take a screenshot showing the last SQL query and its results.**

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.

Graphical user interface, text

Description automatically generated

*Create Procedure UPDATE\_LEADERS\_SCORE (In in\_School\_ID , in\_Leader\_Score)*

*Begin*

*End ;*

**Take a screenshot showing the SQL query.**

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.

*Create Procedure UPDATE\_LEADERS\_SCORE(In in\_School\_ID int, in\_Leader\_Score int)*

*Begin*

*Update CHICAGO\_PUBLIC\_SCHOOLS set in\_Leader\_Score = Leaders\_Score*

*Where in\_School\_ID = in\_School\_ID;*

*End*

**Take a screenshot showing the SQL query.**

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.

*Create Procedure UPDATE\_LEADERS\_SCORE (In in\_School\_ID int, in in\_Leader\_Score int)*

*Begin*

*IF in\_Leader\_Score > 0 AND in\_Leader\_Score < 20 THEN*

*Update CHICAGO\_PUBLIC\_SCHOOLS*

*set Leader\_Score = in\_Leader\_Score,*

*Leaders\_Icon =’Very Weak’*

*Where School\_ID =in\_School\_ID*

*ELSEIF in\_Leader\_Score =20 and in\_Leader\_Score < 40 THEN*

*Update CHICAGO\_PUBLIC\_SCHOOLS*

*set Leader\_Score = in\_Leader\_Score*

*Leaders\_Icon =’Weak’*

*Where School\_ID =in\_School\_ID*

*ELSEIF in\_Leader\_Score =40 and in\_Leader\_Score <59 THEN*

*Update CHICAGO\_PUBLIC\_SCHOOLS*

*set Leader\_Score = in\_Leader\_Score,*

*Leaders\_Icon =’Average’*

*Where School\_ID =in\_School\_ID*

*ELSEIF in\_Leader\_Score =60 and in\_Leader\_Score <79 THEN*

*Update CHICAGO\_PUBLIC\_SCHOOLS*

*set Leader\_Score =in\_Leader\_Score ,*

*Leaders\_Icon =’Strong’*

*Where School\_ID =in\_School\_ID*

*ELSEIF in\_Leader\_Score =80 and in\_Leader\_Score <99 THEN*

*Update CHICAGO\_PUBLIC\_SCHOOLS*

*set Leader\_Score = in\_Leader\_Score*

*Leaders\_Icon =’Very Strong’*

*Where School\_ID =in\_School\_ID*

*ELSE*

*ENDIF*

*End;*

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

Hint 1: Remember that once a clause of the IF statement executes, no further checking occurs

and processing moves to the code below the IF statement.

Hint 2:

Your IF statement should have a structure similar to:

IF in\_Leader\_Score > 0 AND in\_Leader\_Score < 20 THEN

-- update icon for 0-19

ELSEIF in\_Leader\_Score < 40 THEN

-- update icon for 20-39

ELSEIF in\_Leader\_Score < 60 THEN

-- update icon for 40-59

ELSEIF in\_Leader\_Score < 80 THEN

-- update icon for 60-79

ELSEIF in\_Leader\_Score < 100 THEN

-- update icon for 80-99

END IF;

**Take a screenshot showing the SQL query.**

Question 4

* Run your code to create the stored procedure.

**Take a screenshot showing the SQL query and its results.**

* 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 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 fit any of the preceding categories.

Hint 1: You can add an ELSE clause to the IF statement which will only run if none of the previous conditions have been met.

*Create Procedure UPDATE\_LEADERS\_SCORE (In in\_School\_ID , in in\_Leader\_Score)*

*Begin*

*IF* @*in\_Leader\_Score > 0 AND* @*in\_Leader\_Score < 20 THEN*

*Update* CHICAGO\_PUBLIC\_SCHOOLS set in\_Leader\_Score = @*in\_Leader\_Score*

Leaders\_Icon =’Very Weak’

*Where in\_School\_ID =@in\_School\_ID;*

*ELSEIF* @*in\_Leader\_Score =20 and* @*in\_Leader\_Score < 40 THEN*

*Update* CHICAGO\_PUBLIC\_SCHOOLS set in\_Leader\_Score @*in\_Leader\_Score*

Leaders\_Icon =’Weak’

*Where in\_School\_ID =@in\_School\_ID;*

*ELSEIF* @*in\_Leader\_Score =40 and* @*in\_Leader\_Score <59 THEN*

*Update* CHICAGO\_PUBLIC\_SCHOOLS set in\_Leader\_Score =@*in\_Leader\_Score,*

Leaders\_Icon =’Average’

*Where in\_School\_ID =@in\_School\_ID;*

*ELSEIF* @*in\_Leader\_Score =60 and* @*in\_Leader\_Score <79 THEN*

*Update* CHICAGO\_PUBLIC\_SCHOOLS set in\_Leader\_Score =@*in\_Leader\_Score*

Leaders\_Icon =’Strong’

*Where in\_School\_ID =@in\_School\_ID;*

*ELSEIF* @*in\_Leader\_Score =80 and* @*in\_Leader\_Score <99 THEN*

*Update* CHICAGO\_PUBLIC\_SCHOOLS set in\_Leader\_Score =@*in\_Leader\_Score*

Leaders\_Icon =’Very Strong’

*Where in\_School\_ID =@in\_School\_ID;*

*ELSE*

*Roolback;*

*ENDIF*

*End;*

**Take a screenshot showing the SQL query.**

Question 2

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

Hint 1: Remember that as soon as any code inside the IF/ELSE IF/ELSE statements completes,

processing resumes after the END IF, so you can add your commit code there.

*Create Procedure UPDATE\_LEADERS\_SCORE (In in\_School\_ID , in in\_Leader\_Score)*

*Begin*

*IF* @*in\_Leader\_Score > 0 AND* @*in\_Leader\_Score < 20 THEN*

*Update* CHICAGO\_PUBLIC\_SCHOOLS set in\_Leader\_Score = @*in\_Leader\_Score*

Leaders\_Icon =’Very Weak’

*Where in\_School\_ID =@in\_School\_ID;*

*ELSEIF* @*in\_Leader\_Score =20 and* @*in\_Leader\_Score < 40 THEN*

*Update* CHICAGO\_PUBLIC\_SCHOOLS set in\_Leader\_Score @*in\_Leader\_Score*

Leaders\_Icon =’Weak’

*Where in\_School\_ID =@in\_School\_ID;*

*ELSEIF* @*in\_Leader\_Score =40 and* @*in\_Leader\_Score <59 THEN*

*Update* CHICAGO\_PUBLIC\_SCHOOLS set in\_Leader\_Score =@*in\_Leader\_Score,*

Leaders\_Icon =’Average’

*Where in\_School\_ID =@in\_School\_ID;*

*ELSEIF* @*in\_Leader\_Score =60 and* @*in\_Leader\_Score <79 THEN*

*Update* CHICAGO\_PUBLIC\_SCHOOLS set in\_Leader\_Score =@*in\_Leader\_Score*

Leaders\_Icon =’Strong’

*Where in\_School\_ID =@in\_School\_ID;*

*ELSEIF* @*in\_Leader\_Score =80 and* @*in\_Leader\_Score <99 THEN*

*Update* CHICAGO\_PUBLIC\_SCHOOLS set in\_Leader\_Score =@*in\_Leader\_Score*

Leaders\_Icon =’Very Strong’

*Where in\_School\_ID =@in\_School\_ID;*

*ENDIF*

*Commit;*

*End;*

**Take a screenshot showing the SQL query.**

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

Summary

You can now write advanced SQL statements to query data from multiple tables, to obscure sensitive data from users, and to control how information is updated in your tables.

Authors

[Lin Joyner](optional link to profile)

[Ramesh Sannareddy](optional link to profile)

Other Contributor

Rose Malcolm

Changelog

| **Date** | **Version** | **Changed by** | **Change Description** |
| --- | --- | --- | --- |
| 02-July-21 | 1.2 | Lakshmi Holla | Made changes in Hint section. |
| 27-Jan-21 | 1.1 | Rose Malcolm | Added instructions to go to Exercise 1 if tables already created and poplulated. |
| 05-Jun-20 | 1.0 | Rose Malcolm | Initial version created |
|  |  |  |  |
|  |  |  |  |

© IBM Corporation 2020. All rights reserved.