DB0201EN-PeerAssign-v5

January 18, 2023

Assignment: Notebook for Peer Assignment

1 Introduction

Using this Python notebook you will:

1. Understand three Chicago datasets
2. Load the three datasets into three tables in a Db2 database
3. Execute SQL queries to answer assignment questions

1.1 Understand the datasets

To complete the assignment problems in this notebook you will be using three datasets that are available on the city of Chicago's Data Portal:

1. Socioeconomic Indicators in Chicago
2. Chicago Public Schools
3. Chicago Crime Data

1.1.1 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

1.1.2 2. Chicago Public Schools

This dataset shows all school level performance data used to create CPS School Report Cards for the 2011-2012 school year. This dataset is provided by the city of Chicago's Data Portal.

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

1.1.3 3. 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

1.1.4 Download the datasets

This assignment requires you to have these three tables populated with a subset of the whole datasets.

In many cases the dataset to be analyzed is available as a .CSV (comma separated values) file, perhaps on the internet. Click on the links below to download and save the datasets (.CSV files):

- Chicago Census Data
- Chicago Public Schools
- Chicago Crime Data

NOTE: For the learners who are encountering issues with loading from .csv in DB2 on Firefox, you can download the .txt files and load the data with those:

- Chicago Census Data
- Chicago Public Schools
- Chicago Crime Data

NOTE: Ensure you have downloaded the datasets using the links above instead of directly from the Chicago Data Portal. The versions linked here are subsets of the original datasets and have some of the column names modified to be more database friendly which will make it easier to complete this assignment.

1.1.5 Store the datasets in database tables

To analyze the data using SQL, it first needs to be stored in the database.

While it is easier to read the dataset into a Pandas dataframe and then PERSIST it into the database as we saw in Week 3 Lab 3, it results in mapping to default datatypes which may not be optimal for SQL querying. For example a long textual field may map to a CLOB instead of a VARCHAR.

Therefore, **it is highly recommended to manually load the table using the database console LOAD tool, as indicated in Week 2 Lab 1 Part II**. The only difference with that lab is that in Step 5 of the instructions you will need to click on create "(+) New Table" and specify the name of the table you want to create and then click "Next".

Now open the Db2 console, open the LOAD tool, Select / Drag the .CSV file for the first dataset, Next create a New Table, and then follow the steps on-screen instructions to load the data. Name the new tables as follows:

1. **CENSUS_DATA**
2. **CHICAGO_PUBLIC_SCHOOLS**
3. **CHICAGO_CRIME_DATA**

1.1.6 Connect to the database

Let us first load the SQL extension and establish a connection with the database

The following required modules are pre-installed in the Skills Network Labs environment. However if you run this notebook commands in a different Jupyter environment (e.g. Watson Studio or Ananconda) you may need to install these libraries by removing the # sign before !pip in the code cell below.

[1]:
```
!pip install sqlalchemy
!pip install ibm_db_sa
!pip install sqlalchemy==1.3.9

!pip install ipython-sql
```

```
Requirement already satisfied: sqlalchemy in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (1.3.24)
Requirement already satisfied: ibm_db_sa in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (0.3.3)
Requirement already satisfied: sqlalchemy>=0.7.3 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from ibm_db_sa)
(1.3.24)
Collecting sqlalchemy==1.3.9
  Downloading SQLAlchemy-1.3.9.tar.gz (6.0 MB)
                           6.0/6.0 MB
82.3 MB/s eta 0:00:00:00:0100:01
  Preparing metadata (setup.py) … done
Building wheels for collected packages: sqlalchemy
  Building wheel for sqlalchemy (setup.py) … done
  Created wheel for sqlalchemy:
filename=SQLAlchemy-1.3.9-cp37-cp37m-linux_x86_64.whl size=1159122
sha256=cb1099c8bb0770fc0b9d8fc1a134981c82bc418ad3c306dcf2b73ba05fab4fb1
  Stored in directory: /home/jupyterlab/.cache/pip/wheels/ef/95/ac/c232f83b41590
0c26553c64266e1a2b2863bc63e7a5d606c7e
Successfully built sqlalchemy
Installing collected packages: sqlalchemy
  Attempting uninstall: sqlalchemy
    Found existing installation: SQLAlchemy 1.3.24
    Uninstalling SQLAlchemy-1.3.24:
      Successfully uninstalled SQLAlchemy-1.3.24
Successfully installed sqlalchemy-1.3.9
Requirement already satisfied: ipython-sql in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (0.3.9)
Requirement already satisfied: ipython>=1.0 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from ipython-
sql) (7.33.0)
Requirement already satisfied: ipython-genutils>=0.1.0 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from ipython-
sql) (0.2.0)
```

```
Requirement already satisfied: prettytable in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from ipython-
sql) (3.5.0)
Requirement already satisfied: six in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from ipython-
sql) (1.16.0)
Requirement already satisfied: sqlalchemy>=0.6.7 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from ipython-
sql) (1.3.9)
Requirement already satisfied: sqlparse in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from ipython-
sql) (0.4.3)
Requirement already satisfied: jedi>=0.16 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
ipython>=1.0->ipython-sql) (0.18.2)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
ipython>=1.0->ipython-sql) (3.0.33)
Requirement already satisfied: pexpect>4.3 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
ipython>=1.0->ipython-sql) (4.8.0)
Requirement already satisfied: pickleshare in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
ipython>=1.0->ipython-sql) (0.7.5)
Requirement already satisfied: traitlets>=4.2 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
ipython>=1.0->ipython-sql) (5.6.0)
Requirement already satisfied: backcall in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
ipython>=1.0->ipython-sql) (0.2.0)
Requirement already satisfied: decorator in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
ipython>=1.0->ipython-sql) (5.1.1)
Requirement already satisfied: pygments in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
ipython>=1.0->ipython-sql) (2.13.0)
Requirement already satisfied: setuptools>=18.5 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
ipython>=1.0->ipython-sql) (65.5.1)
Requirement already satisfied: matplotlib-inline in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
ipython>=1.0->ipython-sql) (0.1.6)
Requirement already satisfied: wcwidth in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
prettytable->ipython-sql) (0.2.5)
Requirement already satisfied: importlib-metadata in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
prettytable->ipython-sql) (4.11.4)
```

```
Requirement already satisfied: parso<0.9.0,>=0.8.0 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
jedi>=0.16->ipython>=1.0->ipython-sql) (0.8.3)
Requirement already satisfied: ptyprocess>=0.5 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
pexpect>4.3->ipython>=1.0->ipython-sql) (0.7.0)
Requirement already satisfied: typing-extensions>=3.6.4 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from importlib-
metadata->prettytable->ipython-sql) (4.4.0)
Requirement already satisfied: zipp>=0.5 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from importlib-
metadata->prettytable->ipython-sql) (3.11.0)
```

[2]:
```
#import ibm_db
#import ibm_db_sa
#import sqlalchemy
#from sqlalchemy import *
```

[2]:
```
%reload_ext sql
```

In the next cell enter your db2 connection string. Recall you created Service Credentials for your Db2 instance in first lab in Week 3. From your Db2 service credentials copy everything after db2:// (except the double quote at the end) and paste it in the cell below after ibm_db_sa://

[4]:
```
# Remember the connection string is of the format:
# %sql ibm_db_sa://my-username:my-password@my-hostname:my-port/my-db-name?
 ↪security=SSL
# Enter the connection string for your Db2 on Cloud database instance below
%sql ibm_db_sa://zbc64279:b9gIUeFzvXp4wjtO@815fa4db-dc03-4c70-869a-a9cc13f33084.
 ↪bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb?security=SSL
```

[4]:
```
'Connected: zbc64279@bludb'
```

1.2 Problems

Now write and execute SQL queries to solve assignment problems

1.2.1 Problem 1

Find the total number of crimes recorded in the CRIME table.

[5]:
```
%sql select count(*) from chicago_crime_data
```

```
 * ibm_db_sa://zbc64279:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1o
d8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

[5]:
```
[(533,)]
```

[6]:
```
conda install -c conda-forge ipython-sql
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 4.12.0
  latest version: 22.11.1

Please update conda by running

    $ conda update -n base -c defaults conda



## Package Plan ##

  environment location: /home/jupyterlab/conda/envs/python

  added / updated specs:
    - ipython-sql


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    ca-certificates-2022.12.7  |       ha878542_0         143 KB  conda-forge
    certifi-2022.12.7          |     pyhd8ed1ab_0         147 KB  conda-forge
    ipython-sql-0.3.9          |  pyhd8ed1ab_1004          18 KB  conda-forge
    ipython_genutils-0.2.0     |             py_1          21 KB  conda-forge
    prettytable-3.6.0          |     pyhd8ed1ab_0          28 KB  conda-forge
    sqlalchemy-1.3.24          |   py37h540881e_1         1.8 MB  conda-forge
    sqlparse-0.4.3             |     pyhd8ed1ab_0          35 KB  conda-forge
    ------------------------------------------------------------
                                           Total:         2.2 MB

The following NEW packages will be INSTALLED:

  ipython-sql        conda-forge/noarch::ipython-sql-0.3.9-pyhd8ed1ab_1004
  ipython_genutils   conda-forge/noarch::ipython_genutils-0.2.0-py_1
  prettytable        conda-forge/noarch::prettytable-3.6.0-pyhd8ed1ab_0
  sqlalchemy         conda-forge/linux-64::sqlalchemy-1.3.24-py37h540881e_1
  sqlparse           conda-forge/noarch::sqlparse-0.4.3-pyhd8ed1ab_0

The following packages will be UPDATED:
  ca-certificates                      2022.9.24-ha878542_0 -->
2022.12.7-ha878542_0
  certifi                            2022.9.24-pyhd8ed1ab_0 -->
2022.12.7-pyhd8ed1ab_0



Downloading and Extracting Packages
prettytable-3.6.0    | 28 KB     | ##################################### | 100%
certifi-2022.12.7    | 147 KB    | ##################################### | 100%
sqlparse-0.4.3       | 35 KB     | ##################################### | 100%
ipython-sql-0.3.9    | 18 KB     | ##################################### | 100%
ca-certificates-2022 | 143 KB    | ##################################### | 100%
sqlalchemy-1.3.24    | 1.8 MB    | ##################################### | 100%
ipython_genutils-0.2 | 21 KB     | ##################################### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done

Note: you may need to restart the kernel to use updated packages.
```

1.2.2 Problem 2

List community areas with per capita income less than 11000.

```
[7]: %sql SELECT COMMUNITY_AREA_NAME FROM CENSUS_DATA WHERE PER_CAPITA_INCOME <␣
      ↪11000;
```

```
 * ibm_db_sa://zbc64279:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1o
d8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

```
[7]: [('West Garfield Park',),
      ('South Lawndale',),
      ('Fuller Park',),
      ('Riverdale',)]
```

1.2.3 Problem 3

List all case numbers for crimes involving minors?(children are not considered minors for the purposes of crime analysis)

```
[8]: %sql SELECT DISTINCT CASE_NUMBER FROM CHICAGO_CRIME_DATA WHERE DESCRIPTION LIKE␣
      ↪'%MINOR%'
```

```
 * ibm_db_sa://zbc64279:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1o
d8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

```
[8]: [('HK238408',), ('HL266884',)]
```

1.2.4 Problem 4

List all kidnapping crimes involving a child?

```
[9]: %sql SELECT DISTINCT CASE_NUMBER, PRIMARY_TYPE, DATE, DESCRIPTION FROM␣
      ↪CHICAGO_CRIME_DATA \
     WHERE PRIMARY_TYPE='KIDNAPPING'
```

```
 * ibm_db_sa://zbc64279:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1o
d8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

```
[9]: [('HN144152', 'KIDNAPPING', datetime.datetime(2007, 1, 26, 10, 5), 'CHILD
     ABDUCTION/STRANGER')]
```

1.2.5 Problem 5

What kinds of crimes were recorded at schools?

```
[10]: %sql SELECT DISTINCT(PRIMARY_TYPE), LOCATION_DESCRIPTION FROM␣
       ↪CHICAGO_CRIME_DATA \
      WHERE LOCATION_DESCRIPTION LIKE '%SCHOOL%'
```

```
 * ibm_db_sa://zbc64279:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1o
d8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

```
[10]: [('PUBLIC PEACE VI', 'SCHOOL, PRIVATE, BUILDING'),
       ('BATTERY', 'SCHOOL, PUBLIC, BUILDING'),
       ('NARCOTICS', 'SCHOOL, PUBLIC, BUILDING'),
       ('PUBLIC PEACE VI', 'SCHOOL, PUBLIC, BUILDING'),
       ('ASSAULT', 'SCHOOL, PUBLIC, GROUNDS'),
       ('BATTERY', 'SCHOOL, PUBLIC, GROUNDS'),
       ('CRIMINAL DAMAGE', 'SCHOOL, PUBLIC, GROUNDS'),
       ('CRIMINAL TRESPA', 'SCHOOL, PUBLIC, GROUNDS'),
       ('NARCOTICS', 'SCHOOL, PUBLIC, GROUNDS')]
```

1.2.6 Problem 6

List the average safety score for each type of school.

```
[12]: %sql SELECT "ELEMENTARY__MIDDLE__OR_HIGH_SCHOOL", AVG(SAFETY_SCORE)␣
       ↪AVERAGE_SAFETY_SCORE FROM CHICAGO_PUBLIC_SCHOOLS GROUP BY␣
       ↪"ELEMENTARY__MIDDLE__OR_HIGH_SCHOOL";
```

```
 * ibm_db_sa://zbc64279:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1o
d8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

```
[12]: [('ES', 48), ('HS', 47), ('MS', 50)]
```

1.2.7 Problem 7

List 5 community areas with highest % of households below poverty line

```
[13]: %sql SELECT COMMUNITY_AREA_NAME, PERCENT_HOUSEHOLDS_BELOW_POVERTY FROM␣
       ↪CENSUS_DATA ORDER BY PERCENT_HOUSEHOLDS_BELOW_POVERTY DESC LIMIT 5 ;
```

```
 * ibm_db_sa://zbc64279:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1o
d8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

```
[13]: [('Riverdale', Decimal('56.5')),
       ('Fuller Park', Decimal('51.2')),
       ('Englewood', Decimal('46.6')),
       ('North Lawndale', Decimal('43.1')),
       ('East Garfield Park', Decimal('42.4'))]
```

1.2.8 Problem 8

Which community area is most crime prone?

```
[14]: %%sql
      SELECT COMMUNITY_AREA_NUMBER ,COUNT(COMMUNITY_AREA_NUMBER) AS FREQUENCY
      FROM CHICAGO_CRIME_DATA
      GROUP BY COMMUNITY_AREA_NUMBER
      ORDER BY COUNT(COMMUNITY_AREA_NUMBER) DESC
      LIMIT 1;
```

```
 * ibm_db_sa://zbc64279:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1o
d8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

```
[14]: [(25, 43)]
```

Double-click **here** for a hint

1.2.9 Problem 9

Use a sub-query to find the name of the community area with highest hardship index

```
[15]: %sql SELECT COMMUNITY_AREA_NAME FROM  CENSUS_DATA WHERE HARDSHIP_INDEX =␣
       ↪(SELECT MAX(HARDSHIP_INDEX) FROM CENSUS_DATA);
```

```
 * ibm_db_sa://zbc64279:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1o
d8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

```
[15]: [('Riverdale',)]
```

1.2.10 Problem 10

Use a sub-query to determine the Community Area Name with most number of crimes?

```
[16]: %%sql
SELECT community_area_name FROM CENSUS_DATA
WHERE COMMUNITY_AREA_NUMBER = (SELECT COMMUNITY_AREA_NUMBER FROM␣
 ↪CHICAGO_CRIME_DATA
    GROUP BY COMMUNITY_AREA_NUMBER
    ORDER BY COUNT(COMMUNITY_AREA_NUMBER) DESC
    LIMIT 1)
LIMIT 1;
```

```
 * ibm_db_sa://zbc64279:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1o
d8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

```
[16]: [('Austin',)]
```

1.3 Author(s)

Hima Vasudevan

Rav Ahuja

Ramesh Sannreddy

1.4 Contribtuor(s)

Malika Singla

1.5 Change log

Date	Version	Changed by	Change Description
2021-11-17	2.6	Lakshmi	Updated library
2021-05-19	2.4	Lakshmi Holla	Updated the question
2021-04-30	2.3	Malika Singla	Updated the libraries
2021-01-15	2.2	Rav Ahuja	Removed problem 11 and fixed changelog
2020-11-25	2.1	Ramesh Sannareddy	Updated the problem statements, and datasets
2020-09-05	2.0	Malika Singla	Moved lab to course repo in GitLab
2018-07-18	1.0	Rav Ahuja	Several updates including loading instructions
2018-05-04	0.1	Hima Vasudevan	Created initial version

##