University of Victoria
Engineering & Computer Science Co-op
Work Term Report
Term Fall 2019

Generating an estimated 2019 CHASS census by using SQL and Python

University of Victoria
Advancement Services and Information Technology
Victoria, BC, Canada

LOK IN LIANG
V00104104
Work Term 1
Master of Engineering in Telecommunications and Information Security
example5@104.com.tw
15 December, 2019

In partial fulfillment of the academic requirements of this co-op term

Letter of Transmittal

LOK IN LIANG

15 December, 2019

Dr. Imen Bourguiba, Co-op Coordinator
Faculty of Engineering University of Victoria
Victoria, B.C.
V8P 5C2

Please accept the following Co-op term report entitled “Generating an estimated 2019 CHASS census by using SQL and Python” designed for seeking the potential solutions for helping my Coop organization to improve their database and efficiency.

This report is the result of work completed at the University of Victoria, Department of Advancement Services and Information Technology, Canada. This is my first co-op work term as an UVic student, this position mainly focuses on data analytics, modeling, and visualization of the UVic Alumni databases by using the statistical, machine, and deep learning methods. The goal is to find a group of alumni who have a high probability of donating the UVic.

Through the course of the term, I was given the opportunity to learn much about data cleaning, processing, machine learning, deep learning, and automatically programming for running repeated job. I feel that this knowledge will be helpful in future work terms, and in my career.

I would like to thank my supervisor Atsuko Umeki, my manager Gregory Churchill, and the software developer Chris Coey for being very patient to give me directions on work and kindly sharing all their knowledge, as well as all my colleagues and friends who were always willing to offer help during my internship.

Sincerely,

Lok In Liang

Contents

List of Figures

List of Tables

Executive Summary

This report is talking how to find a reliable data from the internet, data cleaning, data processing, Python skill, SQL skill, and having a program to run it automatically for avoiding a lot of repeated work during developing. The main goal of this report is to illustrate how the purchased wealth data for a private data company is replaced by an open-source platform and update it for further use in the UVic Alumni department.

This report will descript the detail about the census data from Statistics Canada, the Python package of running SQL instruction, and automatic run programming.

Glossary

SQL	: Structured Query Language
CHASS	: Computing in the Humanities and Social Sciences
PCCF	: The Postal Code Conversion File
CPC	: Canada Post Corporation
Stat. CA	: Statistics Canada
CREA	: Canadian real restate association
HPI	: House price index
CMHC-SCHL	: Canada Mortgage and Housing Corporation - Scholastic Corp

Introduction

The goal of the University of Victoria Alumni Department is keeping the connection with Alumni, organizing events, benefits, and post news to let Alumni feel close with us and trying to get a donation from alumni. Also, helping the student to actively recruit and retain outstanding students from diverse regions and backgrounds and to remove all barriers to admission and retention other than academic and creative potential. I am working as a Data Analyst in the Information Technology group, and my supervisor is a Data Analytics Officer. This position mainly focuses on data analytics, modeling, and visualization of the UVic Alumni databases by using the statistical, machine, and deep learning methods. The goal is to find a group of alumni who have a high probability of donating the UVic. There is a wealth data which will be purchased every year in this department. It will spend approximately CAD 30,000 for buying this data. Unfortunately, this data is cost a lot and not fully know how they collect those data. It is essential to have our open-source database and no need to depend on others.

Furthermore, building a database which means we can edit it, change it by ourselves for customizing whatever we want. Having our open-source wealth database, not only can reduce the cost of spending in another field in this department, but also can maintain, edit, and customize it on our own for doing deeper and further development. The rest of this paper is organized as follows. The rest of section I (Introduction) presents the background information and problem definition. In section II (Discussion), showing how to generate an estimated database from census data of Canada, and how to make all things faster and efficient. In section III and IV, having a conclusion and recommendation.

Background information

Statistics Canada [1]

Statistics Canada produces statistics that help Canadians better understand their country—its population, resources, economy, society, and culture. In addition to conducting a Census every five years, there are about 350 active surveys on virtually all aspects of Canadian life. In Canada, providing statistics is a federal responsibility. As Canada's central statistical office, Statistics Canada is legislated to serve this function for the whole of Canada and each of the provinces and territories. Objective statistical information is vital to an open and democratic society. It provides a solid foundation for informed decisions by elected representatives, businesses, unions, and non-profit organizations, as well as individual Canadians. As a member of the United Nations Statistical Commission, Statistics Canada endorses the Fundamental principles of official statistics.

CHASS Canadian Census data [2]

Computing maintains this collection of on-line databases and custom-built search and retrieval programs in the Humanities and Social Sciences (CHASS) at the University of Toronto. Census data is commonly used for research, business marketing, and planning as well as a base for sampling surveys. Profile files at enumeration areas, census tracts, census divisions, census subdivisions, federal electoral districts, or provincials' levels, some data are going back to 1961.

PCCF [3]

The Postal Code Conversion File (PCCF) is a digital file that provides a correspondence between the Canada Post Corporation (CPC) six-character postal code and Statistics Canada's standard

geographic areas for which census data and other statistics are produced. Through the link between postal codes and standard geographic areas, the PCCF permits the integration of data from various sources.

The geographic coordinates, which represent the standard geostatistical areas linked to each postal code on the PCCF, are commonly used to map the distribution of data for spatial analysis (e.g., clients, activities). The location information is a powerful tool for marketing, planning, or research purposes.

Problem definition

1. The purchased wealth data cost too much for each year. It can be replaced by other open-source database and edited it.
2. There are a lot of repeated steps during the developing process. It can be improved by using the Python package to build up the developing efficiency.
3. Some database has to update it every day. It is necessary to have a program to maintain the everyday update.

Discussion

The purchased wealth data

The Advancement Services department at UVic purchased the wealth data from Environics (Toronto) for the past couple of years. As shown in Table 1, several values are essential in the purchased wealth data.

Table 1: Features of the purchased wealth data form Environics

- Location
- Income
- Liquid Assets
- Real Estate
- Debt
- Charitable contributions

  Etc.

Having a reliable and trusted source of data to replace the purchased Environics wealth data is essential. Therefore, statistics Canada is the best choice for providing good quality of data. The CHASS census data from Statistics Canada offer many different data, which is highly similar to the purchased wealth data, depending on each location in Canada, as shown in Table 2.

Table 2: Features of the CHASS census data

- Total income (Median / Average) of households
- After-tax income (Median / Average) of among recipients
- Monthly shelter costs (Median / Average)
- After-tax liquid assets (Median / Average) of economic families
- Average value of dwellings
- (Median / Average) annual shelter costs for rented dwellings

  Etc.

By comparing Table 1 and Table 2, the income from purchased data can be replaced by the total income (median/average) of households; then, the real estate can be replaced by the average

value of dwellings. Also, liquid assets can be calculated by average income minus the average annual shelter costs. Lastly, the regional charitable contributions data can be found on the Statistic Canada.

Most importantly, the CHASS census data is released every four years. The recent release was in 2016. Therefore, it is necessary to update the census on our own for generating the estimated census data for analyzing. It can be updated by inflation and the real estate rate. For example, the income, liquid assets, debt, those are depending on the inflation rate of each year. Also, real estate value can be updated by the real estate rate. Then, the location information can be replaced by PCCF. PCCF includes postal code, province or territory code, census subdivision name, latitude, longitude, etc. It can be used for classifying alumni or students by Canadian location. The census subdivision unique identifier is used as the GEO ID for marking the location information. There are three parts of this identifier, here is the example of the code 3512002, and first two digits is the province or territory code, 35 in Ontario. Then, the next two digits are the census division code, 3512 is the Hastings County in Ontario. Lastly, the last three digits are the census subdivision's code, 002 is the Deseronto Town. PCCF can be used for matching the postal code and census sub-division code. Therefore, it is a convenience to merge with the Alumni database for knowing their location.

Fortunately, the latest inflation rate in 2019, which is the consumer price index and charitable contributions data, can be found in Statistics Canada, and PCCF can be found in the ABACUS Licensed Data Collection Dataverse. Lastly, real estate rate can be found in different sources. It

can mainly get it from the Canadian real restate association (CREA) [4] MLS House price index (HPI). However, CREA has not provided all provinces and main cities of real estate rate. Therefore, some locations of real estate are found in other sources like CMHC-SCHL from Canada and RBC Housing Forecast report.

The objective of matching the census data and PCCF data is to build in-house wealth data for fundraising operations.

Integration

For integrating the data to generate the estimated version of CHASS census data, Python and SQL provide the power and convenience packages, functions, and tools. First of all, data processing and cleaning by using Python. For example, calculating the mean value of the full-year customer prices index and the percentage change of the index for each year. Having string processing and cleaning. Also, the same cleaning and processing procedure for real estate and charitable donors. Then, import the cleaned data to the department SQL server for merging several databases into one, such as inflation, real estate rate, and charitable donor data, have to join with the PCCF. The "join" instruction in SQL is more powerful and faster than using Python, therefore, there are two ways processing for the data. It is not friendly and convenient to process the data in this step. There is a lot of repeated work that wastes too much time. It is better to have an integration program for doing all the repeated work and spend the time thinking of innovation thing. As shown in the steps below, for example of how to get the inflation rate with location information,

1. Download the 'Consumer Price Index, monthly' CSV data in Statistics Canada

2. Generating a Python code for data cleaning and processing
3. Output the cleaned data into CSV file
4. Import the output CSV file to the department SQL server
5. Creating a SQL instruction (left join) for merging the PCCF with the inflation rate

The third and fourth steps are repeated steps. Additionally, these steps easy to have small errors such as failing to import CSV / excel file, a human mistake for selecting the wrong source/destination file type. When the data is updated, it needs to import the data to SQL server again. When we are still developing, it is tedious to work with many repeated jobs. Finally, Python has a lot of grateful package for making life simple and easy. There is two Python package (sqlalchemy and pyodbc) that are used for import or run SQL code by Python.

SQL Package for Python

Pyodbc [5]

Pyodbc is an open-source Python module that makes accessing ODBC databases simple. This Python package is used for connecting, reading, and executing the SQL query. Before doing SQL jobs, it has to create a connection first. The example code below:

‘conn = pyodbc.connect(Driver= {SQL Server Native Client 11.0}, Server = [address], Database = [name], Trusted_Connection='yes')’

There is a function that can read a SQL query and transform to dataframe. ‘sql_to_df = pd.read_sql(SQL_query, conn)’. When the program is run this function, it can be edited by the pandas package.

Also, pyodbc package provides a function for executing SQL query. The example code below:

‘cursor = conn.cursor()’ # generating a connection cursor

‘cursor.execute("SELECT TOP * FROM SQL_DATABASE”)’

When pyodbc package is being in use, do not forget to have a ‘conn.commit()’. It is important to call this method after every transaction that modifies data for tables that use transactional storage engines. Finally, please run ‘conn.close()’ at the end of the Python code for killing the connection between Python code and SQL server.

SQLAlchemy [6]

SQLAlchemy provides a nice “Pythonic” way of interacting with databases. So rather than dealing with the differences between specific dialects of traditional SQL such as MySQL or PostgreSQL or Oracle, you can leverage the Pythonic framework of SQLAlchemy to streamline your workflow and more efficiently query your data.

SQLAlchemy is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL.

There is a great function that is used in this package which is “dataframe.to_sql”. Before using this function, it has to create a SQL engine for connecting to SQL server first. The example code below:

SQL_engine =

sqlalchemy.create_engine('mssql+pyodbc://%s/%s?driver=%s&trusted_connection=yes'

%(SQL_Server, SQL_Database, SQL_Driver_engine))

‘mssql+pyodbc://’ means that using two packages, which is Microsoft SQL and pyodbc for connection of SQL. Then, the link includes the SQL database information such as address, SQL database name, and driver.

When the SQL engine is established, the ‘dataframe.to_sql’ can be used for changing the dataframe data into sql format and import to the SQL server. The example code below:

‘df.to_sql(name = ‘CHASS’, schema = [schema], con = SQL_engine, if_exists='replace', index = False)’

Combining these two packages by using Python programming language, it can reduce a lot of time. Additionally, when the SQL instruction includes in the Python code, some parts of the SQL instruction can be replaced by a variable, as shown in Figure 1, then processing by Python as we know about the logic of Python and SQL. Python can process the data row by row. However, the logic of SQL is different. SQL having a global view of the whole data. Therefore, having both advantages together for processing a database is useful for analyzing data. It can apply the Python function to SQL instruction. That can make the whole process more flexible and powerful.

```
SQL_query = \
"""
SELECT main_data.[Record_ID]
    ,coalesce([{first_year}],0) as [{file_name}_{first_year}]
    ,coalesce([{second_year}],0) as [{file_name}_{second_year}]
    ,coalesce([{third_year}],0) as [{file_name}_{third_year}]
    ,coalesce([{fourth_year}],0) as [{file_name}_{fourth_year}]
    ,coalesce([{fifth_year}],0) as [{file_name}_{fifth_year}]
    ,coalesce([{sixth_year}],0) as [{file_name}_{sixth_year}]
    ,coalesce([{seventh_year}],0) as [{file_name}_{seventh_year}]
    ,coalesce([{eighth_year}],0) as [{file_name}_{eighth_year}]
    ,coalesce([{ninth_year}],0) as [{file_name}_{ninth_year}]
    ,coalesce([{tenth_year}],0) as [{file_name}_{tenth_year}]
FROM [{SQL_Database}].[{SQL_schema}].[{SQL_name}] as main_data
left outer join
[{SQL_Database}].[{SQL_schema}].[{file_name}] as second_data
on main_data.Record_ID = second_data.Record_ID
order by Record_ID
""".format( SQL_Database = SQL_Database,
            SQL_schema   = SQL_schema,
            SQL_name     = SQL_name,
            file_name    = files_name[int(index)],
            first_year   = years[0], second_year = years[1],
            third_year   = years[2], fourth_year = years[3],
            fifth_year   = years[4], sixth_year  = years[5],
            seventh_year = years[6], eighth_year = years[7],
            ninth_year   = years[8], tenth_year  = years[9])
```

Figure 1: SQL instruction in Python program

When the inflation, real estate rate, and charitable contributions data, these data can be used for updating the latest version of the CHASS census data. There is an around four years gap of the newest database, then capturing the four years rate of inflation and real estate rate and multiplying them to the specific columns of the census database. For further use, if we would like to generate an estimated census database version of 2020, downloading all databases from Statistic Canada and importing the source database to the Python program. The Python program will help us to generate the estimated census data to the department SQL server.

Although combining Python and SQL is helps a lot, there is one more condition that should be improved. Alumni data will update every single day; there is a lot of students become alumni, especially during convocation period. Also, the department of the alumni has a huge formatted dataset depend on the alumni data. It is necessary to have the up-to-date dataset every day for having a good quality and completion data. However, it is not a good idea to update the dataset manually every day. Firstly, it is tedious with a lot of repeated jobs; secondly, when the dataset has to use instantly, it has to wait for some time to process the data; lastly, it is easy for a human to forget to update it manually.

For this situation, it would be great to have an automatic program to help us run programs every day. At this time, Python always provides an excellent package to the World, as shown in Figure 2.

```
from datetime import datetime, timedelta
import schedule
import time
import Generate_All_features_from_SQL

print("Schedule everyday at 14:17 do Generate_All_features_from_SQL.run_all")
schedule.every().day.at("14:17").do(Generate_All_features_from_SQL.run_all)

while True:
    schedule.run_pending()
    time.sleep(1)
```

Figure 2: Schedule package for Python program

This package called schedule [7]; it is an easy and friendly function, just like using English to talk to the computer. In this program, is scheduled to run the function of 'run_all' in the Python code of 'Generate_All_features_from_SQL' every day at 14:17. Therefore, we no need to care about

this program anymore because the computer acts like our assistant to help us maintain the repeated job. Except for this automatically program, having a menu within a plan is right for the user to understand what is going on of this program, as shown in Figure 3.

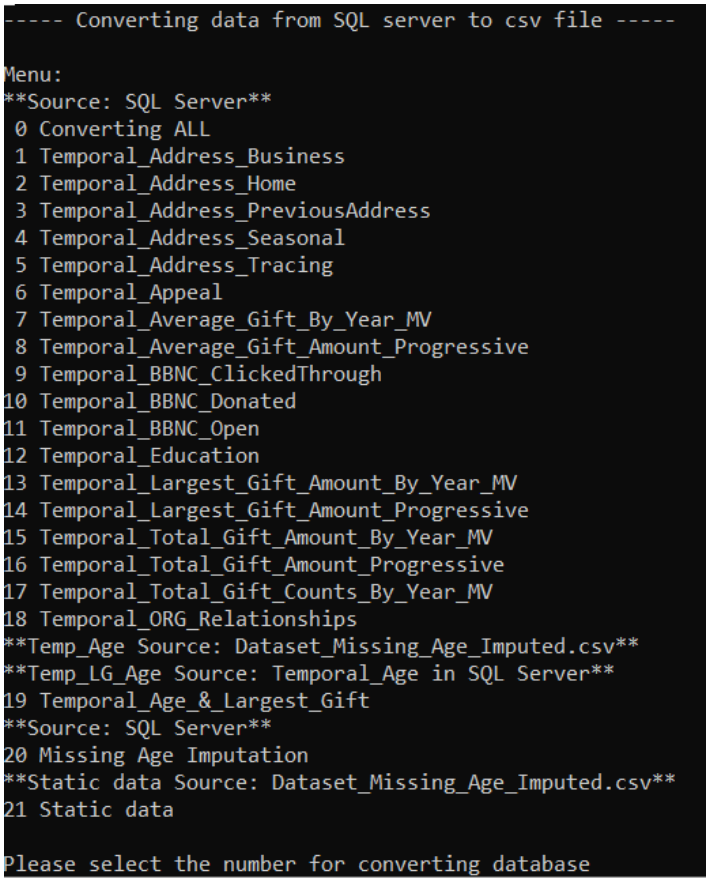

Figure 3: Updating database with a menu in Python

This program is for updating the database; it can select only one dataset to update or update all. Therefore, another user no need to understand the Python code before using this program because it will provide a user-friendly menu.

Conclusions

The estimated version of the CHASS census data used inflation, real estate rate, and charitable donor data to update it. Since the source of the census data, inflation, and real estate rate are from Statistic Canada, which is a reliable data source, the purchased wealth data can be covered by the estimated version of CHASS census data. Additionally, Statistic Canada descript how they get the data and have a detailed description of all data features. Also, if we have any questions about the data, Statistic Canada will quickly give us the result and detailed description. On the one hand, it can be improved the developing efficiency and reducing the tedious repeated step for checking and updating the database by combining the SQL in Python. On the other hand, advancing the automatically program by using the schedule program is suitable for the data that need to update it every day.

Recommendations

For the estimated CHASS census data, it is recommended to have an all-in-one program to generate the inflation, real estate rate, and charitable donor data and merge all databases to the alumni database. Currently, inflation, real estate rate, and charitable donor are not running in the same program; they are all separated. Also, it needs to download the database from Statistic Canada manually. For the recommendations in the future,

1. Python code for downloading the database from Statistic Canada (HTTP request)
2. Combining all programs into once, having a one-click program and generating all databases
3. Keep using the SQL package for Python

References

[1] “Home page - Statistics Canada,” Statistics Canada: Canada's national statistical agency, 17-Dec-2019. [Online]. Available: https://www.statcan.gc.ca/eng/start.

[2] “Canadian Census Analyser,” Canadian Census Analyser, 17-Dec-2019. [Online]. Available: http://datacentre.chass.utoronto.ca/census/

[3] “PCCF (Postal Code Conversion File): Canadian Research Data Centre Network,” PCCF (Postal Code Conversion File) | Canadian Research Data Centre Network. [Online]. Available: https://crdcn.org/datasets/pccf-postal-code-conversion-file.

[4] “Canadian Real Estate Association,” CREA. [Online]. Available: https://www.crea.ca/.

[5] Mkleehammer, “mkleehammer/pyodbc,” GitHub. [Online]. Available: https://github.com/mkleehammer/pyodbc/wiki.

[6] “The Python SQL Toolkit and Object Relational Mapper,” SQLAlchemy. [Online]. Available: https://www.sqlalchemy.org/.

[7] “schedule,” PyPI. [Online]. Available: https://pypi.org/project/schedule/.