Alperen Ugus – Murat Tuter 12/03/2019

CSCI 403

Database Management

Final Project

1. Dataset

In this project, we used population and GDP (Gross Domestic Product) per capita data from 1960 to 2018 for G7 countries consisting of Canada, France, Germany, Italy, Japan, the United Kingdom and the United States. Since we pulled the data from World Bank Open Data archive there was no restrictions on it.

The interesting thing about the data is that we could use it to predict future life standards of G7 countries using one of the machine learning technics: Linear Regression. Using past 59 years of data for each country, we tried to predict the situation in 2060.

2. Approach

2.1 Bulk-Loading (See bulkLoad.py and tables.psql)

First of all, we have downloaded the data from the World Bank Open Data archive as csv files. One of the files had country names and their populations from 1960 to 2018 and the other one had country names and their GDP per capita for the same years.

We needed to bulk-load this data into our database. To do that, first we created 2 tables for each csv file according to their column names. Then, we used a library named "psycopg2". We used "copy_from" method from that library cursor object to be able to copy data from the csv file to the database.

While copying the data, we dealt with many situations such as not all the data available for each country for every year. So, we needed empty cells to be converted to null values. Also, some countries had long names which were separated by commas and those were problematic for the csv structure. To overcome this problem, we erased all the commas from country names.

2.2 Extracting Data for G7 Countries (See tables.psql)

We copied the data to the database but that is not enough because we do not need to have all the countries data instead, we need 7 specific country data. So, we decided to create 2 new tables named g7_gdp and g7_population. We wrote a PostgreSQL script to extract data from our main tables for these 7 countries.

2.3 Linear Regression (See linearRegression.py)

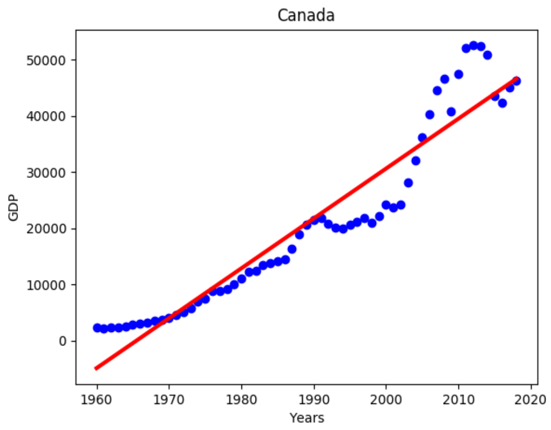
Linear Regression is a famous technic in machine learning that aims to fit a line to spread data points by defining an error function and minimizing it. We used "sklearn" library to use Linear Regression module. This module needed x and y values for the data. So, we created the dates array using python range function. Then for the data points, we created select queries that takes country name as parameter. Using sklearn library functions we fitted a line to the data points for each country and showed it using matplotlib. We saved all the figures for future analysis.

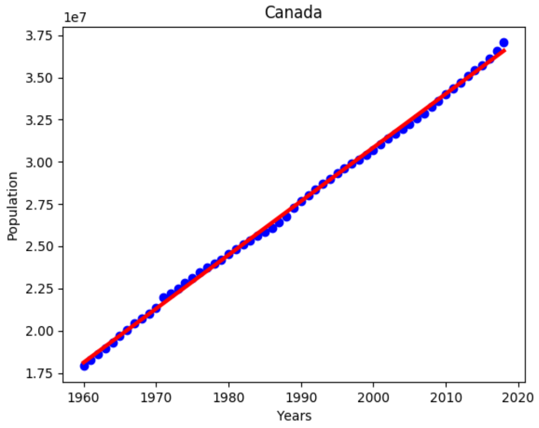
2.3 Predictions (See linearRegression.py)

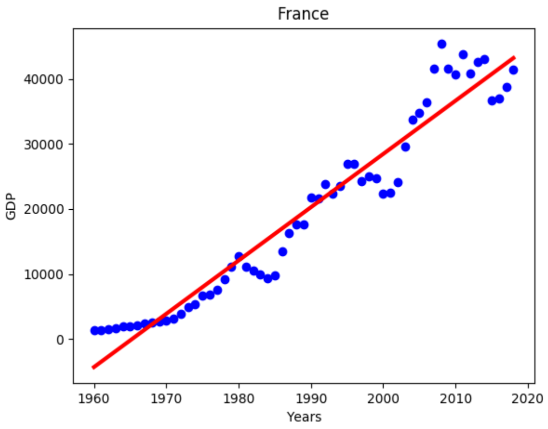
After training the linear model that takes year as a parameter, now we are capable of predicting future values for both GDP and populations of countries. The only thing we need to do is to use the models to predict results for countries by giving it a date!

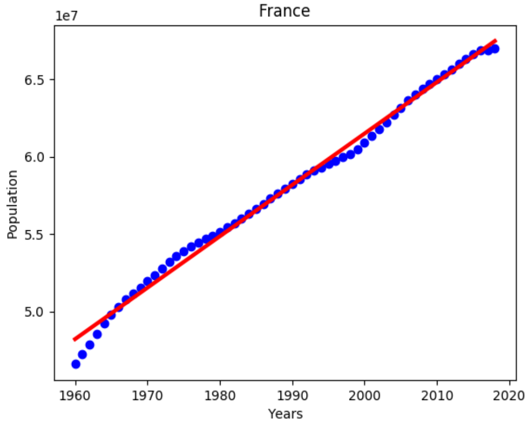
3. Results

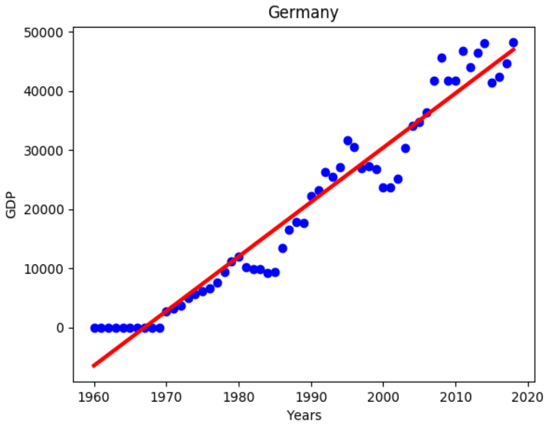
3.1 Linear Regression Lines

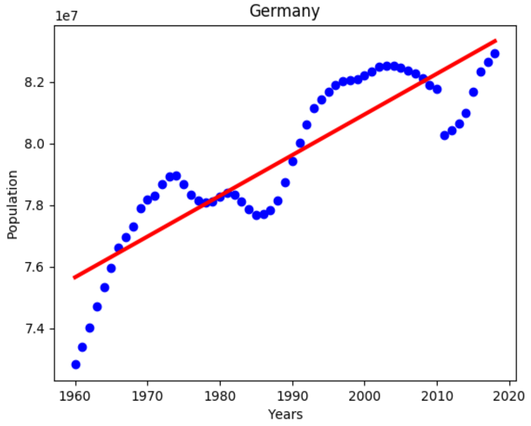


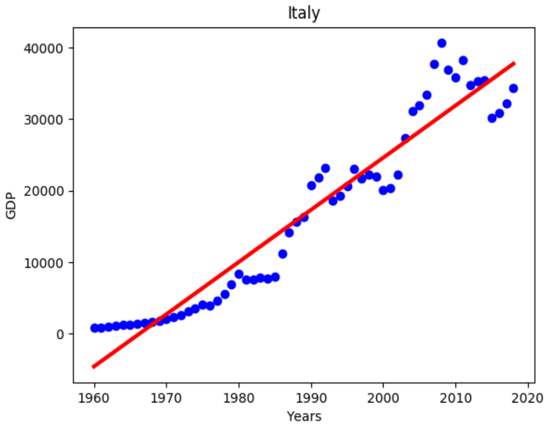


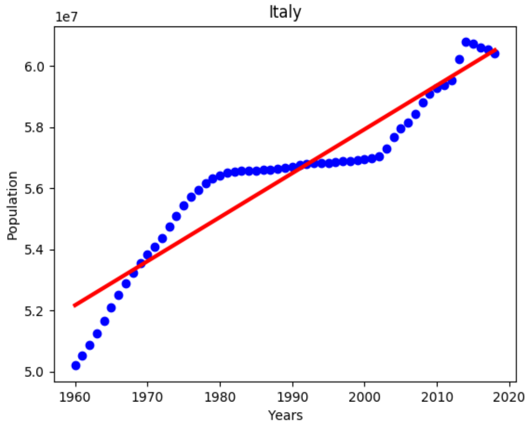

France
GDP
40000
30000
20000
10000
0
1960
1970
1980
1990
2000
2010
2020
Years

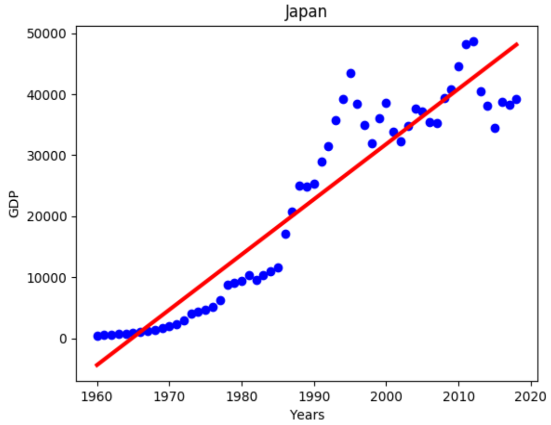

France
1e7
Population
6.5
6.0
5.5
5.0
1960
1970
1980
1990
2000
2010
2020
Years

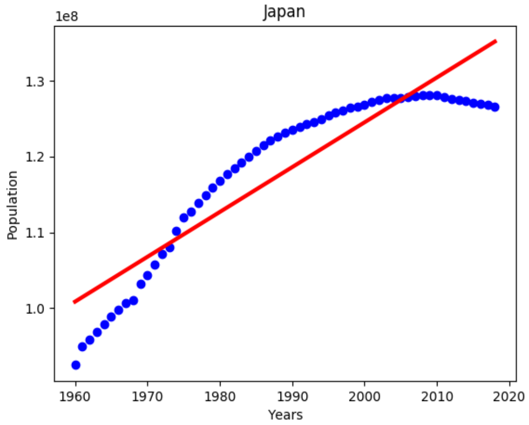

Germany
GDP
50000
40000
30000
20000
10000
0
1960
1970
1980
1990
2000
2010
2020
Years

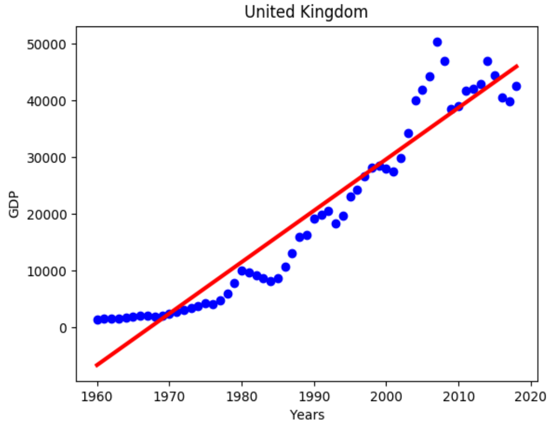

Germany
1e7
Population
8.2
8.0
7.8
7.6
7.4
1960
1970
1980
1990
2000
2010
2020
Years

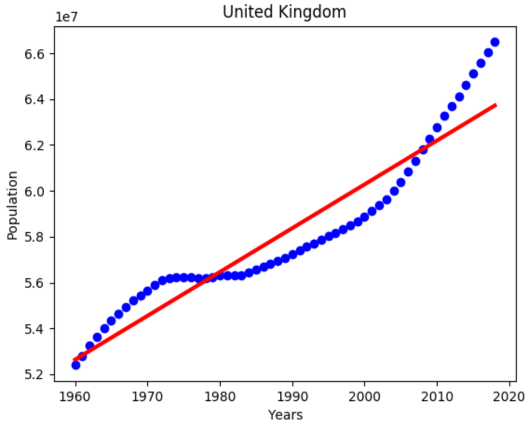

Italy
GDP
40000
30000
20000
10000
0
1960
1970
1980
1990
2000
2010
2020
Years

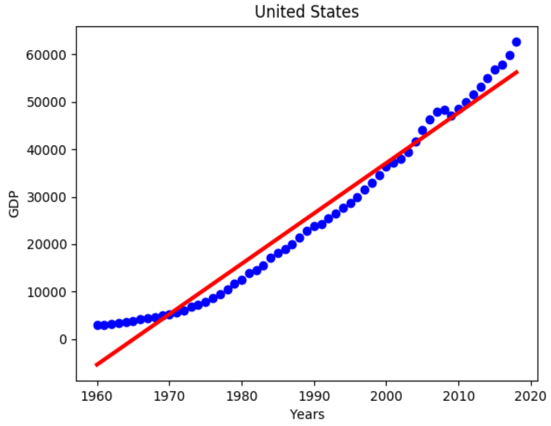

Italy
1e7
Population
6.0
5.8
5.6
5.4
5.2
5.0
1960
1970
1980
1990
2000
2010
2020
Years

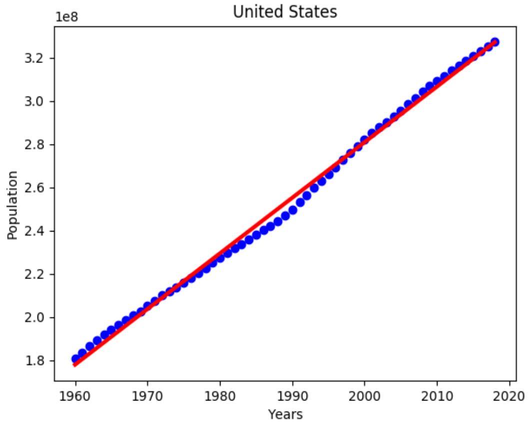

Japan
GDP
50000
40000
30000
20000
10000
0
1960
1970
1980
1990
2000
2010
2020
Years


Japan
1e8
Population
1.3
1.2
1.1
1.0
1960
1970
1980
1990
2000
2010
2020
Years


United Kingdom
GDP
50000
40000
30000
20000
10000
0
1960
1970
1980
1990
2000
2010
2020
Years


United Kingdom
1e7
Population
6.6
6.4
6.2
6.0
5.8
5.6
5.4
5.2
1960
1970
1980
1990
2000
2010
2020
Years


United States
GDP
60000
50000
40000
30000
20000
10000
0
1960
1970
1980
1990
2000
2010
2020
Years


United States
1e8
Population
3.2
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1960
1970
1980
1990
2000
2010
2020
Years

3.2 Current Analysis and Predictions for 2060 for each Country in G7:

Canada:

- Current GDP per capita: 46210.54762
- Predicted GDP per capita: 83911.48634488
- Current population: 37058856
- Predicted population: 49941029

France:

- Current GDP per capita: 41463.64402
- Predicted GDP per capita: 77549.00154831
- Current population: 66987244
- Predicted population: 81397535

Germany:

- Current GDP per capita: 48195.5799
- Predicted GDP per capita: 85686.94616482
- Current population: 82927922
- Predicted population: 88891603

Italy:

- Current GDP per capita: 34318.35112
- Predicted GDP per capita: 68404.58506032
- Current population: 60431283
- Predicted population: 66574480

Japan:

- Current GDP per capita: 39286.73765
- Predicted GDP per capita: 86081.63207218
- Current population: 126529100
- Predicted population: 1.59996004e+08

United Kingdom:

- Current GDP per capita: 42491.36444
- Predicted GDP per capita: 84113.41592733
- Current population: 66488991
- Predicted population: 71747047

United States:

- Current GDP per capita: 62641.01457
- Predicted GDP per capita: 100888.4400335
- Current population: 327167434
- Predicted population: 4.35082527e+08

4. Technical Challenges

While doing the project, we faced many technical challenges such as empty data points, commas in the country names, outlier data points, bulk-loading format errors, finding the proper library to bulk-load the csv files to the database etc.

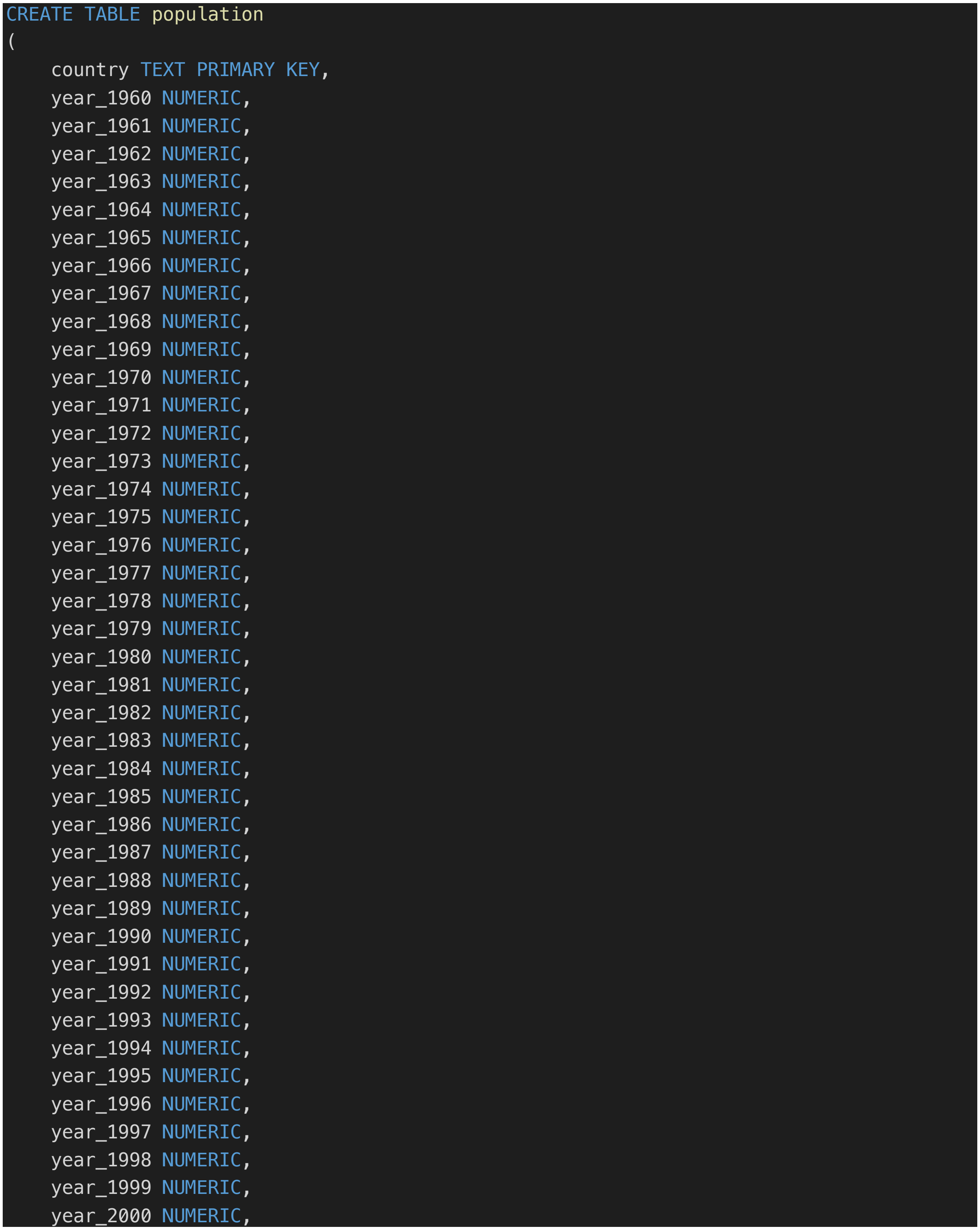
5. Concluding Remark

In general, it is a time-consuming task to prepare a dataset as an input to a machine learning algorithm. We have benefitted from both Python and PostgreSQL features to make this process really easy and we made logical guests for the statistical values that indicates the future of G7 countries.

Machine learning algorithms shed light on the future as the day goes on. As humankind, if we manage to develop these algorithms to an optimal level and use them in a good way, we will reach to the top of civilization.

6. Addendum

6.1 table.psql



```
CREATE TABLE population
(
    country TEXT PRIMARY KEY,
    year_1960 NUMERIC,
    year_1961 NUMERIC,
    year_1962 NUMERIC,
    year_1963 NUMERIC,
    year_1964 NUMERIC,
    year_1965 NUMERIC,
    year_1966 NUMERIC,
    year_1967 NUMERIC,
    year_1968 NUMERIC,
    year_1969 NUMERIC,
    year_1970 NUMERIC,
    year_1971 NUMERIC,
    year_1972 NUMERIC,
    year_1973 NUMERIC,
    year_1974 NUMERIC,
    year_1975 NUMERIC,
    year_1976 NUMERIC,
    year_1977 NUMERIC,
    year_1978 NUMERIC,
    year_1979 NUMERIC,
    year_1980 NUMERIC,
    year_1981 NUMERIC,
    year_1982 NUMERIC,
    year_1983 NUMERIC,
    year_1984 NUMERIC,
    year_1985 NUMERIC,
    year_1986 NUMERIC,
    year_1987 NUMERIC,
    year_1988 NUMERIC,
    year_1989 NUMERIC,
    year_1990 NUMERIC,
    year_1991 NUMERIC,
    year_1992 NUMERIC,
    year_1993 NUMERIC,
    year_1994 NUMERIC,
    year_1995 NUMERIC,
    year_1996 NUMERIC,
    year_1997 NUMERIC,
    year_1998 NUMERIC,
    year_1999 NUMERIC,
    year_2000 NUMERIC,
    year_2001 NUMERIC,
    year_2002 NUMERIC,
    year_2003 NUMERIC,
    year_2004 NUMERIC,
    year_2005 NUMERIC,
    year_2006 NUMERIC,
    year_2007 NUMERIC,
    year_2008 NUMERIC,
    year_2009 NUMERIC,
    year_2010 NUMERIC,
    year_2011 NUMERIC,
    year_2012 NUMERIC,
    year_2013 NUMERIC,
    year_2014 NUMERIC,
    year_2015 NUMERIC,
    year_2016 NUMERIC,
    year_2017 NUMERIC,
    year_2018 NUMERIC
);

CREATE TABLE gdp
(
    country TEXT PRIMARY KEY,
    year_1960 NUMERIC,
    year_1961 NUMERIC,
    year_1962 NUMERIC,
    year_1963 NUMERIC,
    year_1964 NUMERIC,
    year_1965 NUMERIC,
    year_1966 NUMERIC,
    year_1967 NUMERIC,
    year_1968 NUMERIC,
    year_1969 NUMERIC,
    year_1970 NUMERIC,
    year_1971 NUMERIC,
    year_1972 NUMERIC,
    year_1973 NUMERIC,
    year_1974 NUMERIC,
    year_1975 NUMERIC,
    year_1976 NUMERIC,
    year_1977 NUMERIC,
    year_1978 NUMERIC,
    year_1979 NUMERIC,
    year_1980 NUMERIC,
    year_1981 NUMERIC,
    year_1982 NUMERIC,
    year_1983 NUMERIC,
    year_1984 NUMERIC,
    year_1985 NUMERIC,
    year_1986 NUMERIC,
    year_1987 NUMERIC,
    year_1988 NUMERIC,
    year_1989 NUMERIC,
    year_1990 NUMERIC,
    year_1991 NUMERIC,
    year_1992 NUMERIC,
    year_1993 NUMERIC,
    year_1994 NUMERIC,
    year_1995 NUMERIC,
    year_1996 NUMERIC,
    year_1997 NUMERIC,
    year_1998 NUMERIC,
    year_1999 NUMERIC,
    year_2000 NUMERIC,
    year_2001 NUMERIC,
    year_2002 NUMERIC,
    year_2003 NUMERIC,
    year_2004 NUMERIC,
    year_2005 NUMERIC,
    year_2006 NUMERIC,
    year_2007 NUMERIC,
    year_2008 NUMERIC,
    year_2009 NUMERIC,
    year_2010 NUMERIC,
    year_2011 NUMERIC,
    year_2012 NUMERIC,
    year_2013 NUMERIC,
    year_2014 NUMERIC,
    year_2015 NUMERIC,
    year_2016 NUMERIC,
    year_2017 NUMERIC,
    year_2018 NUMERIC
);

CREATE TABLE g7_population AS
  SELECT * FROM population WHERE country = 'France' or country = 'United States' or
  country = 'Japan' or country = 'United Kingdom' or country = 'Italy' or
  country = 'Canada' or country = 'Germany';

  CREATE TABLE g7_gdp AS
  SELECT * FROM gdp WHERE country = 'France' or country = 'United States' or
  country = 'Japan' or country = 'United Kingdom' or country = 'Italy' or
  country = 'Canada' or country = 'Germany';
```

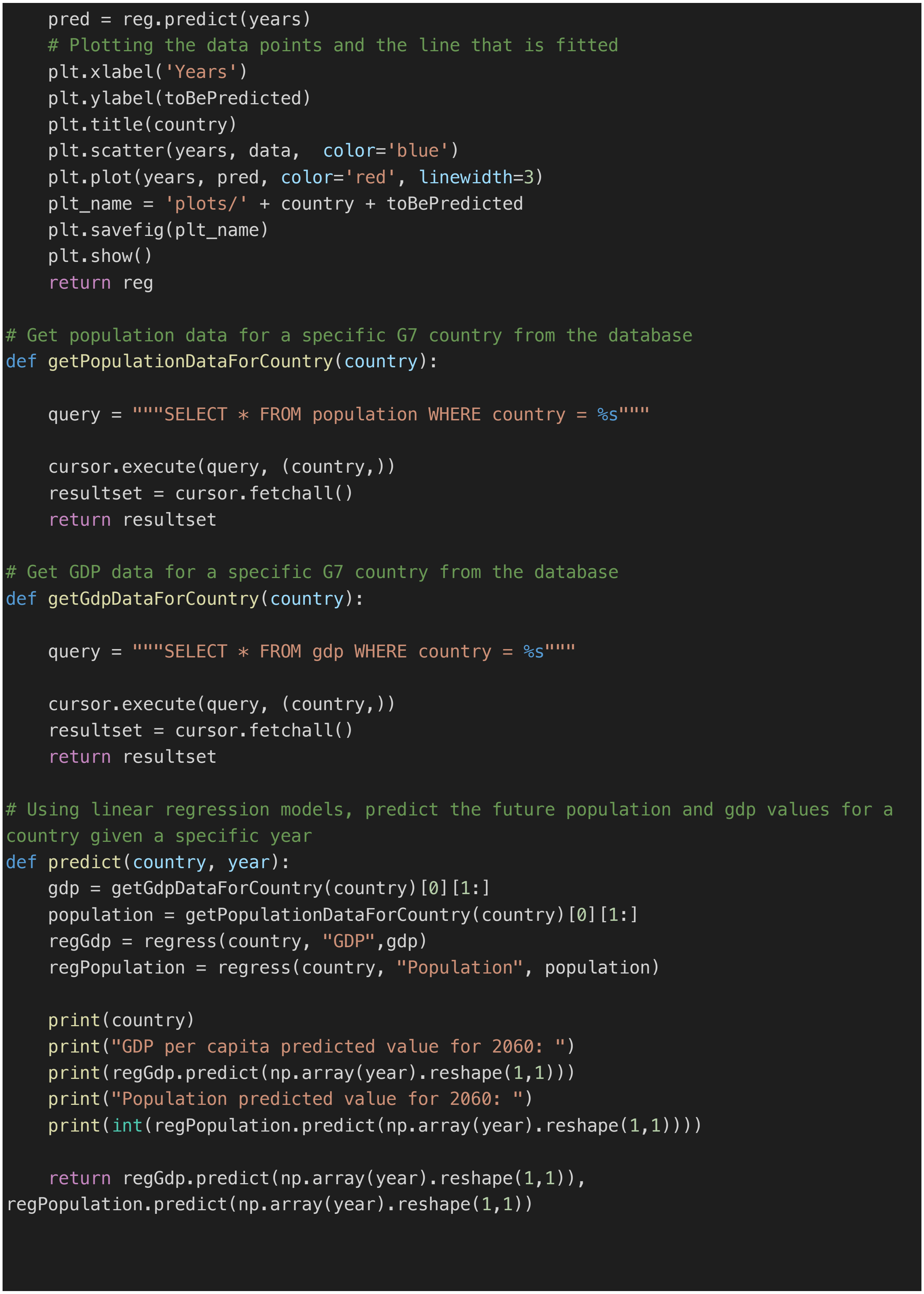
```sql
DROP TABLE gdp;
DROP TABLE population;
DROP TABLE g7_gdp;
DROP TABLE g7_population;
```

6.2 bulkLoad.py

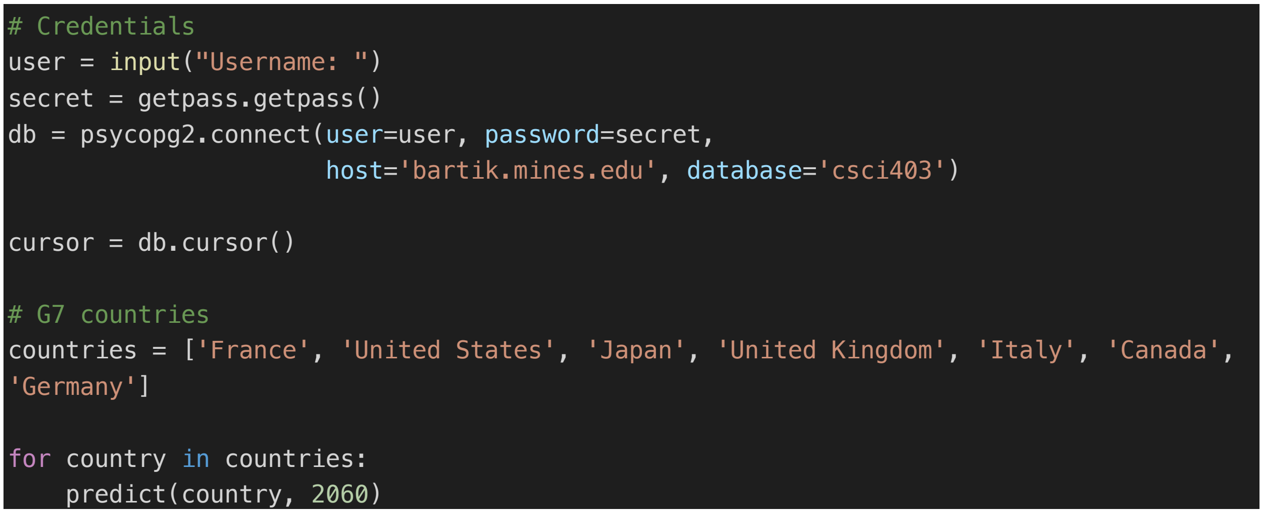
```python
import psycopg2
import getpass

# Bulkloading population and gdp csv files into the database.

# Credentials
user = input("Username: ")
secret = getpass.getpass()
db = psycopg2.connect(user=user, password=secret,
                      host='bartik.mines.edu', database='csci403')

cursor = db.cursor()

# Copying the data from the CSV file to the database
with open('population.csv', 'r') as f:
    next(f)  # Skip the header row.
    cursor.copy_from(f, 'population', sep=',', null="")
    db.commit()

# Same for gdp
with open('gdp.csv', 'r') as f:
    next(f)
    cursor.copy_from(f, 'gdp', sep=',', null="")
    db.commit()
```

6.3 linearRegression.py

```python
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import numpy as np
import psycopg2
import getpass

# Linear regression using sklearn library
def regress(country, toBePredicted, data):
    years = np.arange(1960, 2019)
    years = years.reshape(-1, 1)

    reg = LinearRegression().fit(years, data)
```



```
    pred = reg.predict(years)
    # Plotting the data points and the line that is fitted
    plt.xlabel('Years')
    plt.ylabel(toBePredicted)
    plt.title(country)
    plt.scatter(years, data,  color='blue')
    plt.plot(years, pred, color='red', linewidth=3)
    plt_name = 'plots/' + country + toBePredicted
    plt.savefig(plt_name)
    plt.show()
    return reg

# Get population data for a specific G7 country from the database
def getPopulationDataForCountry(country):

    query = """SELECT * FROM population WHERE country = %s"""

    cursor.execute(query, (country,))
    resultset = cursor.fetchall()
    return resultset

# Get GDP data for a specific G7 country from the database
def getGdpDataForCountry(country):

    query = """SELECT * FROM gdp WHERE country = %s"""

    cursor.execute(query, (country,))
    resultset = cursor.fetchall()
    return resultset

# Using linear regression models, predict the future population and gdp values for a
country given a specific year
def predict(country, year):
    gdp = getGdpDataForCountry(country)[0][1:]
    population = getPopulationDataForCountry(country)[0][1:]
    regGdp = regress(country, "GDP",gdp)
    regPopulation = regress(country, "Population", population)

    print(country)
    print("GDP per capita predicted value for 2060: ")
    print(regGdp.predict(np.array(year).reshape(1,1)))
    print("Population predicted value for 2060: ")
    print(int(regPopulation.predict(np.array(year).reshape(1,1))))

    return regGdp.predict(np.array(year).reshape(1,1)),
regPopulation.predict(np.array(year).reshape(1,1))
```



```
# Credentials
user = input("Username: ")
secret = getpass.getpass()
db = psycopg2.connect(user=user, password=secret,
                      host='bartik.mines.edu', database='csci403')

cursor = db.cursor()

# G7 countries
countries = ['France', 'United States', 'Japan', 'United Kingdom', 'Italy', 'Canada',
'Germany']

for country in countries:
    predict(country, 2060)
```