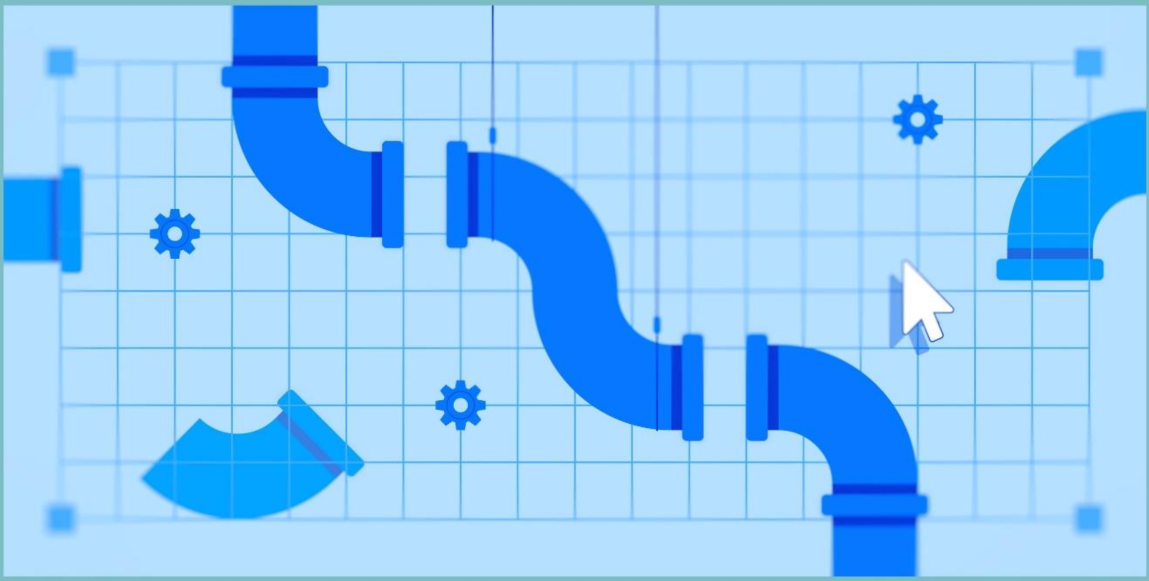
Data Pipeline

FOR

SOCIO-ECONOMIC DATA Relating to SDG 1: No poverty



Melvin M. Managbanag
SPARTA PH DE Pathway scholar

Data Engineering Capstone Project

A. Introduction

According to Kraak et al. (2018), The global population is growing at an incomprehensible rate and with it come complex environmental consequences that often result in social injustice. The United Nations has established a set of Sustainable Development Goals (SDGs) in an attempt to ameliorate inequality and promise safety for the masses. To reach these goals, a set of indicators have been identified and their associated data for each country are publicly available to measure how close each country is to each goal.

B. Statement of the Problem/ Objectives

B.1 **Problem Statement:**

The Philippines faces socio-economic challenges, particularly related to poverty, which is a critical concern in the context of the United Nations Sustainable Development Goal (SDG) 1 - "No Poverty." To address this issue effectively, there is a need to collect, store, and present socioeconomic data. Current data management approaches may lack the necessary structure and data-driven capabilities to support this goal.

B.2 **Objective:**

The primary objective of this project is to design and implement a comprehensive database schema tailored to the specific socioeconomic needs of the Philippine’s Region. This schema includes dimension (Dim) and fact (Fact) tables that closely align with the objectives of SDG 1, focusing on the eradication of poverty in all its forms. The proposed database pipeline aims to:

Collect and store data by creating a structured system capable of efficiently collecting, storing, and managing socio-economic data from various sources, with each dataset corresponding to a specific year and region.

Enables the systematic monitoring of poverty reduction progress, a fundamental aspect of SDG 1, by providing a framework for organizing and presenting data related to critical factors, including poverty rates, income, employment, and population.

In summary, the project's objective is to develop a robust database schema that supports the goals of Socio-Economic factor relating to SDG 1, by providing a structured and data-driven approach to address and monitor issues related to poverty within the Philippines.

C. **Solution**

Establishing a simple yet effective data pipeline for collecting and storing socioeconomic data is crucial, as illustrated in Figure 1 on page 3. We will divide the process into two phases. Phase I shown in Image 1 below, involves the extraction of data from the source webpage/site, followed by pre-processing, which will be done in Excel. Another transformation process to conforms to the structure of the Normalized Model will be carried out using Python, and finally, the transformed data will be loaded into the database using Python and SQL.

. For Phase II, as illustrated in Image 2 on page 3, the process will entail extracting data from the database, followed by transforming the schema based on the dimension and fact tables, then the data will be loaded into the data warehouse, and lastly the data warehouse will be connected to Excel via ODBC direct query connection for data visualization.

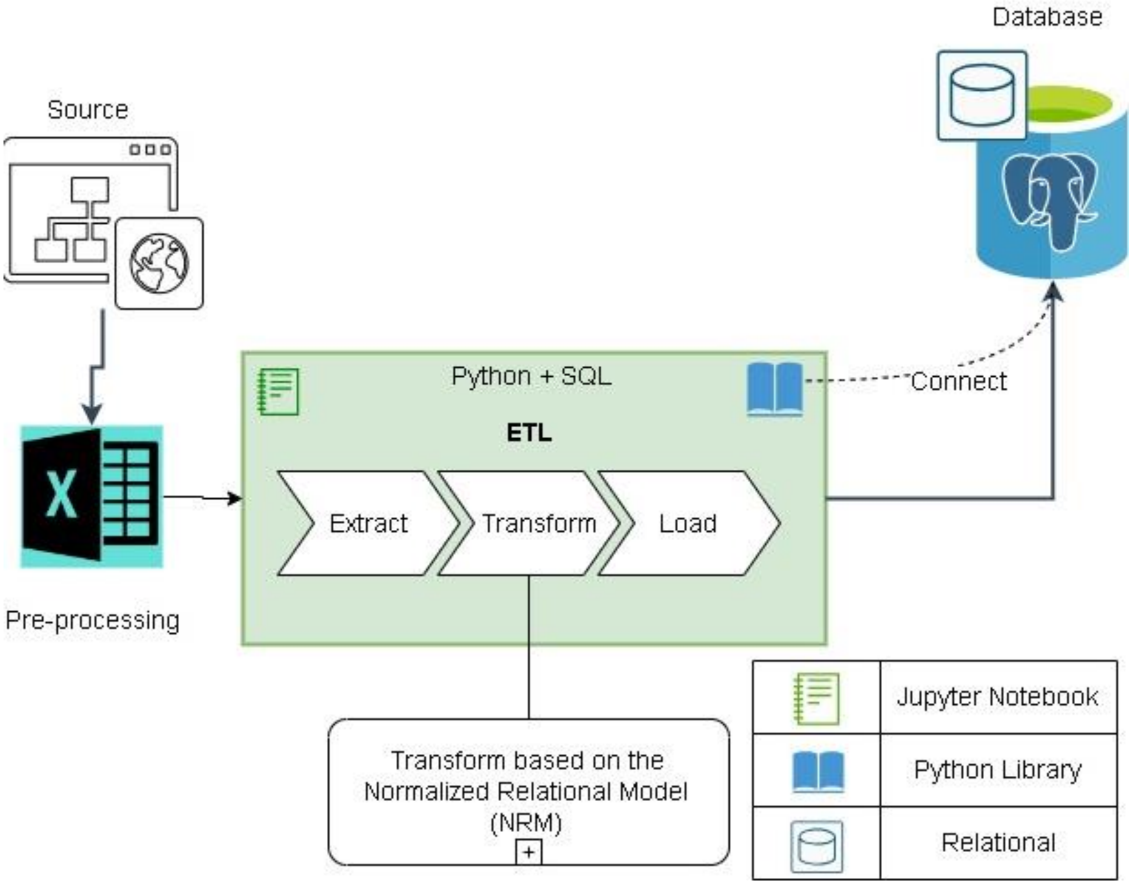


Image 1. PHASE I (Data pipeline)

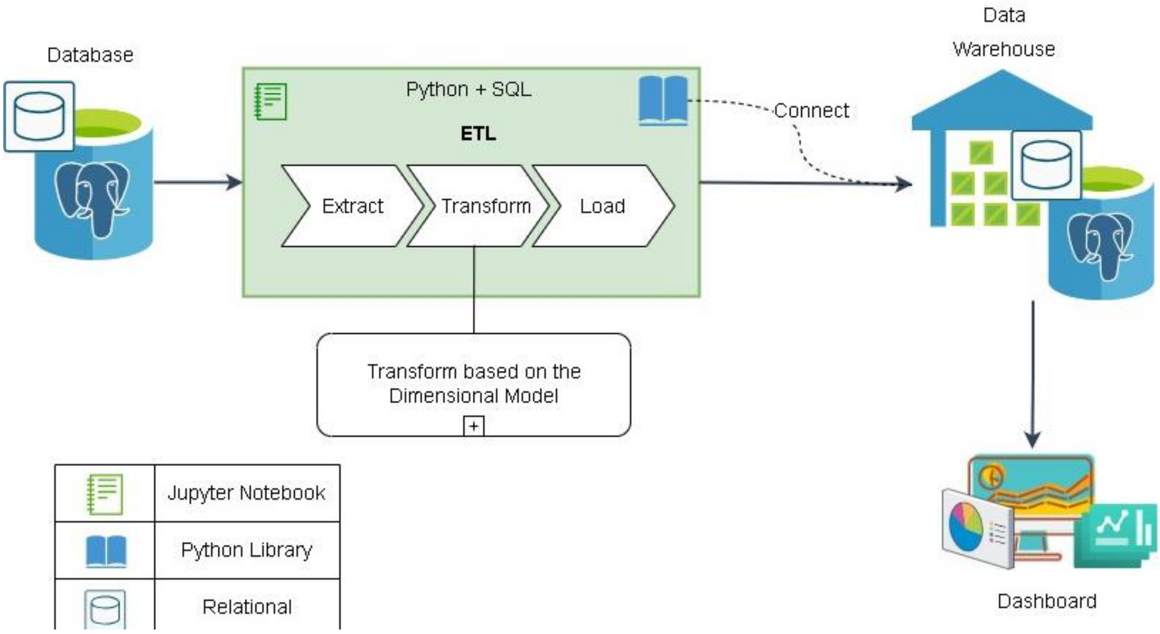


Image 2. PHASE II (Data pipeline)

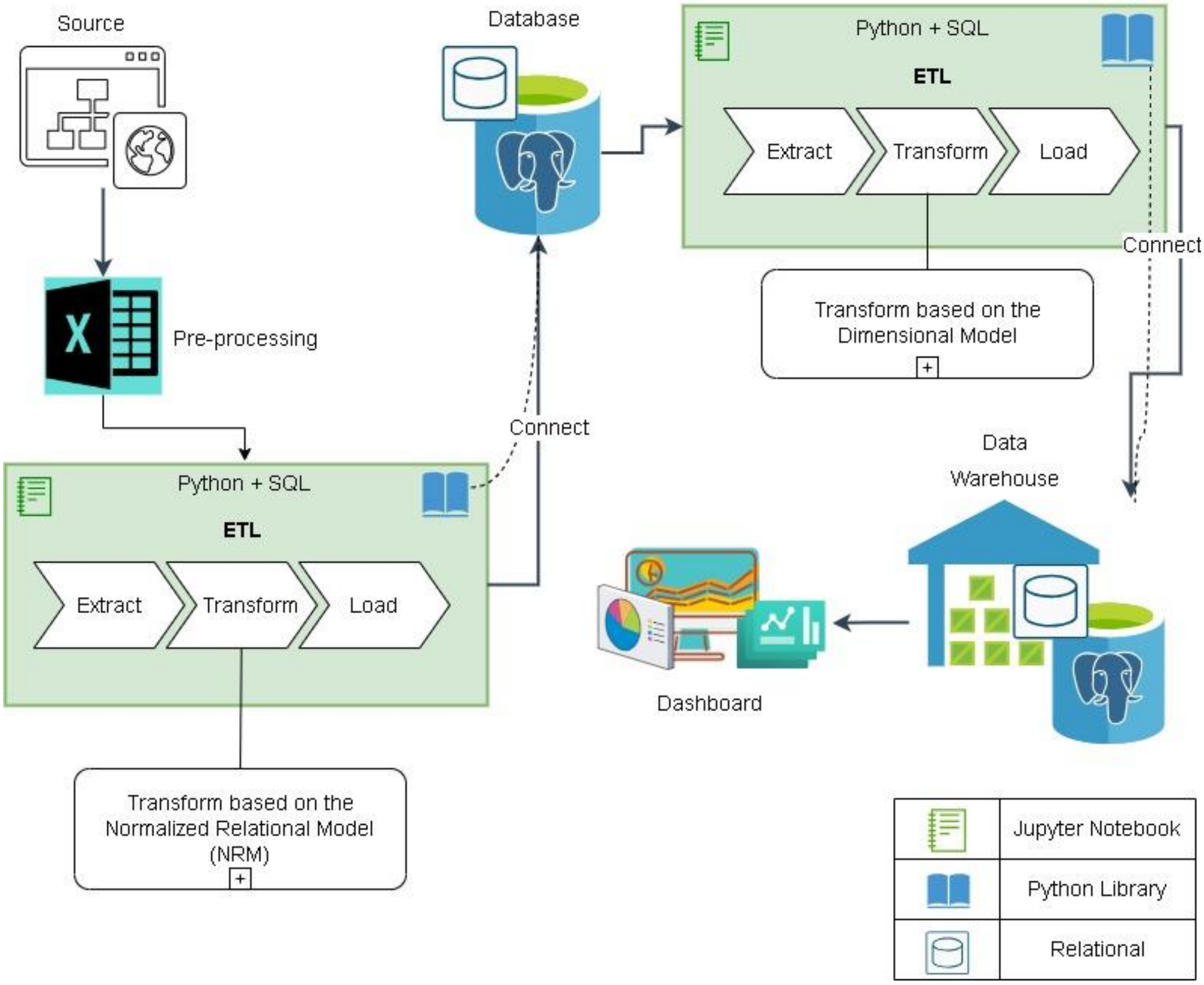


Figure 1. Data pipeline (Phase I & II)

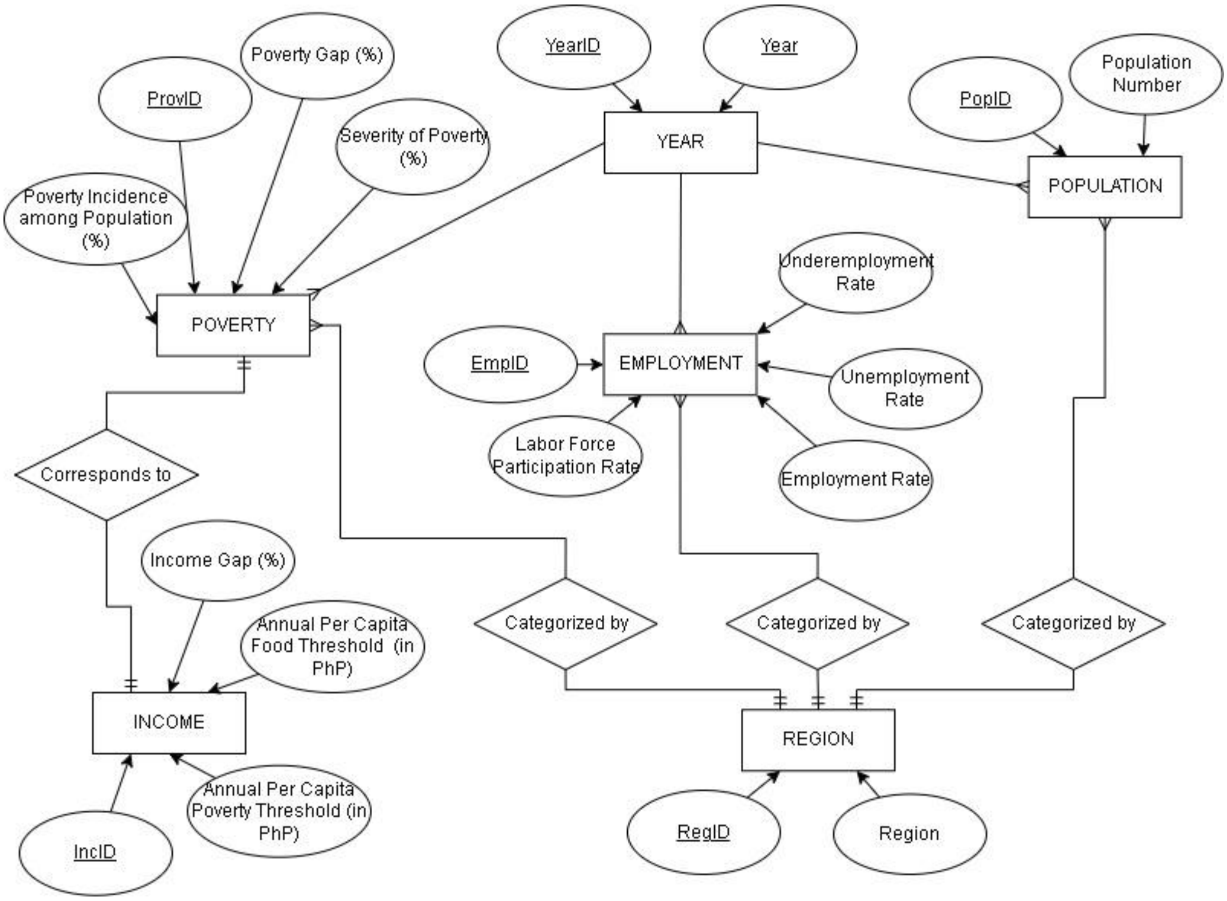


Figure 2. Entity-Relational Diagram

Definition of Attributes

Population Number

The number of persons in a specific region including all individuals, regardless of their age.

Annual Per Capita Food Threshold

Refers to a specific level or amount of expenditure or consumption of food that is considered the minimum necessary to meet basic nutritional needs for an individual or a household over the course of a year. "Annual Per Capita" means that this threshold is calculated on a per-person basis and is applicable for a year. When a person or household's food consumption or expenditure falls below this threshold, it indicates a risk of food insecurity or poverty.

Annual Per Capita Poverty Threshold (in PhP)

This indicator represents the income or consumption level (measured in Philippine Pesos, PhP) below which a person or household is considered to be living in poverty. It takes into account not only the cost of food but also other basic necessities such as housing, clothing, education, healthcare, and transportation. Similar to the food threshold, it is calculated on a per-person basis, considering the size of the household.

Poverty Incidence among Population (%)

A statistical measure that represents the percentage of a population living in poverty within a specific region. This indicator is a fundamental measure for assessing the extent and prevalence of poverty in a given geographic area.

Income Gap

The disparity or difference in income levels among individuals or households within a population. It is often measured using metrics such as the Gini coefficient or other income distribution measures. A higher income gap indicates greater income inequality, while a lower gap suggests more equal distribution of income.

Poverty Gap

This will measure the depth or severity of poverty within a population. It represents the average income shortfall of people living in poverty compared to the poverty line or threshold. In other words, it quantifies how far below the poverty line the average poor person's income falls. A higher poverty gap indicates a larger income shortfall among those in poverty.

Severity of Poverty

Assesses the intensity or severity of deprivation experienced by people living in poverty. It considers not only the incidence (percentage of people in poverty) but also the depth of poverty. It provides insight into how poor individuals or households are relative to the poverty line. A higher severity of poverty indicates that people living in poverty are experiencing more significant deprivation.

Labor Force Participation Rate

Measures the proportion of the working-age population (typically individuals aged 15-64). by calculating the percentage of people who are either employed or actively looking for work relative to the total population within the specified age range. The labor force participation rate also provides insights into the willingness and ability of the working-age population to engage in the labor market.

Employment Rate (Employment-to-Population Ratio)

Measures the proportion of the working-age population that is employed/has a job. It calculates the percentage of people who are currently employed relative to the total working-age population. The employment rate is a measure of the extent to which the working-age population is participating in gainful employment.

Unemployment Rate

Calculates the percentage of unemployed individuals relative to the total labor work force. The unemployment rate provides insight into the level of joblessness within the labor force, it is also a key indicator of economic health and labor market conditions.

Underemployment Rate (or Inadequate Employment Rate)

The underemployment rate measures the proportion of employed individuals who are working in jobs that do not fully utilize their skills and qualifications or who are working part-time involuntarily (i.e., they want full-time work but can't find it). It calculates the percentage of underemployed individuals relative to the total employed population. The underemployment rate highlights the extent to which individuals are not fully employed or are in jobs that do not match their qualifications.

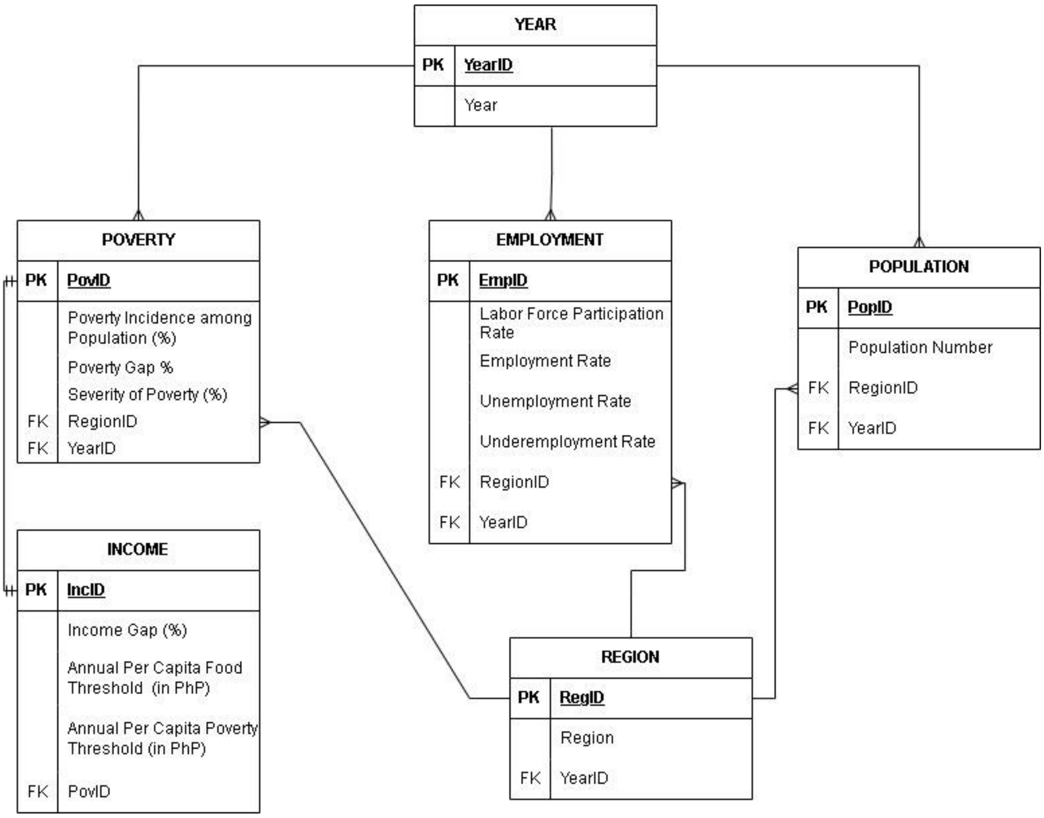


Figure 3. Normalized Model

Based on the ERD (Entity-Relational Diagram) in Figure 2 on page 4, we can now derive the NM (Normalized Model) for the structure of the database, named "povdb". as show in Figure 3 on page 6.

Relationship:

One to One (1 : 1)

- Income to Poverty

One to Many (1 : N)

- Year to Poverty
- Year to Employment
- Year to Population
- Region to Poverty
- Region to Employment
- Region to Population

Example Table: (just to visualized the table format, not the actual data presented)

Year	Region	Poverty score	Employment rate	Population No. (M)
2015	Region I	12	92	5.1
2015	Region II	9	95	3.6
2015	Region III	10	95	12.4
2015	Region IV	13	93	16.1
2015	Region V	12	96	7.9
2015	Region VI	13	93	16.1
2019	Region VII	13	93	16.1
2019	Region I	13	93	16.1
2019	Region II	11	92	6
2019	Region III	12	96	7.9
2019	Region IV	9	95	3.6
2019	Region V	10	95	12.4
2019	Region VI	13	93	16.1
2019	Region VII	13	93	16.1
2021	Region I	13	93	16.1
2021	Region II	12	96	7.9
2021	Region IV	12	96	7.9
2021	Region V	9	95	3.6
2021	Region VI	10	95	12.4
2021	Region VII	13	93	16.1

The table above is an example of the merge data of year, region, poverty, employment, and population table. The Normalized Model Figure 3 in page 6, is designed in-order to eliminate redundancy and to achieved normalization.

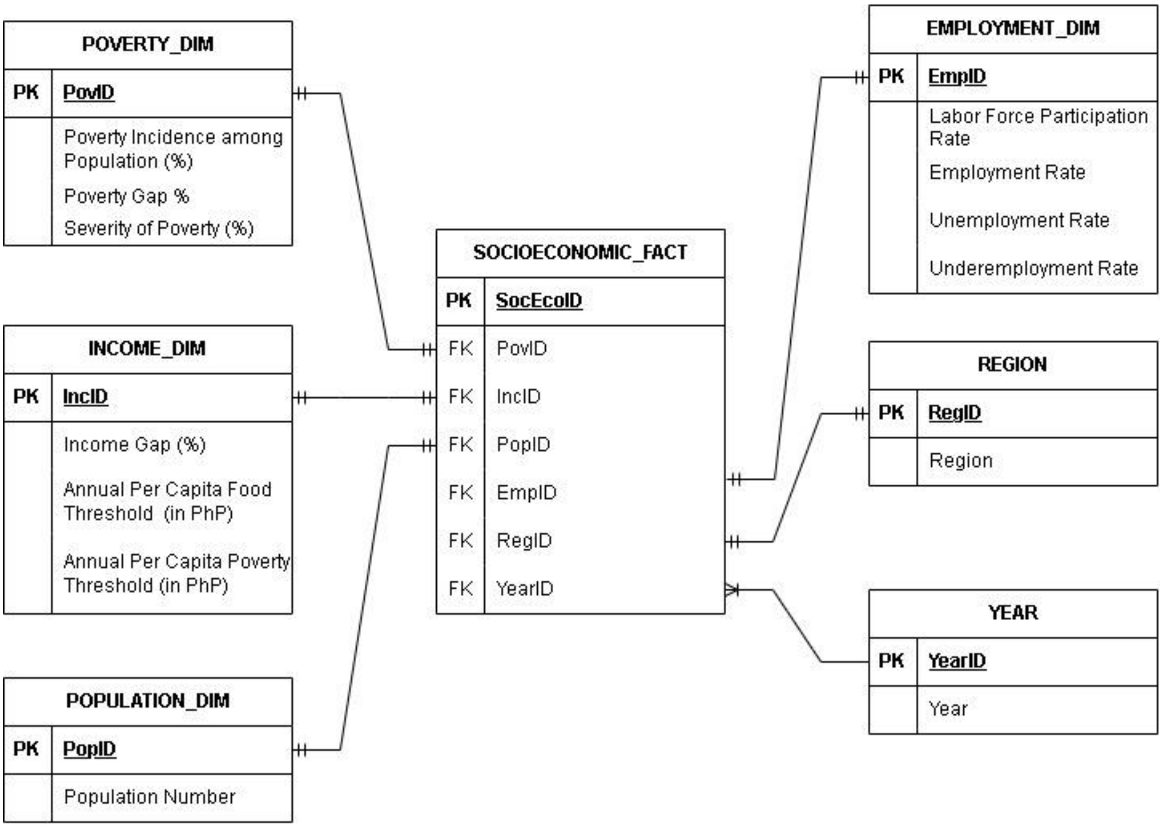


Figure 4. Dimension and Fact Table (Star schema)

The dimension and Fact table was derived from the normalized model. It will be use for the structure of the data warehouse, named "povdw," as depicted in Figure 4 above. In this schema, the fact table is referred to as "fact-less" because its primary purpose is to establish relationships between elements from different dimensions and it does not contain aggregated or calculated data.

ABOUT DATA

Dataset	Year	Frequency of Update (PSA)
Population	2010, 2015, 2020	Every 5 years
Poverty	2015, 2018, 2021p	Every after 2 years
Income	2015, 2018, 2021p	Every after 2 years
Employment	2015, 2018, 2021p	Every after 2 years

The frequency of update for the population dataset does not align with the poverty, income and employment data except for year 2015, therefore we can employ a linear interpolation formula on the population data.

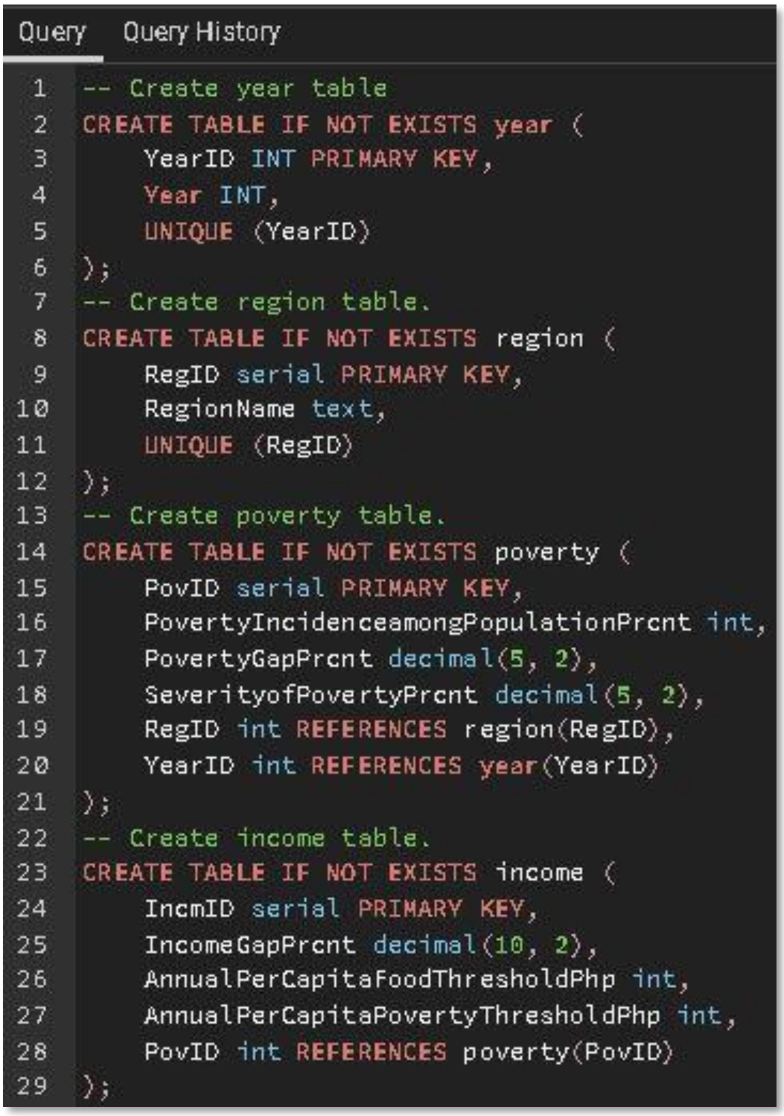
Linear Interpolation:

$$y' = y1 + (x - x1) * ((y2 - y1) / (x2 - x1))$$

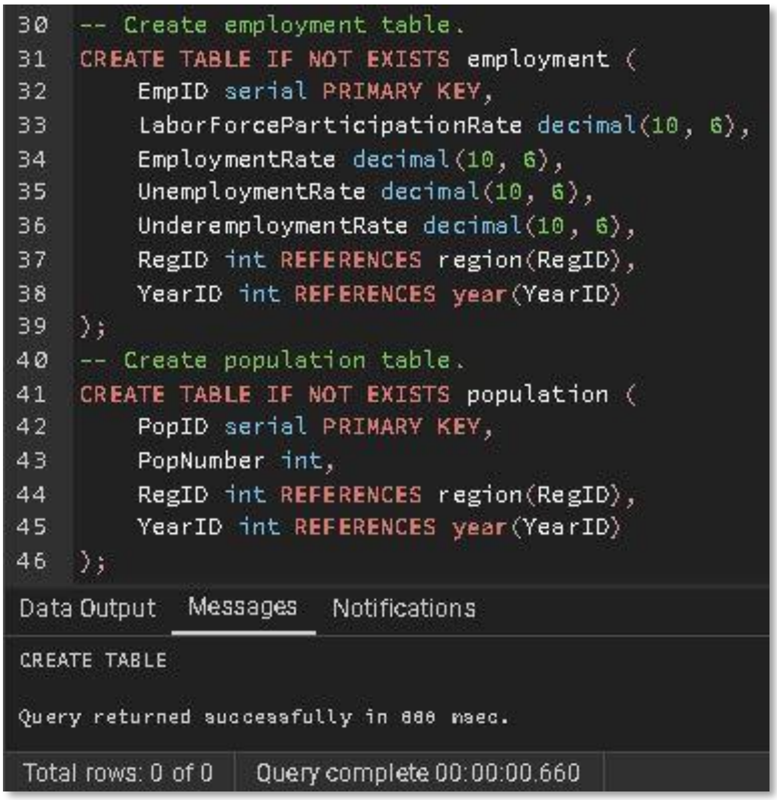
The formula above will be utilized during transformation staged of population data to get the year 2018 and 2021.

After presenting the data pipeline and establishing the ERD, NM (Normalized Model), and Dimension & Fact Tables, will now proceed to create the database in PostgreSQL based on the Normalized Model structure and create the data warehouse based on the Dimension & Fact table (Star-schema) structure.

SQL script for povdb based on NM (database)



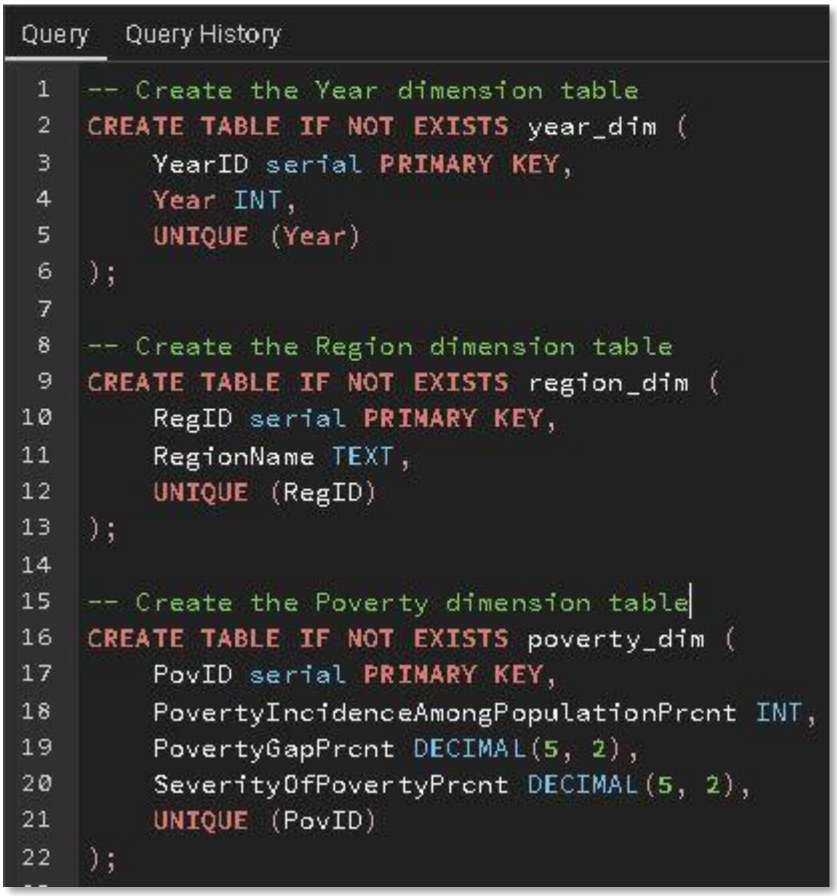
```sql
-- Create year table
CREATE TABLE IF NOT EXISTS year (
    YearID INT PRIMARY KEY,
    Year INT,
    UNIQUE (YearID)
);
-- Create region table.
CREATE TABLE IF NOT EXISTS region (
    RegID serial PRIMARY KEY,
    RegionName text,
    UNIQUE (RegID)
);
-- Create poverty table.
CREATE TABLE IF NOT EXISTS poverty (
    PovID serial PRIMARY KEY,
    PovertyIncidenceamongPopulationPrcnt int,
    PovertyGapPrcnt decimal(5, 2),
    SeverityofPovertyPrcnt decimal(5, 2),
    RegID int REFERENCES region(RegID),
    YearID int REFERENCES year(YearID)
);
-- Create income table.
CREATE TABLE IF NOT EXISTS income (
    IncmID serial PRIMARY KEY,
    IncomeGapPrcnt decimal(10, 2),
    AnnualPerCapitaFoodThresholdPhp int,
    AnnualPerCapitaPovertyThresholdPhp int,
    PovID int REFERENCES poverty(PovID)
);
```



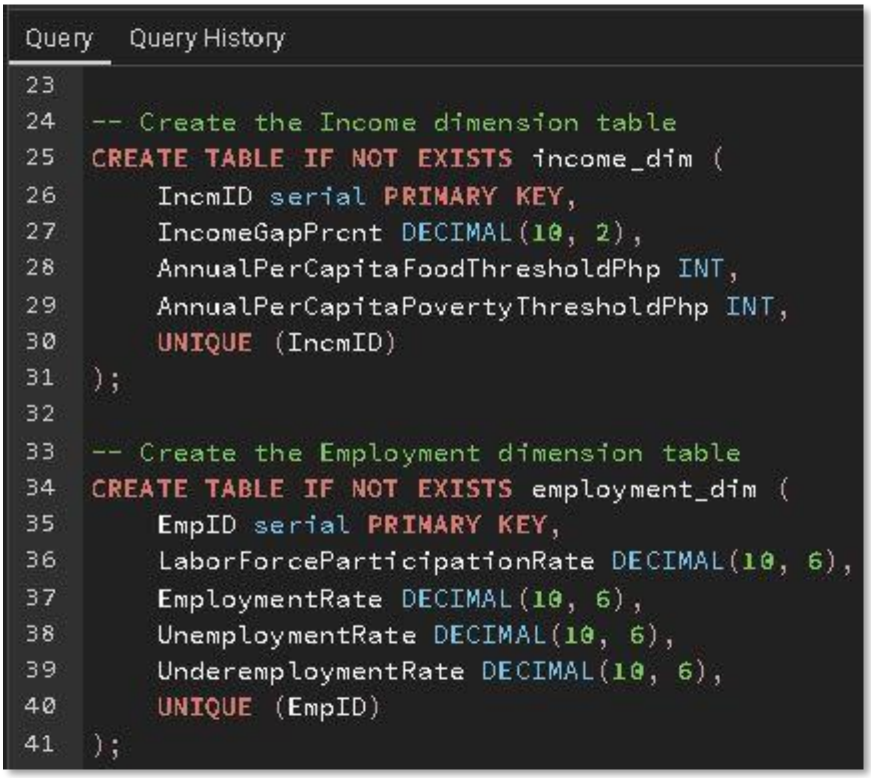
```sql
-- Create employment table.
CREATE TABLE IF NOT EXISTS employment (
    EmpID serial PRIMARY KEY,
    LaborForceParticipationRate decimal(10, 6),
    EmploymentRate decimal(10, 6),
    UnemploymentRate decimal(10, 6),
    UnderemploymentRate decimal(10, 6),
    RegID int REFERENCES region(RegID),
    YearID int REFERENCES year(YearID)
);
-- Create population table.
CREATE TABLE IF NOT EXISTS population (
    PopID serial PRIMARY KEY,
    PopNumber int,
    RegID int REFERENCES region(RegID),
    YearID int REFERENCES year(YearID)
);
```

```
CREATE TABLE

Query returned successfully in 660 msec.

Total rows: 0 of 0    Query complete 00:00:00.660
```

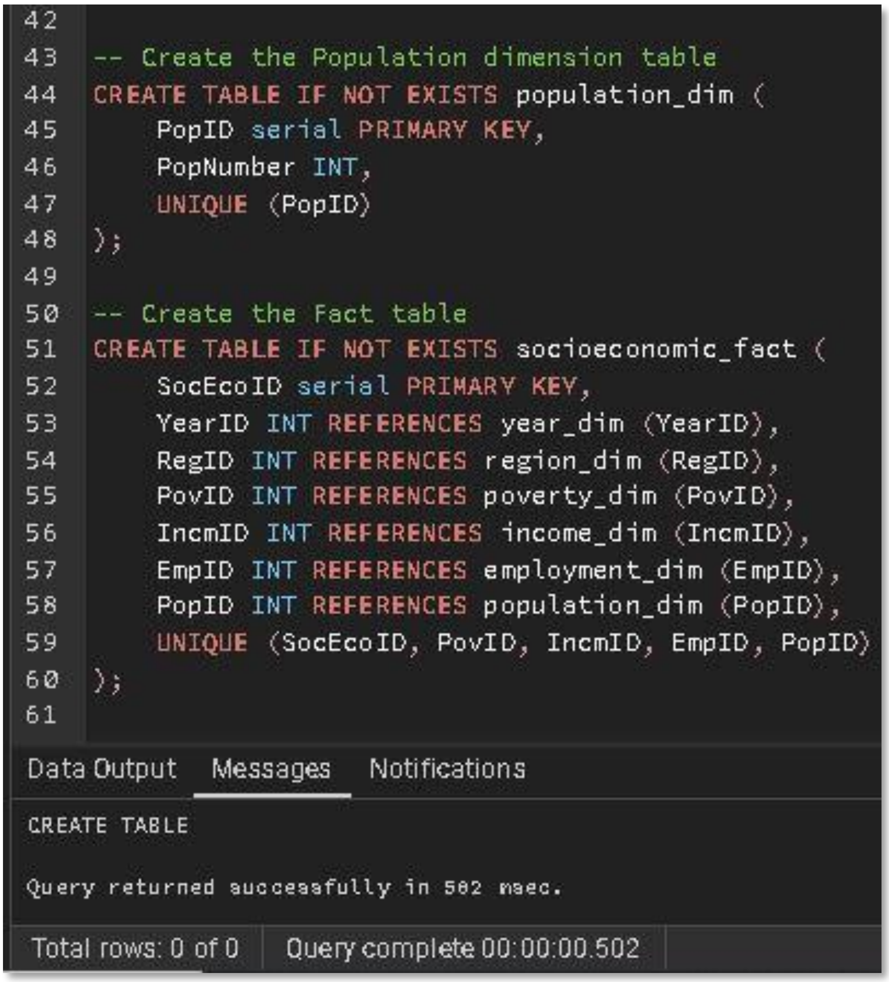
SQL script for povdw base on Dimension and Fact Table (data warehouse)



```sql
-- Create the Year dimension table
CREATE TABLE IF NOT EXISTS year_dim (
    YearID serial PRIMARY KEY,
    Year INT,
    UNIQUE (Year)
);

-- Create the Region dimension table
CREATE TABLE IF NOT EXISTS region_dim (
    RegID serial PRIMARY KEY,
    RegionName TEXT,
    UNIQUE (RegID)
);

-- Create the Poverty dimension table
CREATE TABLE IF NOT EXISTS poverty_dim (
    PovID serial PRIMARY KEY,
    PovertyIncidenceAmongPopulationPrcnt INT,
    PovertyGapPrcnt DECIMAL(5, 2),
    SeverityOfPovertyPrcnt DECIMAL(5, 2),
    UNIQUE (PovID)
);
```



```sql

-- Create the Income dimension table
CREATE TABLE IF NOT EXISTS income_dim (
    IncmID serial PRIMARY KEY,
    IncomeGapPrcnt DECIMAL(10, 2),
    AnnualPerCapitaFoodThresholdPhp INT,
    AnnualPerCapitaPovertyThresholdPhp INT,
    UNIQUE (IncmID)
);

-- Create the Employment dimension table
CREATE TABLE IF NOT EXISTS employment_dim (
    EmpID serial PRIMARY KEY,
    LaborForceParticipationRate DECIMAL(10, 6),
    EmploymentRate DECIMAL(10, 6),
    UnemploymentRate DECIMAL(10, 6),
    UnderemploymentRate DECIMAL(10, 6),
    UNIQUE (EmpID)
);
```



```sql

-- Create the Population dimension table
CREATE TABLE IF NOT EXISTS population_dim (
    PopID serial PRIMARY KEY,
    PopNumber INT,
    UNIQUE (PopID)
);

-- Create the Fact table
CREATE TABLE IF NOT EXISTS socioeconomic_fact (
    SocEcoID serial PRIMARY KEY,
    YearID INT REFERENCES year_dim (YearID),
    RegID INT REFERENCES region_dim (RegID),
    PovID INT REFERENCES poverty_dim (PovID),
    IncmID INT REFERENCES income_dim (IncmID),
    EmpID INT REFERENCES employment_dim (EmpID),
    PopID INT REFERENCES population_dim (PopID),
    UNIQUE (SocEcoID, PovID, IncmID, EmpID, PopID)
);

```

```
CREATE TABLE

Query returned successfully in 502 msec.
```

Total rows: 0 of 0   Query complete 00:00:00.502

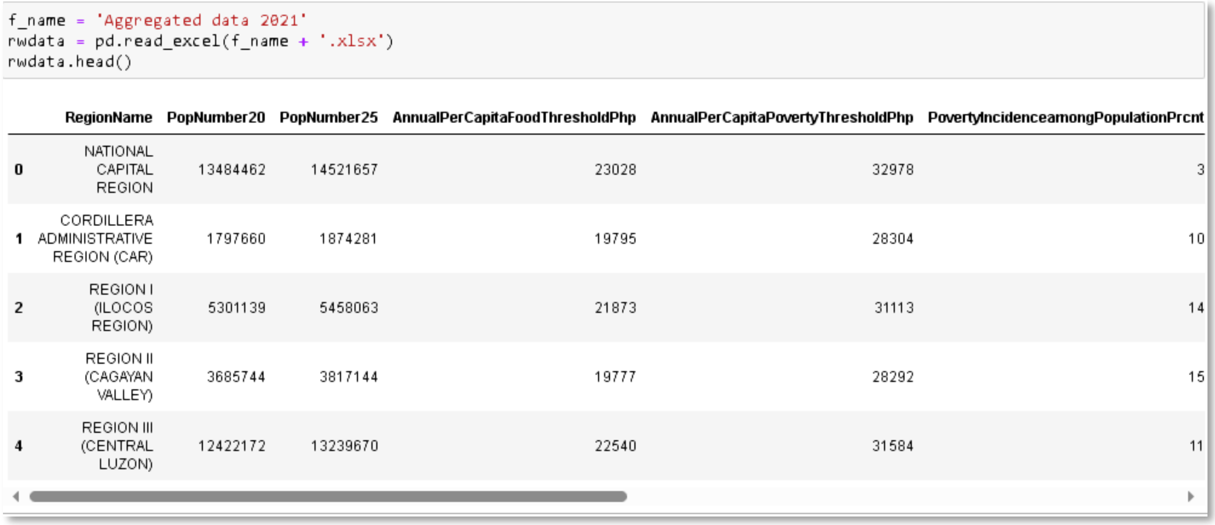
After the creation of the database (povdb) and data warehouse (povdw) in PostgreSQL, the next step is to perform the Phase I of the data pipeline.

PHASE I

Note: The following screen cap shows the 2021 data undergoing Phase I and II processes.

I. 1 EXTRACT

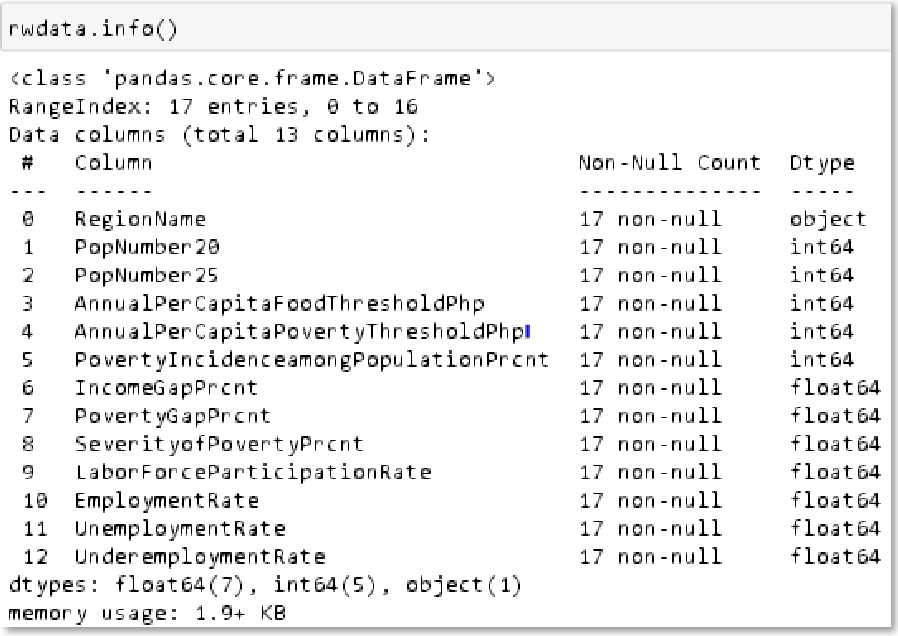
Output of the pre-processing done by aggregating the data using excel.



```
f_name = 'Aggregated data 2021'
rwdata = pd.read_excel(f_name + '.xlsx')
rwdata.head()
```

	RegionName	PopNumber20	PopNumber25	AnnualPerCapitaFoodThresholdPhp	AnnualPerCapitaPovertyThresholdPhp	PovertyIncidenceamongPopulationPrcnt
0	NATIONAL CAPITAL REGION	13484462	14521657	23028	32978	3
1	CORDILLERA ADMINISTRATIVE REGION (CAR)	1797660	1874281	19795	28304	10
2	REGION I (ILOCOS REGION)	5301139	5458063	21873	31113	14
3	REGION II (CAGAYAN VALLEY)	3685744	3817144	19777	28292	15
4	REGION III (CENTRAL LUZON)	12422172	13239670	22540	31584	11

Information of all the columns in the aggregated file.



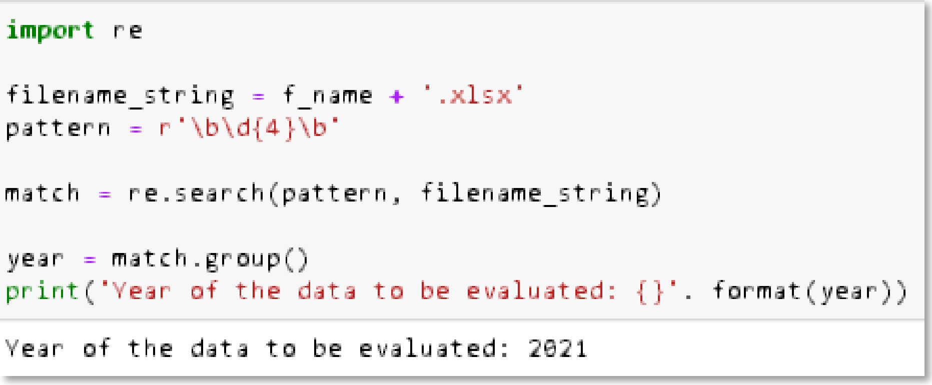
```
rwdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17 entries, 0 to 16
Data columns (total 13 columns):
 #   Column                                Non-Null Count  Dtype  
---  ------                                --------------  -----  
 0   RegionName                            17 non-null     object 
 1   PopNumber20                           17 non-null     int64  
 2   PopNumber25                           17 non-null     int64  
 3   AnnualPerCapitaFoodThresholdPhp       17 non-null     int64  
 4   AnnualPerCapitaPovertyThresholdPhp    17 non-null     int64  
 5   PovertyIncidenceamongPopulationPrcnt  17 non-null     int64  
 6   IncomeGapPrcnt                        17 non-null     float64
 7   PovertyGapPrcnt                       17 non-null     float64
 8   SeverityofPovertyPrcnt                17 non-null     float64
 9   LaborForceParticipationRate           17 non-null     float64
 10  EmploymentRate                        17 non-null     float64
 11  UnemploymentRate                      17 non-null     float64
 12  UnderemploymentRate                   17 non-null     float64
dtypes: float64(7), int64(5), object(1)
memory usage: 1.9+ KB
```

I. 2 EVALUATION

In this section of the ETL process, it will check if the data are already on the database or it is a new data to be added.

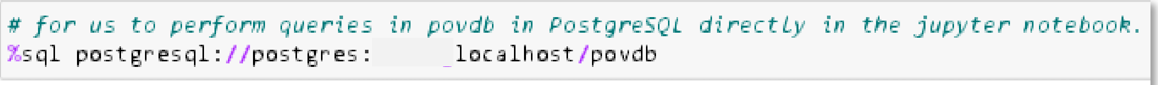
The code below will checked what year is the data to be evaluated.



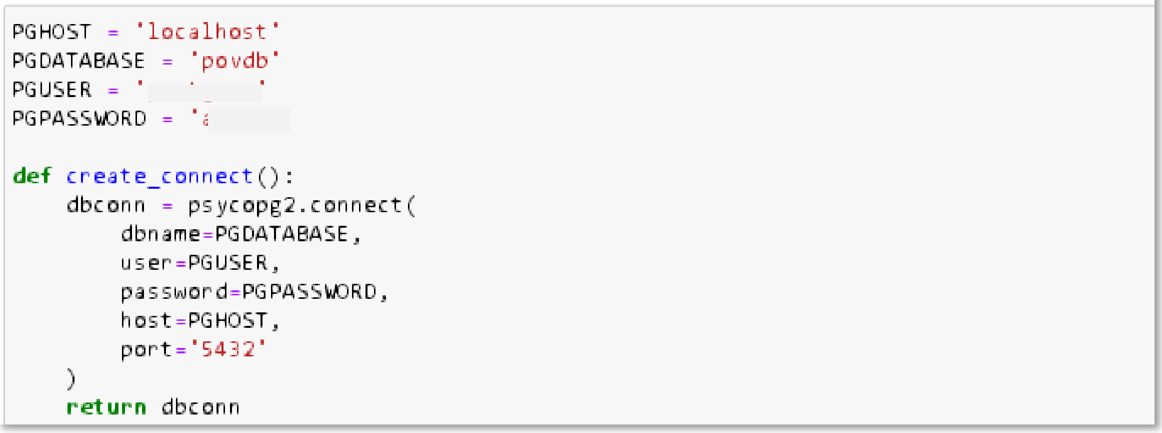
```
import re

filename_string = f_name + '.xlsx'
pattern = r'\b\d{4}\b'

match = re.search(pattern, filename_string)

year = match.group()
print('Year of the data to be evaluated: {}'. format(year))
```

```
Year of the data to be evaluated: 2021
```

Then connect the jupyter notebook to database povdb in PostgreSQL.

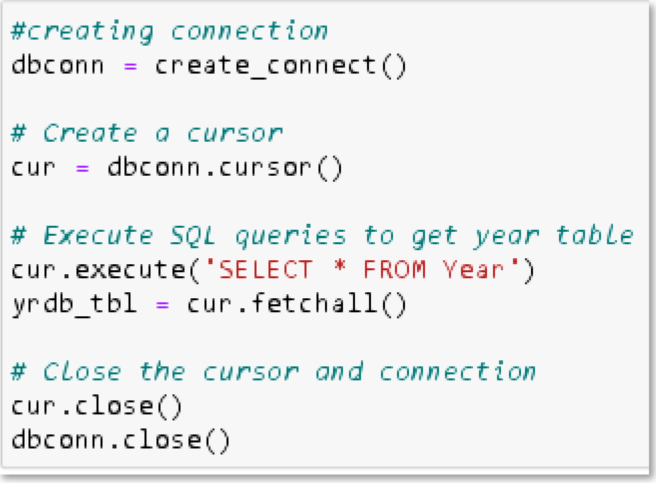


```
# for us to perform queries in povdb in PostgreSQL directly in the jupyter notebook.
%sql postgresql://postgres:      localhost/povdb
```



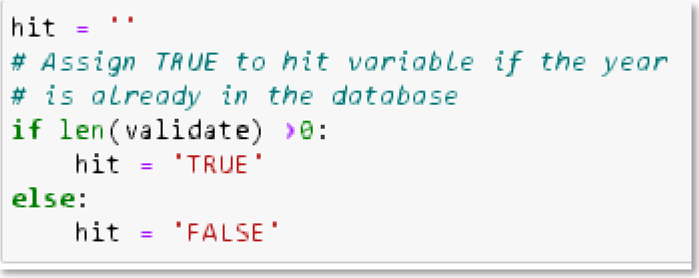
```
PGHOST = 'localhost'
PGDATABASE = 'povdb'
PGUSER = '          '
PGPASSWORD = '

def create_connect():
    dbconn = psycopg2.connect(
        dbname=PGDATABASE,
        user=PGUSER,
        password=PGPASSWORD,
        host=PGHOST,
        port='5432'
    )
    return dbconn
```

Get the year table in the database povdb to be evaluated

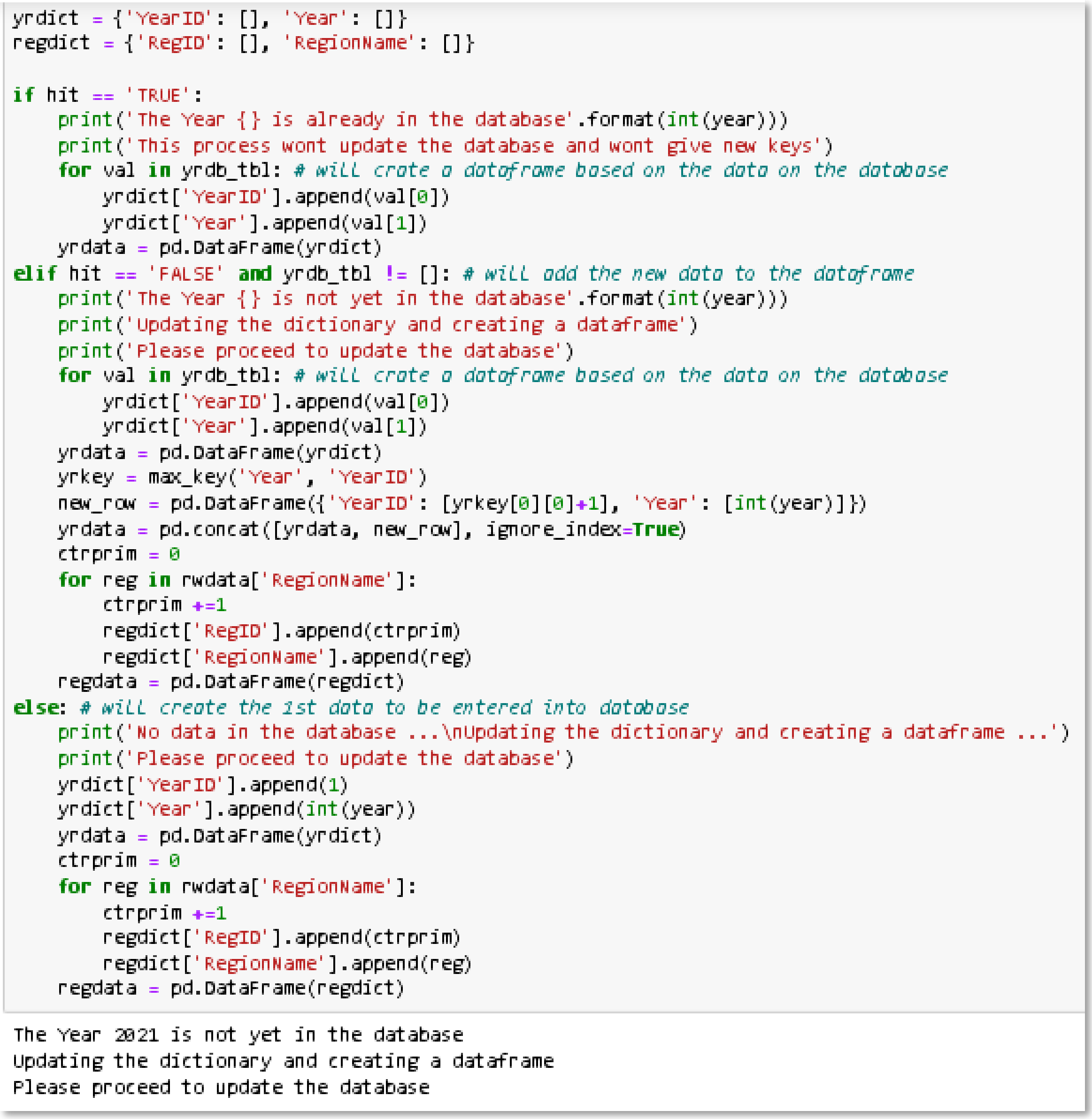


```
#creating connection
dbconn = create_connect()

# Create a cursor
cur = dbconn.cursor()

# Execute SQL queries to get year table
cur.execute('SELECT * FROM Year')
yrdb_tbl = cur.fetchall()

# Close the cursor and connection
cur.close()
dbconn.close()
```

The code below with "hit:" variable will be assigned a "TRUE" value if the 2021 data are already in the database and "FALSE" if otherwise.



```
hit = ''
# Assign TRUE to hit variable if the year
# is already in the database
if len(validate) >0:
    hit = 'TRUE'
else:
    hit = 'FALSE'
```



```
yrdict = {'YearID': [], 'Year': []}
regdict = {'RegID': [], 'RegionName': []}

if hit == 'TRUE':
    print('The Year {} is already in the database'.format(int(year)))
    print('This process wont update the database and wont give new keys')
    for val in yrdb_tbl: # will crate a dataframe based on the data on the database
        yrdict['YearID'].append(val[0])
        yrdict['Year'].append(val[1])
    yrdata = pd.DataFrame(yrdict)
elif hit == 'FALSE' and yrdb_tbl != []: # will add the new data to the dataframe
    print('The Year {} is not yet in the database'.format(int(year)))
    print('Updating the dictionary and creating a dataframe')
    print('Please proceed to update the database')
    for val in yrdb_tbl: # will crate a dataframe based on the data on the database
        yrdict['YearID'].append(val[0])
        yrdict['Year'].append(val[1])
    yrdata = pd.DataFrame(yrdict)
    yrkey = max_key('Year', 'YearID')
    new_row = pd.DataFrame({'YearID': [yrkey[0][0]+1], 'Year': [int(year)]})
    yrdata = pd.concat([yrdata, new_row], ignore_index=True)
    ctrprim = 0
    for reg in rwdata['RegionName']:
        ctrprim +=1
        regdict['RegID'].append(ctrprim)
        regdict['RegionName'].append(reg)
    regdata = pd.DataFrame(regdict)
else: # will create the 1st data to be entered into database
    print('No data in the database ...\nUpdating the dictionary and creating a dataframe ...')
    print('Please proceed to update the database')
    yrdict['YearID'].append(1)
    yrdict['Year'].append(int(year))
    yrdata = pd.DataFrame(yrdict)
    ctrprim = 0
    for reg in rwdata['RegionName']:
        ctrprim +=1
        regdict['RegID'].append(ctrprim)
        regdict['RegionName'].append(reg)
    regdata = pd.DataFrame(regdict)
```

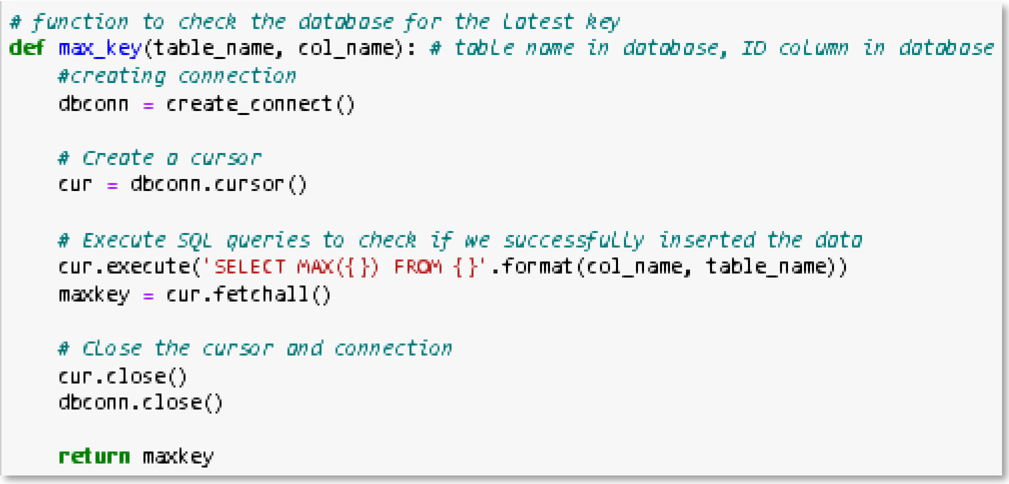
```
The Year 2021 is not yet in the database
Updating the dictionary and creating a dataframe
Please proceed to update the database
```

The code above is a simple if statement implemented to determine whether a newly imported file constitutes new data or if its already exists in the database. For instance, if the code has been executed multiple times, the output will indicate, “The Year 2021 is already in the database.” This mechanism serves the dual purpose of ensuring data integrity and preventing duplicate entries in the database by avoiding the assignment of new primary and foreign keys during the transformation process.

I. 3 TRANSFORM

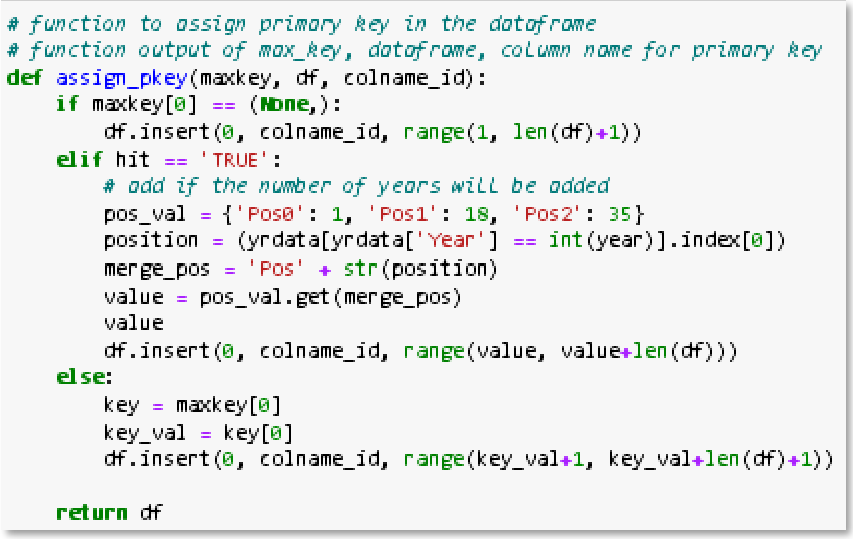
During this stage, the data frame will be partitioned into its respective categories, and primary as well as foreign keys will be assigned. It's important to note that this assignment will only take place if the algorithm that was established during the evaluation phase returns a false outcome and hit variable will have the value of “FALSE”, signifying that the data is not yet present in the database.

Check for the max key value in the database (function)



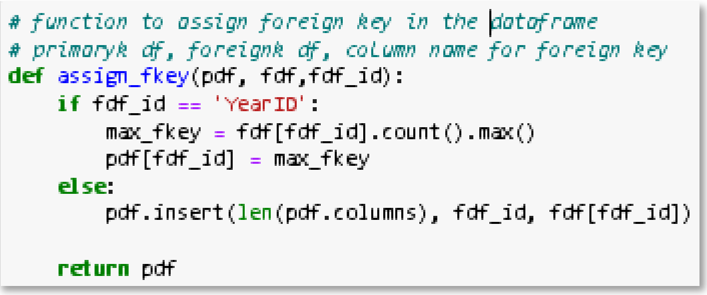
```
# function to check the database for the latest key
def max_key(table_name, col_name): # table name in database, ID column in database
    #creating connection
    dbconn = create_connect()

    # Create a cursor
    cur = dbconn.cursor()

    # Execute SQL queries to check if we successfully inserted the data
    cur.execute('SELECT MAX({}) FROM {}'.format(col_name, table_name))
    maxkey = cur.fetchall()

    # Close the cursor and connection
    cur.close()
    dbconn.close()

    return maxkey
```

Assign pkey (function)



```
# function to assign primary key in the dataframe
# function output of max_key, dataframe, column name for primary key
def assign_pkey(maxkey, df, colname_id):
    if maxkey[0] == (None,):
        df.insert(0, colname_id, range(1, len(df)+1))
    elif hit == 'TRUE':
        # add if the number of years will be added
        pos_val = {'Pos0': 1, 'Pos1': 18, 'Pos2': 35}
        position = (yrdata[yrdata['Year'] == int(year)].index[0])
        merge_pos = 'Pos' + str(position)
        value = pos_val.get(merge_pos)
        value
        df.insert(0, colname_id, range(value, value+len(df)))
    else:
        key = maxkey[0]
        key_val = key[0]
        df.insert(0, colname_id, range(key_val+1, key_val+len(df)+1))

    return df
```

Assign fkey (function)

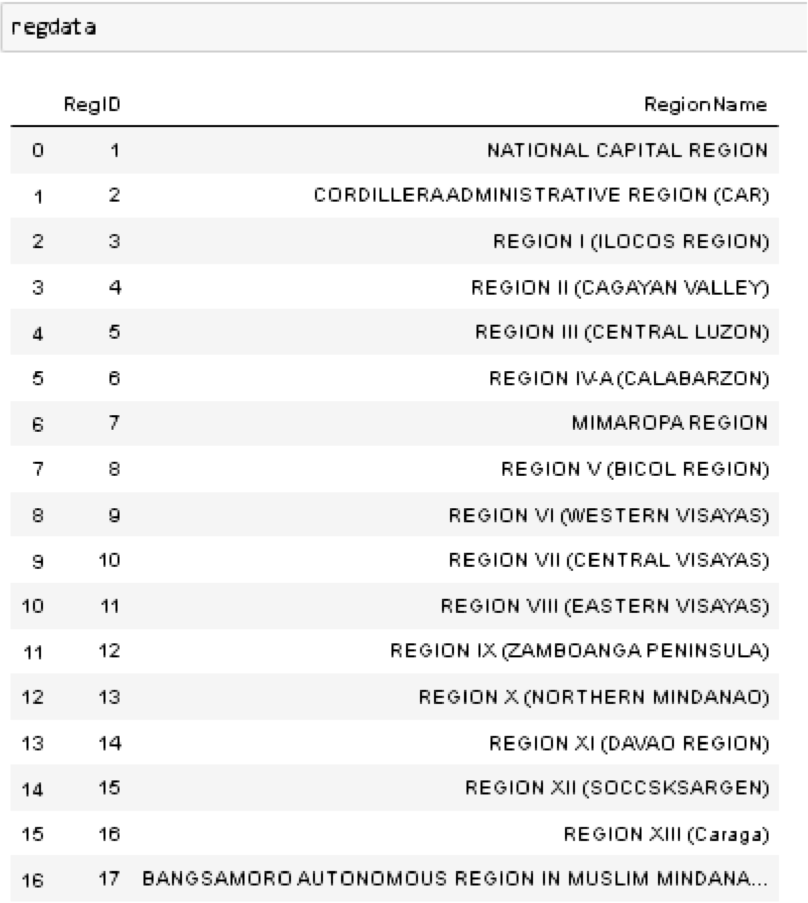


```
# function to assign foreign key in the dataframe
# primaryk df, foreignk df, column name for foreign key
def assign_fkey(pdf, fdf,fdf_id):
    if fdf_id == 'YearID':
        max_fkey = fdf[fdf_id].count().max()
        pdf[fdf_id] = max_fkey
    else:
        pdf.insert(len(pdf.columns), fdf_id, fdf[fdf_id])

    return pdf
```

The max_key function will determine the most recent primary key in the povdb database. For instance, when it examines the PovID column (the primary key) of the Poverty table for the first time, the max_key function returns the value of (None,), indicating that no data has been loaded yet. This returned value will be used in the conditional statement within the assign_pkey function.

For example, when the max_key value is 34, the assign_pkey function proceeds to assign primary key values ranging from 35 to the 'n' rows of data in the 'Poverty' column. This process is also applicable to the assign_fkey function. However, a minor adjustment is made in the conditional statement due to the relationship between the 'Year' and 'Region' tables versus the 'Poverty', 'Employment', 'Income', and 'Population' tables, as shown on page 7.

Region

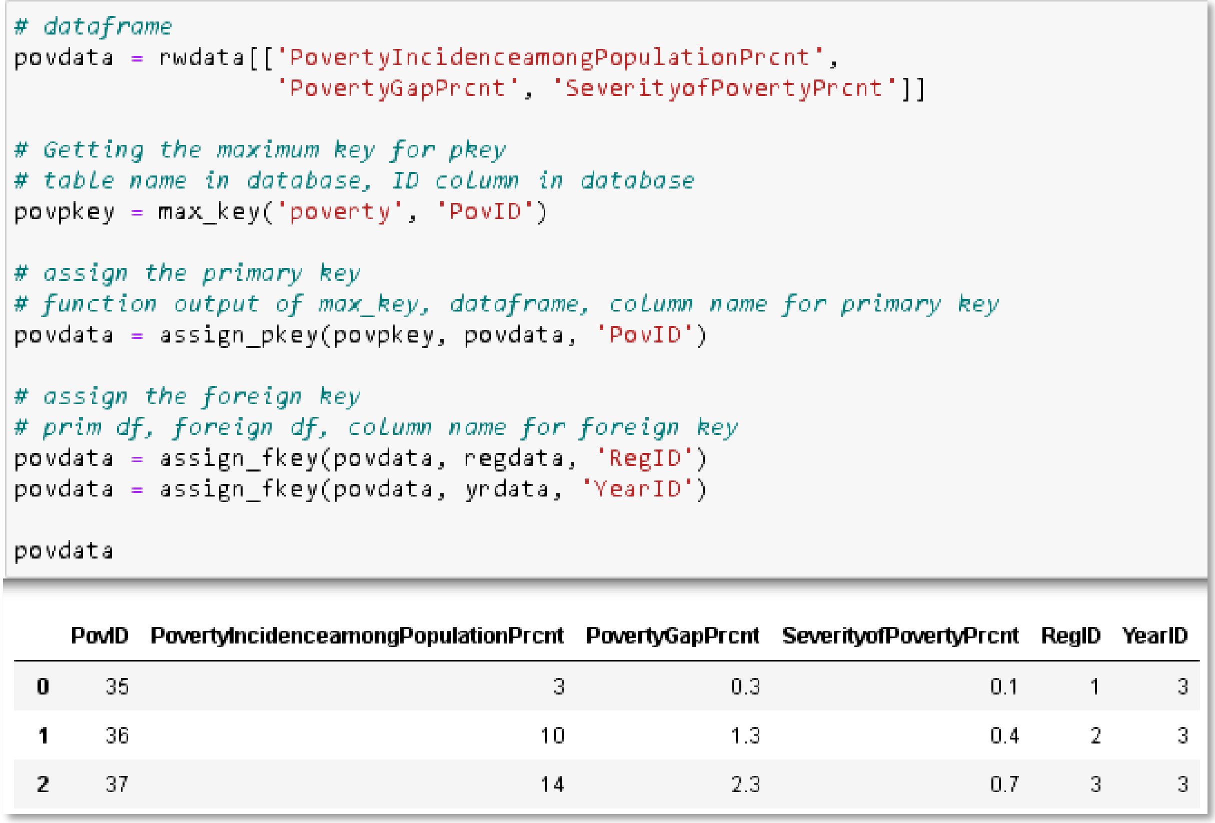


```
regdata
```

	RegID	RegionName
0	1	NATIONAL CAPITAL REGION
1	2	CORDILLERAADMINISTRATIVE REGION (CAR)
2	3	REGION I (ILOCOS REGION)
3	4	REGION II (CAGAYAN VALLEY)
4	5	REGION III (CENTRAL LUZON)
5	6	REGION IV-A (CALABARZON)
6	7	MIMAROPA REGION
7	8	REGION V (BICOL REGION)
8	9	REGION VI (WESTERN VISAYAS)
9	10	REGION VII (CENTRAL VISAYAS)
10	11	REGION VIII (EASTERN VISAYAS)
11	12	REGION IX (ZAMBOANGA PENINSULA)
12	13	REGION X (NORTHERN MINDANAO)
13	14	REGION XI (DAVAO REGION)
14	15	REGION XII (SOCCSKSARGEN)
15	16	REGION XIII (Caraga)
16	17	BANGSAMORO AUTONOMOUS REGION IN MUSLIM MINDANA...

The Region table above contains static values and is classified as a lookup table since its primary purpose is to standardize and categorize data in other tables.

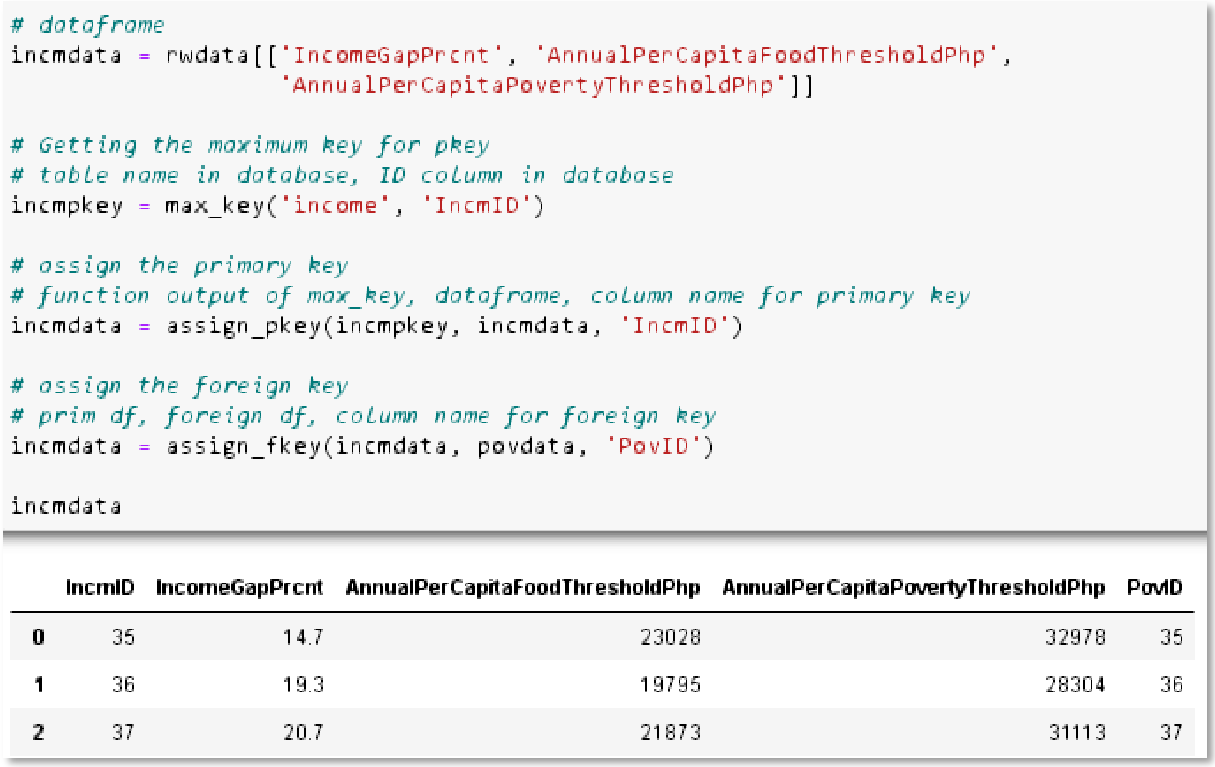
Poverty



```
# dataframe
povdata = rwdata[['PovertyIncidenceamongPopulationPrcnt',
                  'PovertyGapPrcnt', 'SeverityofPovertyPrcnt']]

# Getting the maximum key for pkey
# table name in database, ID column in database
povpkey = max_key('poverty', 'PovID')

# assign the primary key
# function output of max_key, dataframe, column name for primary key
povdata = assign_pkey(povpkey, povdata, 'PovID')

# assign the foreign key
# prim df, foreign df, column name for foreign key
povdata = assign_fkey(povdata, regdata, 'RegID')
povdata = assign_fkey(povdata, yrdata, 'YearID')

povdata
```

	PovID	PovertyIncidenceamongPopulationPrcnt	PovertyGapPrcnt	SeverityofPovertyPrcnt	RegID	YearID
0	35	3	0.3	0.1	1	3
1	36	10	1.3	0.4	2	3
2	37	14	2.3	0.7	3	3

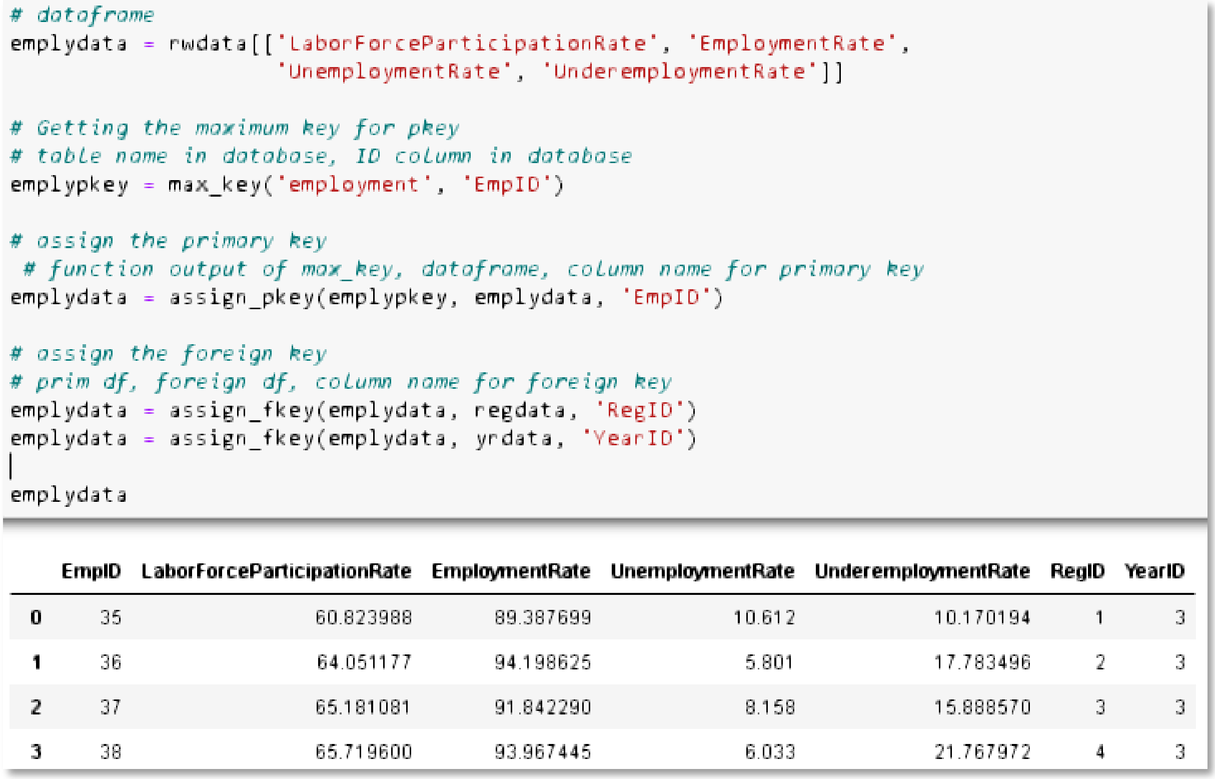
Income



```
# dataframe
incmdata = rwdata[['IncomeGapPrcnt', 'AnnualPerCapitaFoodThresholdPhp',
                   'AnnualPerCapitaPovertyThresholdPhp']]

# Getting the maximum key for pkey
# table name in database, ID column in database
incmpkey = max_key('income', 'IncmID')

# assign the primary key
# function output of max_key, dataframe, column name for primary key
incmdata = assign_pkey(incmpkey, incmdata, 'IncmID')

# assign the foreign key
# prim df, foreign df, column name for foreign key
incmdata = assign_fkey(incmdata, povdata, 'PovID')

incmdata
```

	IncmID	IncomeGapPrcnt	AnnualPerCapitaFoodThresholdPhp	AnnualPerCapitaPovertyThresholdPhp	PovID
0	35	14.7	23028	32978	35
1	36	19.3	19795	28304	36
2	37	20.7	21873	31113	37

Employment

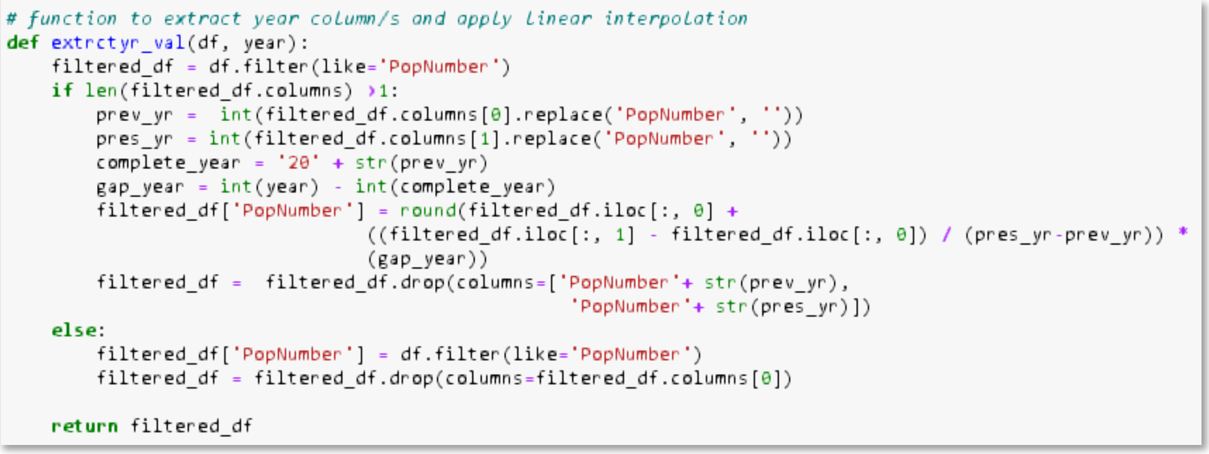


```
# dataframe
emplydata = rwdata[['LaborForceParticipationRate', 'EmploymentRate',
                    'UnemploymentRate', 'UnderemploymentRate']]

# Getting the maximum key for pkey
# table name in database, ID column in database
emplypkey = max_key('employment', 'EmpID')

# assign the primary key
 # function output of max_key, dataframe, column name for primary key
emplydata = assign_pkey(emplypkey, emplydata, 'EmpID')

# assign the foreign key
# prim df, foreign df, column name for foreign key
emplydata = assign_fkey(emplydata, regdata, 'RegID')
emplydata = assign_fkey(emplydata, yrdata, 'YearID')
|
emplydata
```

	EmpID	LaborForceParticipationRate	EmploymentRate	UnemploymentRate	UnderemploymentRate	RegID	YearID
0	35	60.823988	89.387699	10.612	10.170194	1	3
1	36	64.051177	94.198625	5.801	17.783496	2	3
2	37	65.181081	91.842290	8.158	15.888570	3	3
3	38	65.719600	93.967445	6.033	21.767972	4	3

Population

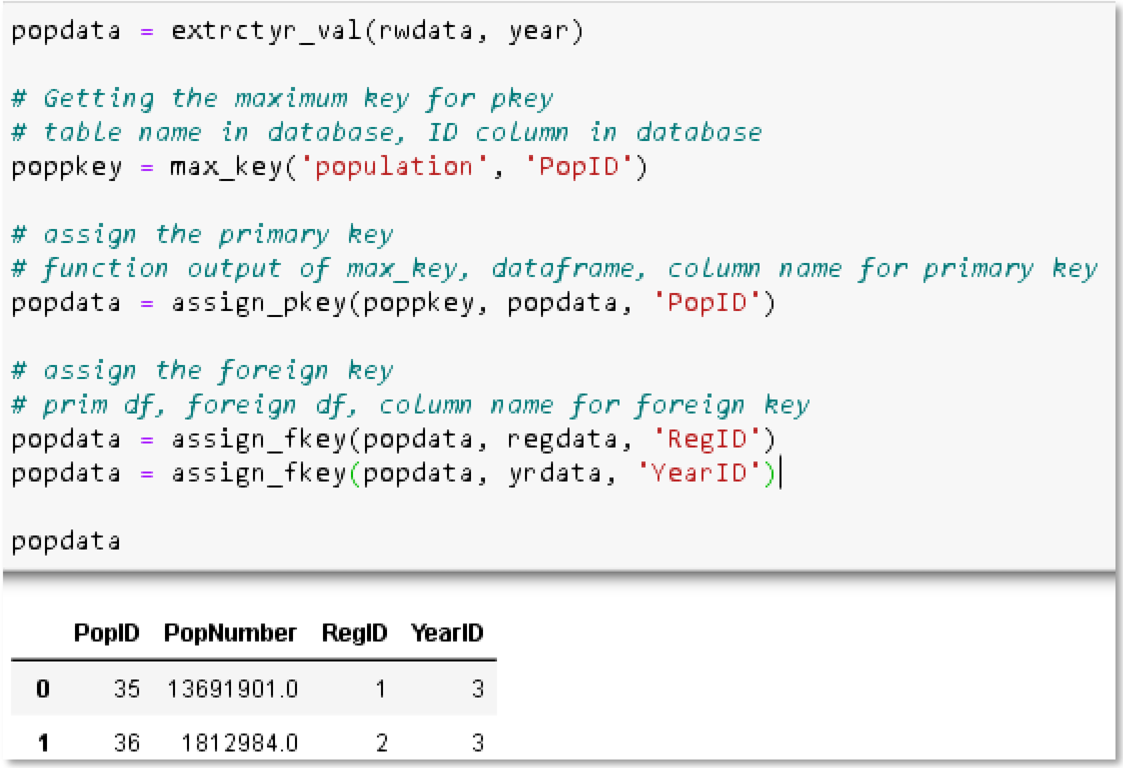
This is where the Linear Interpolation formula will be used and place inside a function to align the population dataset with poverty, income, and employment data. The function below will also output non-interpolated data if there are no two columns consisting population data in the file, similar to the ‘Aggregated data 2015’.



```
# function to extract year column/s and apply linear interpolation
def extrctyr_val(df, year):
    filtered_df = df.filter(like='PopNumber')
    if len(filtered_df.columns) >1:
        prev_yr =  int(filtered_df.columns[0].replace('PopNumber', ''))
        pres_yr = int(filtered_df.columns[1].replace('PopNumber', ''))
        complete_year = '20' + str(prev_yr)
        gap_year = int(year) - int(complete_year)
        filtered_df['PopNumber'] = round(filtered_df.iloc[:, 0] +
                                   ((filtered_df.iloc[:, 1] - filtered_df.iloc[:, 0]) / (pres_yr-prev_yr)) *
                                   (gap_year))
        filtered_df =  filtered_df.drop(columns=['PopNumber'+ str(prev_yr),
                                                 'PopNumber'+ str(pres_yr)])
    else:
        filtered_df['PopNumber'] = df.filter(like='PopNumber')
        filtered_df = filtered_df.drop(columns=filtered_df.columns[0])

    return filtered_df
```

For the year 2021 dataset, linear interpolation will be performed since the if statement will be true as there are two columns with population data in the file, namely 'PopNumber20' and 'PopNumber25 as shown in page 12.

Population (Interpolated)



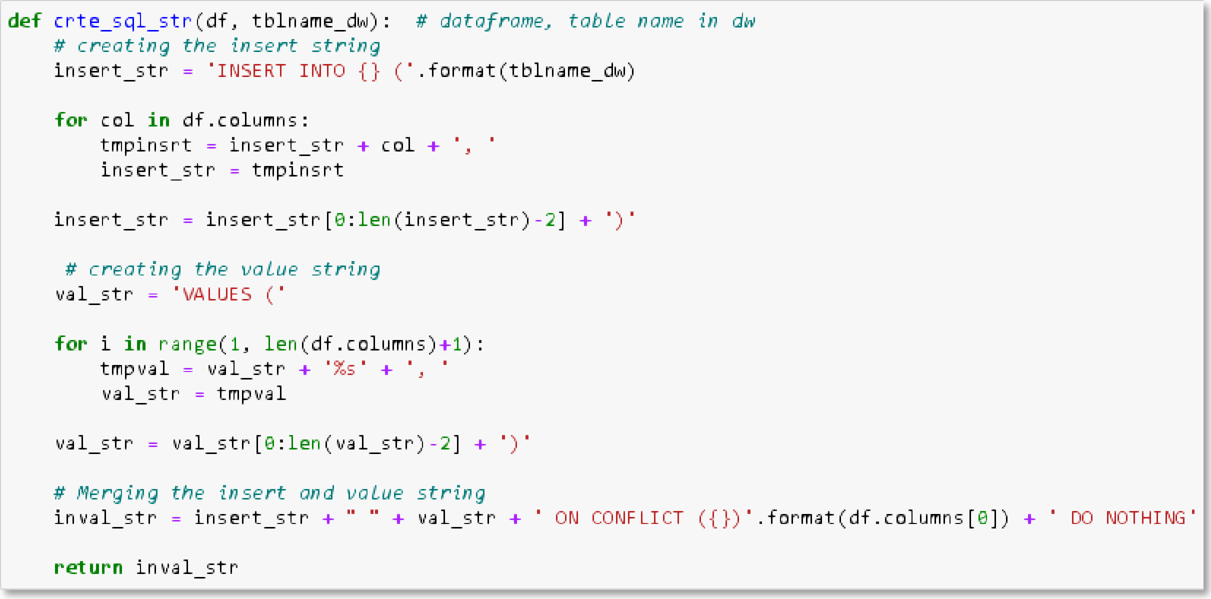
```
popdata = extrctyr_val(rwdata, year)

# Getting the maximum key for pkey
# table name in database, ID column in database
poppkey = max_key('population', 'PopID')

# assign the primary key
# function output of max_key, dataframe, column name for primary key
popdata = assign_pkey(poppkey, popdata, 'PopID')

# assign the foreign key
# prim df, foreign df, column name for foreign key
popdata = assign_fkey(popdata, regdata, 'RegID')
popdata = assign_fkey(popdata, yrdata, 'YearID')

popdata
```

```
   PopID  PopNumber   RegID  YearID
0     35  13691901.0      1       3
1     36   1812984.0      2       3
```

I.4 LOAD

Load the dataset into povdb database after the transformation process.

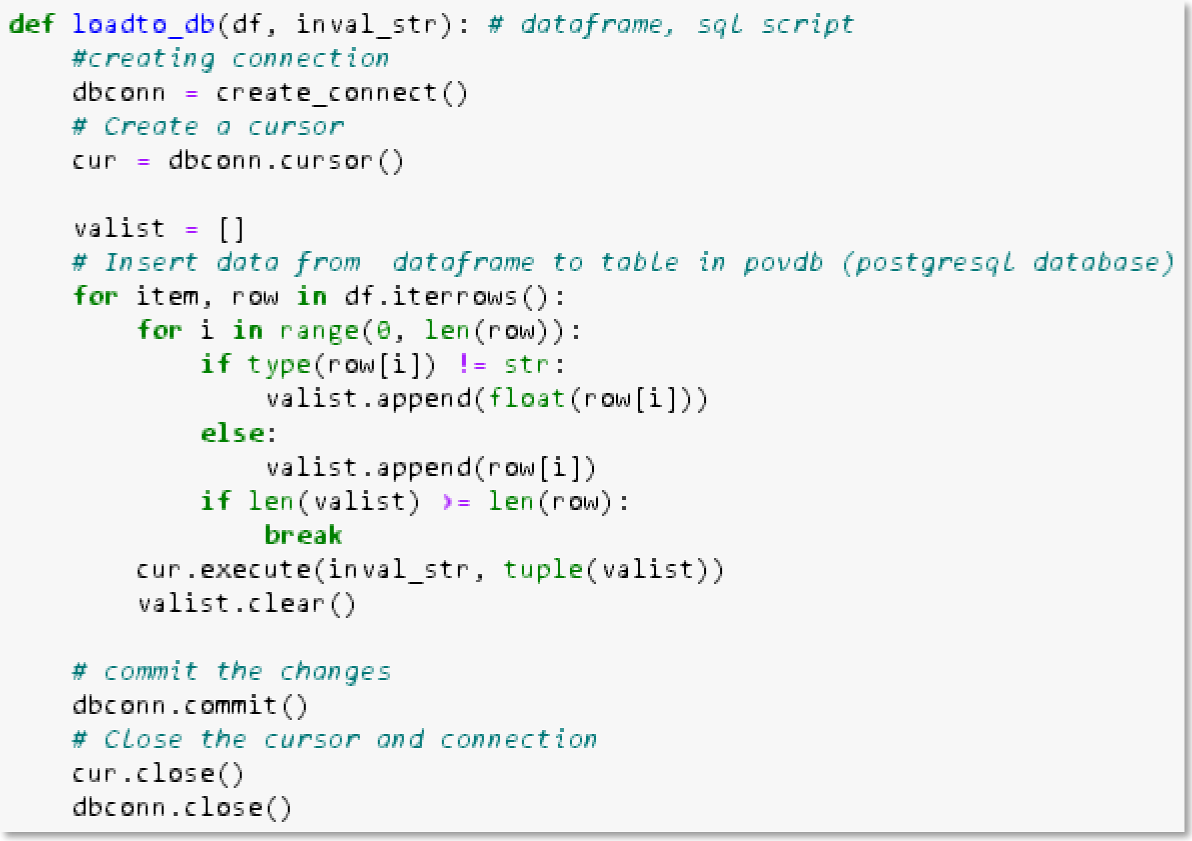
Create SQL script for loading data (function)



```
def crte_sql_str(df, tblname_dw):  # dataframe, table name in dw
    # creating the insert string
    insert_str = 'INSERT INTO {} ('.format(tblname_dw)

    for col in df.columns:
        tmpinsrt = insert_str + col + ', '
        insert_str = tmpinsrt

    insert_str = insert_str[0:len(insert_str)-2] + ')'

     # creating the value string
    val_str = 'VALUES ('

    for i in range(1, len(df.columns)+1):
        tmpval = val_str + '%s' + ', '
        val_str = tmpval

    val_str = val_str[0:len(val_str)-2] + ')'

    # Merging the insert and value string
    inval_str = insert_str + " " + val_str + ' ON CONFLICT ({})'.format(df.columns[0]) + ' DO NOTHING'

    return inval_str
```

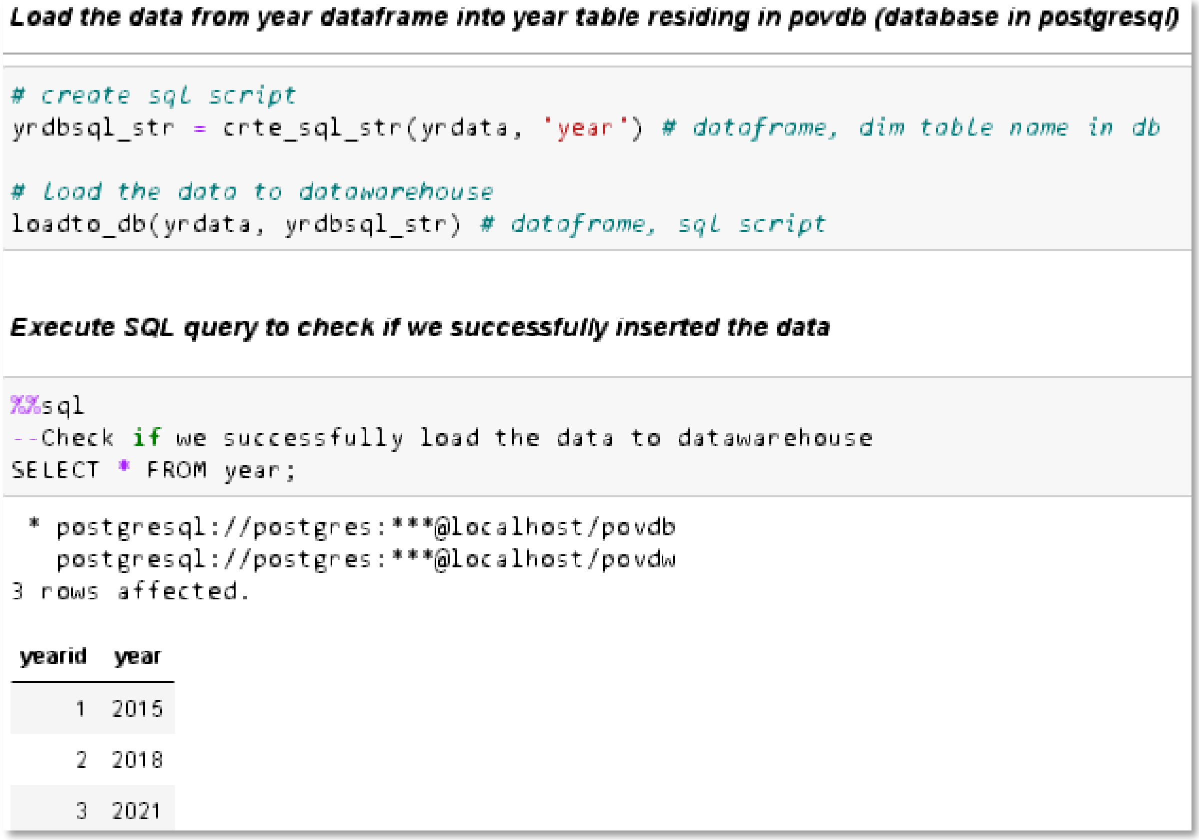
The crte_sql_str function above will output a string of SQL syntax, this syntax will be use in the loadto_db function to load the data into povdb database.

Load data to database (function)



```
def loadto_db(df, inval_str): # dataframe, sql script
    #creating connection
    dbconn = create_connect()
    # Create a cursor
    cur = dbconn.cursor()

    valist = []
    # Insert data from  dataframe to table in povdb (postgresql database)
    for item, row in df.iterrows():
        for i in range(0, len(row)):
            if type(row[i]) != str:
                valist.append(float(row[i]))
            else:
                valist.append(row[i])
            if len(valist) >= len(row):
                break
        cur.execute(inval_str, tuple(valist))
        valist.clear()

    # commit the changes
    dbconn.commit()
    # Close the cursor and connection
    cur.close()
    dbconn.close()
```

Year



Load the data from year dataframe into year table residing in povdb (database in postgresql)

```
# create sql script
yrdbsql_str = crte_sql_str(yrdata, 'year') # dataframe, dim table name in db

# Load the data to datawarehouse
loadto_db(yrdata, yrdbsql_str) # dataframe, sql script
```

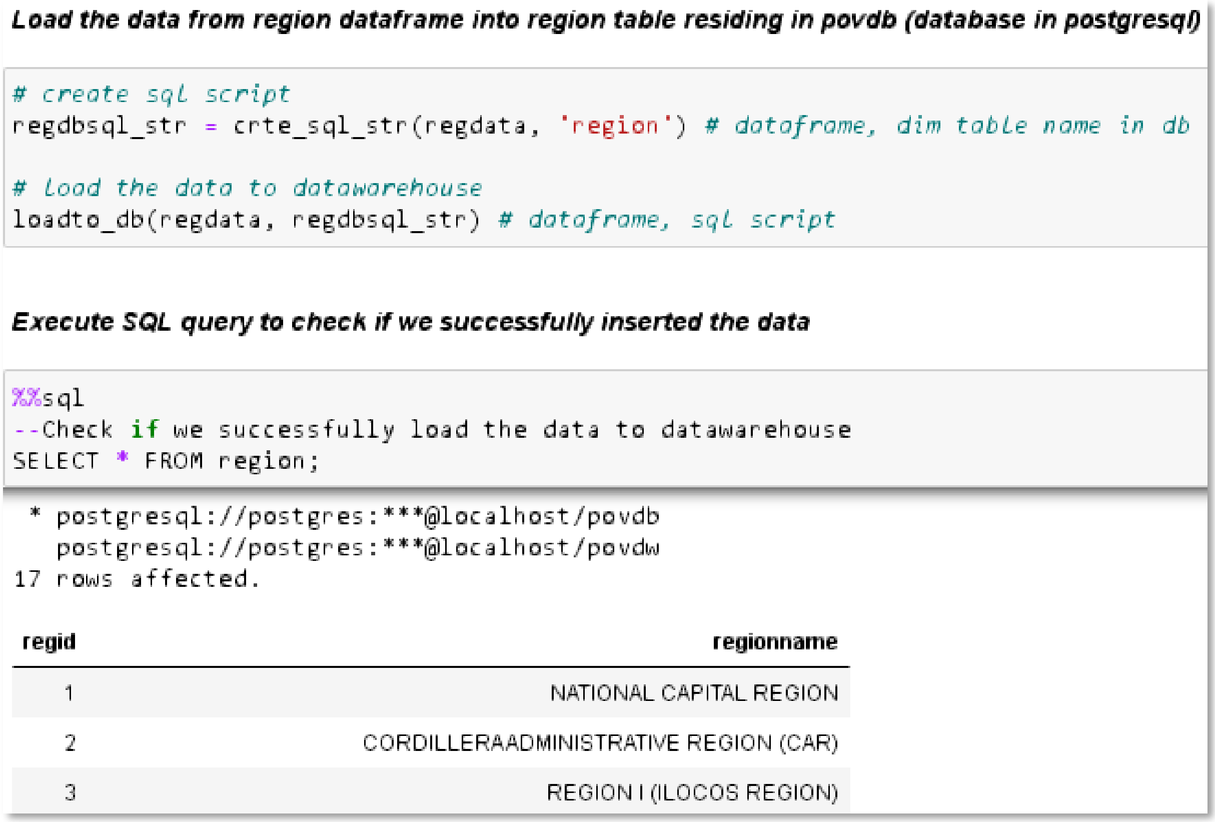
Execute SQL query to check if we successfully inserted the data

```
%%sql
--Check if we successfully load the data to datawarehouse
SELECT * FROM year;
```

```
 * postgresql://postgres:***@localhost/povdb
   postgresql://postgres:***@localhost/povdw
3 rows affected.
```

yearid	year
1	2015
2	2018
3	2021

Region



Load the data from region dataframe into region table residing in povdb (database in postgresql)

```
# create sql script
regdbsql_str = crte_sql_str(regdata, 'region') # dataframe, dim table name in db

# load the data to datawarehouse
loadto_db(regdata, regdbsql_str) # dataframe, sql script
```

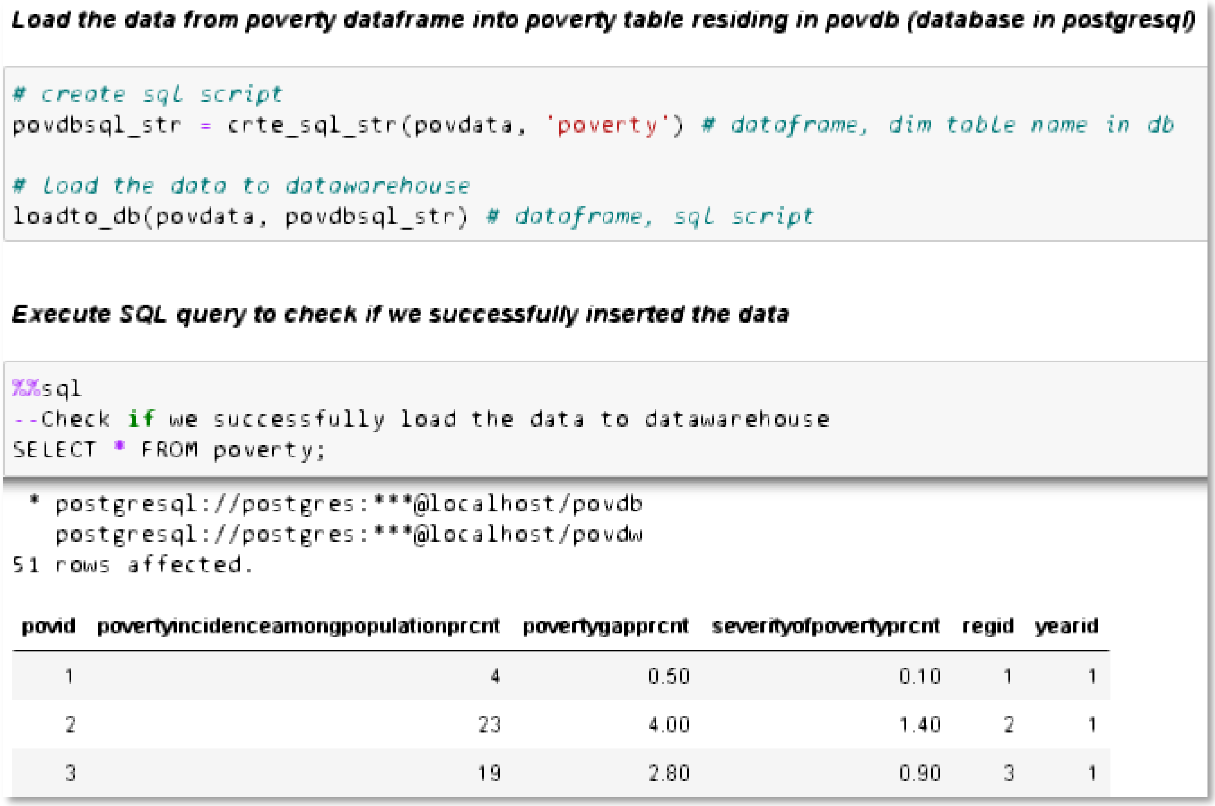
Execute SQL query to check if we successfully inserted the data

```
%%sql
--Check if we successfully load the data to datawarehouse
SELECT * FROM region;
```

```
 * postgresql://postgres:***@localhost/povdb
   postgresql://postgres:***@localhost/povdw
17 rows affected.
```

regid	regionname
1	NATIONAL CAPITAL REGION
2	CORDILLERAADMINISTRATIVE REGION (CAR)
3	REGION I (ILOCOS REGION)

Poverty



Load the data from poverty dataframe into poverty table residing in povdb (database in postgresql)

```
# create sql script
povdbsql_str = crte_sql_str(povdata, 'poverty') # dataframe, dim table name in db

# load the data to datawarehouse
loadto_db(povdata, povdbsql_str) # dataframe, sql script
```

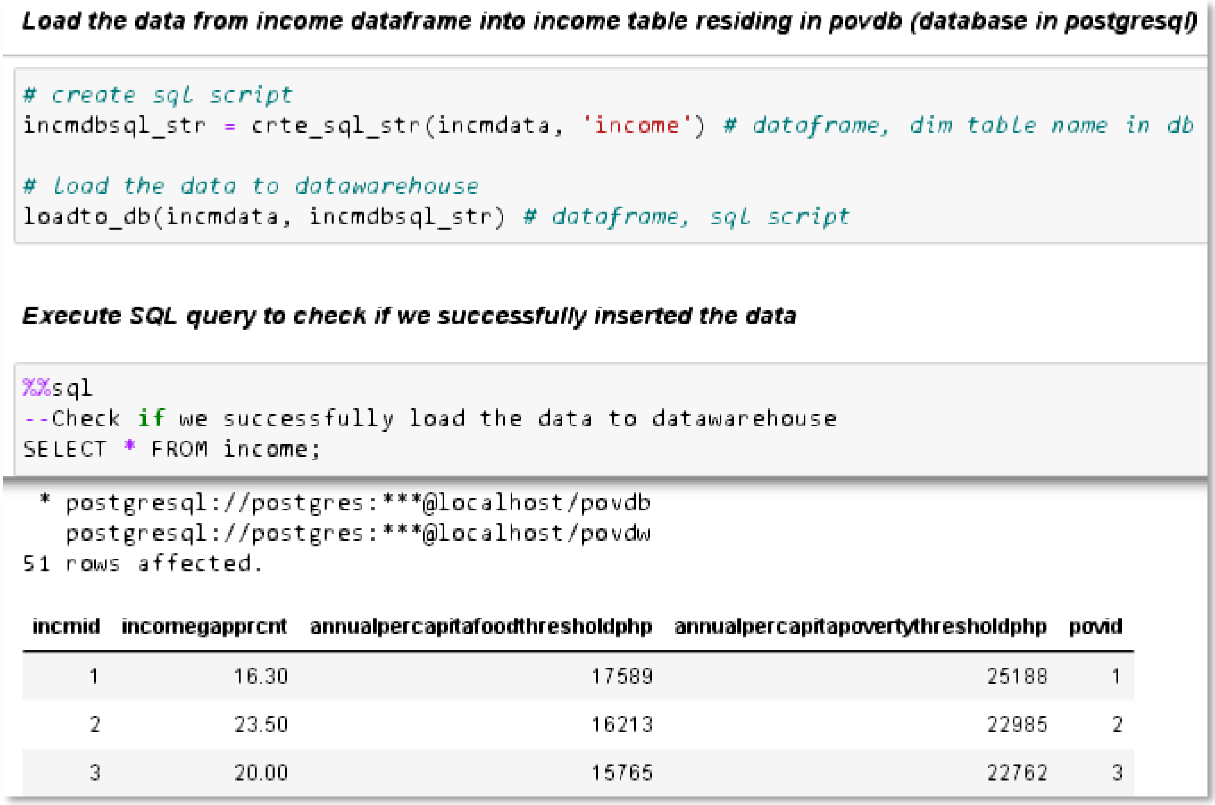
Execute SQL query to check if we successfully inserted the data

```
%%sql
--Check if we successfully load the data to datawarehouse
SELECT * FROM poverty;
```

```
 * postgresql://postgres:***@localhost/povdb
   postgresql://postgres:***@localhost/povdw
51 rows affected.
```

povid	povertyincidenceamongpopulationprcnt	povertygapprcnt	severityofpovertyprcnt	regid	yearid
1	4	0.50	0.10	1	1
2	23	4.00	1.40	2	1
3	19	2.80	0.90	3	1

Income



Load the data from income dataframe into income table residing in povdb (database in postgresql)

```
# create sql script
incmdbsql_str = crte_sql_str(incmdata, 'income') # dataframe, dim table name in db

# load the data to datawarehouse
loadto_db(incmdata, incmdbsql_str) # dataframe, sql script
```

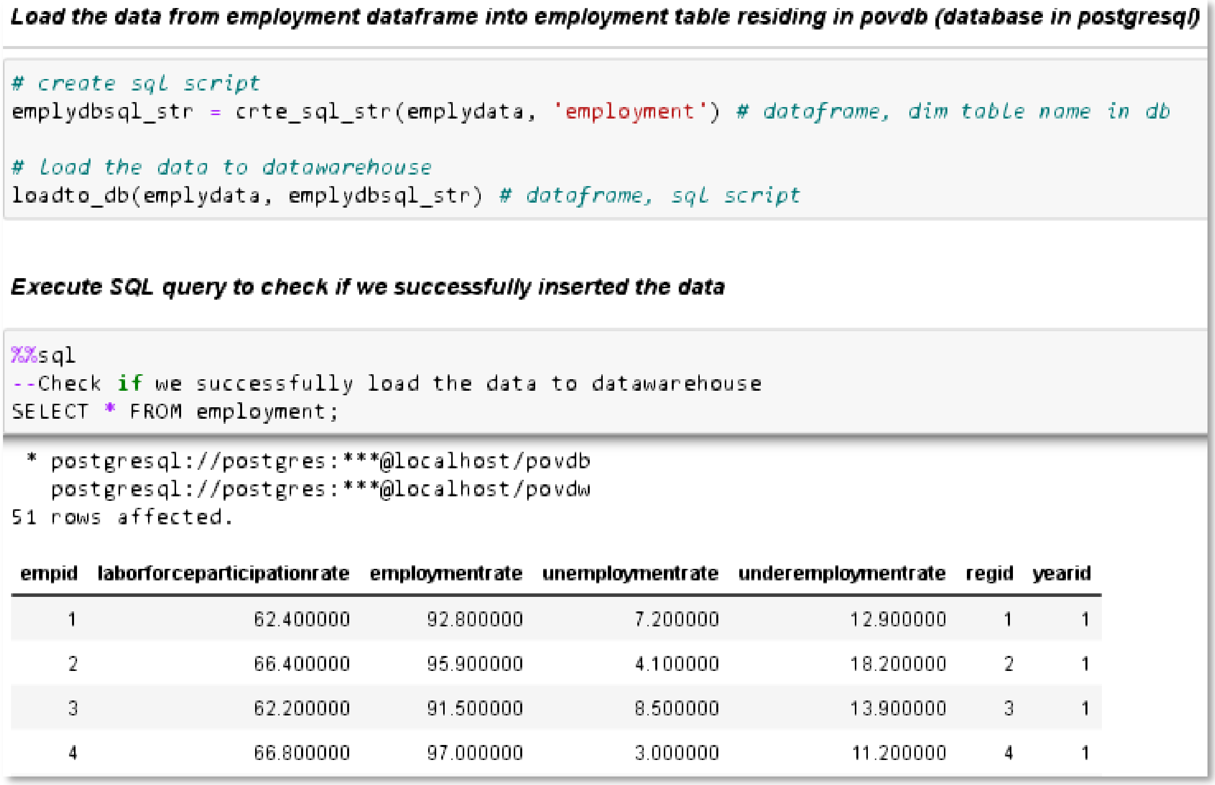
Execute SQL query to check if we successfully inserted the data

```
%%sql
--Check if we successfully load the data to datawarehouse
SELECT * FROM income;
```

```
 * postgresql://postgres:***@localhost/povdb
   postgresql://postgres:***@localhost/povdw
51 rows affected.
```

incmid	incomegapprcnt	annualpercapitafoodthresholdphp	annualpercapitapovertythresholdphp	povid
1	16.30	17589	25188	1
2	23.50	16213	22985	2
3	20.00	15765	22762	3

Employment



Load the data from employment dataframe into employment table residing in povdb (database in postgresql)

```
# create sql script
emplydbsql_str = crte_sql_str(emplydata, 'employment') # dataframe, dim table name in db

# load the data to datawarehouse
loadto_db(emplydata, emplydbsql_str) # dataframe, sql script
```

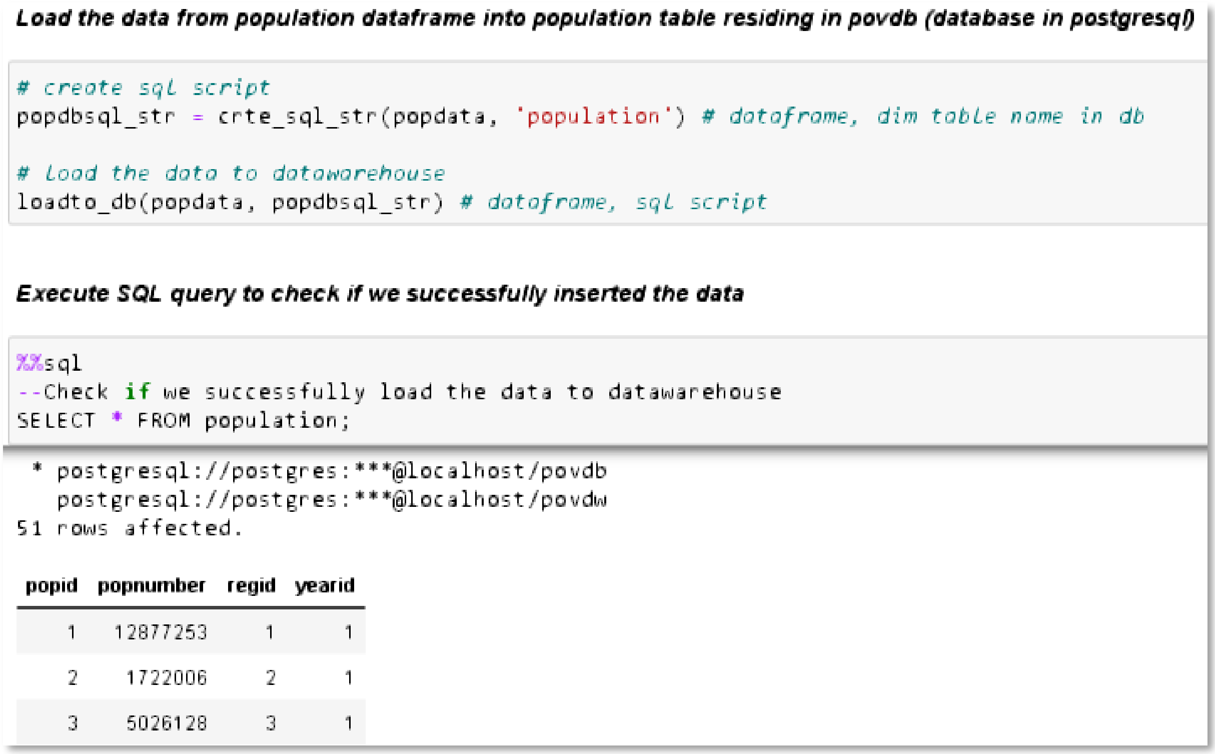
Execute SQL query to check if we successfully inserted the data

```
%%sql
--Check if we successfully load the data to datawarehouse
SELECT * FROM employment;
```

```
 * postgresql://postgres:***@localhost/povdb
   postgresql://postgres:***@localhost/povdw
51 rows affected.
```

empid	laborforceparticipationrate	employmentrate	unemploymentrate	underemploymentrate	regid	yearid
1	62.400000	92.800000	7.200000	12.900000	1	1
2	66.400000	95.900000	4.100000	18.200000	2	1
3	62.200000	91.500000	8.500000	13.900000	3	1
4	66.800000	97.000000	3.000000	11.200000	4	1

Population



Load the data from population dataframe into population table residing in povdb (database in postgresql)

```
# create sql script
popdbsql_str = crte_sql_str(popdata, 'population') # dataframe, dim table name in db

# load the data to datawarehouse
loadto_db(popdata, popdbsql_str) # dataframe, sql script
```

Execute SQL query to check if we successfully inserted the data

```
%%sql
--Check if we successfully load the data to datawarehouse
SELECT * FROM population;
```

```
 * postgresql://postgres:***@localhost/povdb
   postgresql://postgres:***@localhost/povdw
51 rows affected.
```

popid	popnumber	regid	yearid
1	12877253	1	1
2	1722006	2	1
3	5026128	3	1

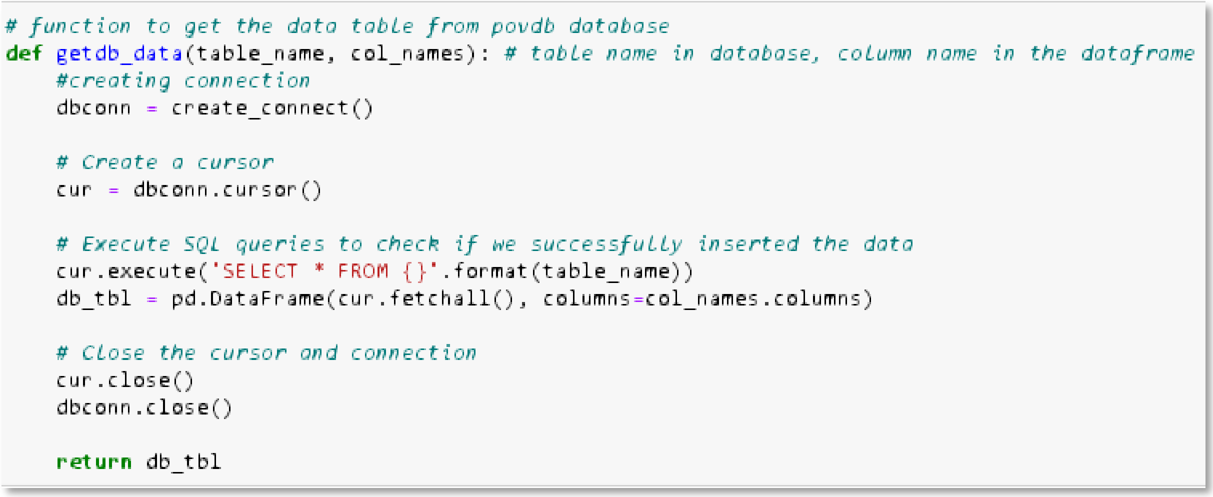
PHASE II

After the complete execution of Phase I pipeline, the Phase II will immediately be executed.

II.1 EXTRACT

Extracting the data from povdb database.

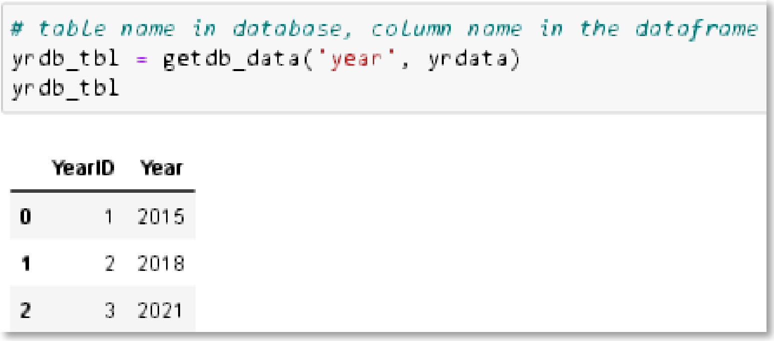
Fetch the data from the database povdb (function)



```
# function to get the data table from povdb database
def getdb_data(table_name, col_names): # table name in database, column name in the dataframe
    #creating connection
    dbconn = create_connect()

    # Create a cursor
    cur = dbconn.cursor()

    # Execute SQL queries to check if we successfully inserted the data
    cur.execute('SELECT * FROM {}'.format(table_name))
    db_tbl = pd.DataFrame(cur.fetchall(), columns=col_names.columns)

    # Close the cursor and connection
    cur.close()
    dbconn.close()

    return db_tbl
```

The function above will return a data frame after fetching the data from the 'povdb' database. The screenshot below will display only the first two rows of the entire data frame to conserve space in this documentation.

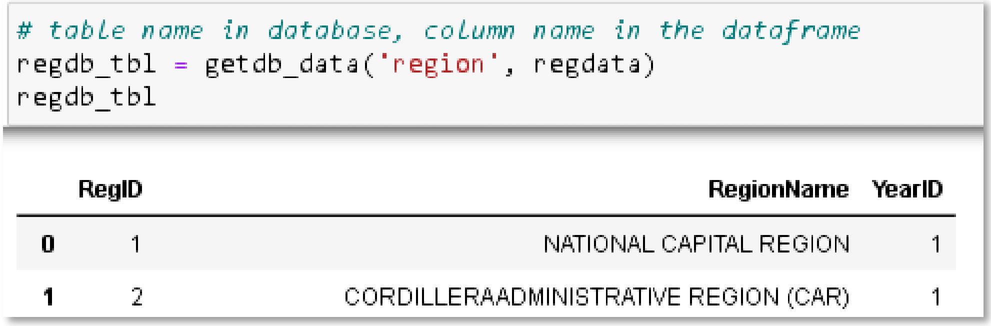
Year



```python
# table name in database, column name in the dataframe
yrdb_tbl = getdb_data('year', yrdata)
yrdb_tbl
```

	YearID	Year
0	1	2015
1	2	2018
2	3	2021

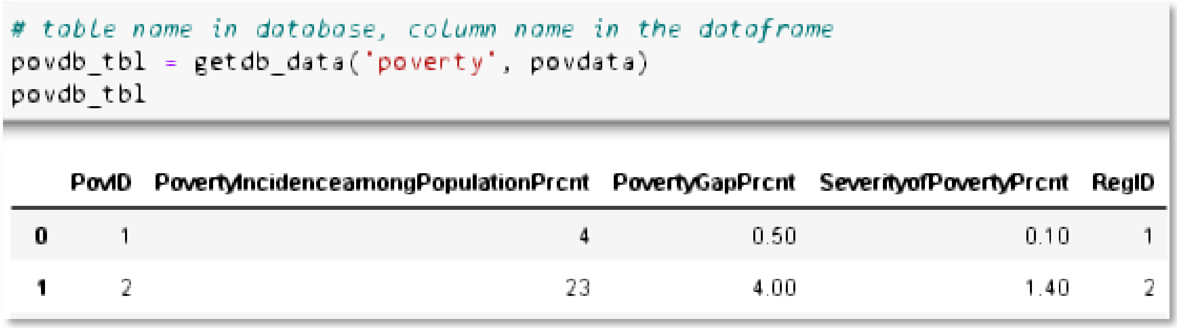
Region



```python
# table name in database, column name in the dataframe
regdb_tbl = getdb_data('region', regdata)
regdb_tbl
```

	RegID	RegionName	YearID
0	1	NATIONAL CAPITAL REGION	1
1	2	CORDILLERAADMINISTRATIVE REGION (CAR)	1

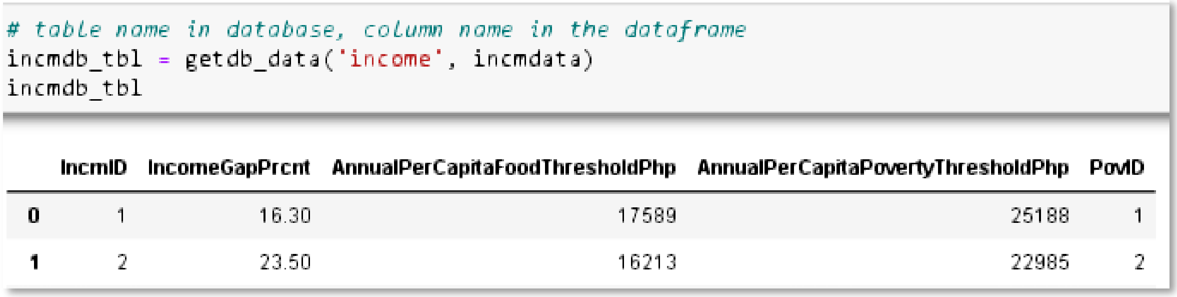
Poverty



```python
# table name in database, column name in the dataframe
povdb_tbl = getdb_data('poverty', povdata)
povdb_tbl
```

	PovID	PovertyIncidenceamongPopulationPrcnt	PovertyGapPrcnt	SeverityofPovertyPrcnt	RegID
0	1	4	0.50	0.10	1
1	2	23	4.00	1.40	2

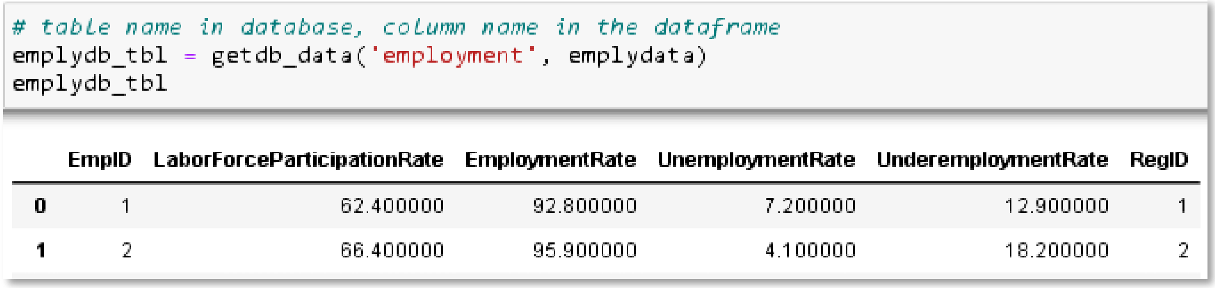
Income



```python
# table name in database, column name in the dataframe
incmdb_tbl = getdb_data('income', incmdata)
incmdb_tbl
```

	IncmID	IncomeGapPrcnt	AnnualPerCapitaFoodThresholdPhp	AnnualPerCapitaPovertyThresholdPhp	PovID
0	1	16.30	17589	25188	1
1	2	23.50	16213	22985	2

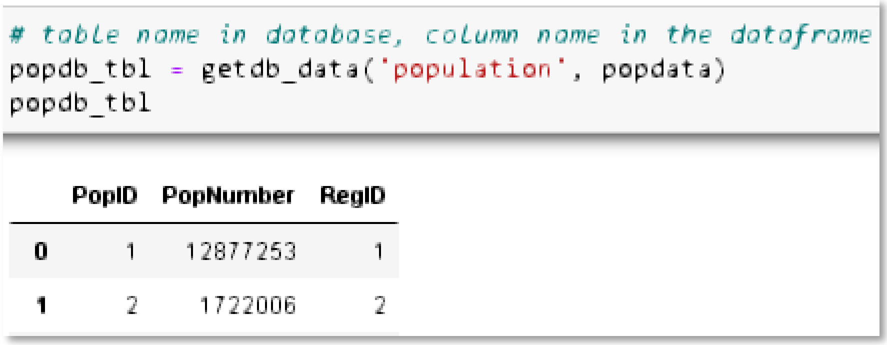
Employment



```python
# table name in database, column name in the dataframe
emplydb_tbl = getdb_data('employment', emplydata)
emplydb_tbl
```

	EmpID	LaborForceParticipationRate	EmploymentRate	UnemploymentRate	UnderemploymentRate	RegID
0	1	62.400000	92.800000	7.200000	12.900000	1
1	2	66.400000	95.900000	4.100000	18.200000	2

Population



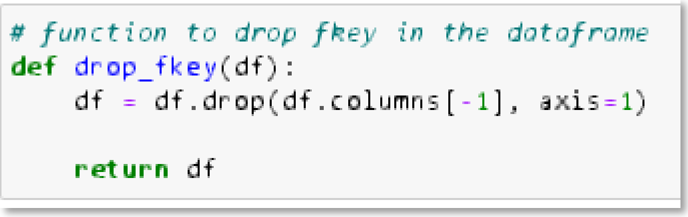
```
# table name in database, column name in the dataframe
popdb_tbl = getdb_data('population', popdata)
popdb_tbl
```

	PopID	PopNumber	RegID
0	1	12877253	1
1	2	1722006	2

II. 2 TRANSFORM

The data extracted from povdb database needs to be transformed to conform to the structure of dimension and fact table in data warehouse povdw except for the year and region table.

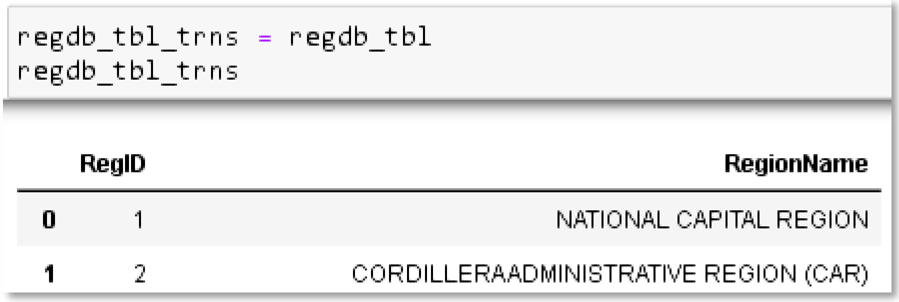
Removing the foreign key in the table (function)



```
# function to drop fkey in the dataframe
def drop_fkey(df):
    df = df.drop(df.columns[-1], axis=1)

    return df
```

Region

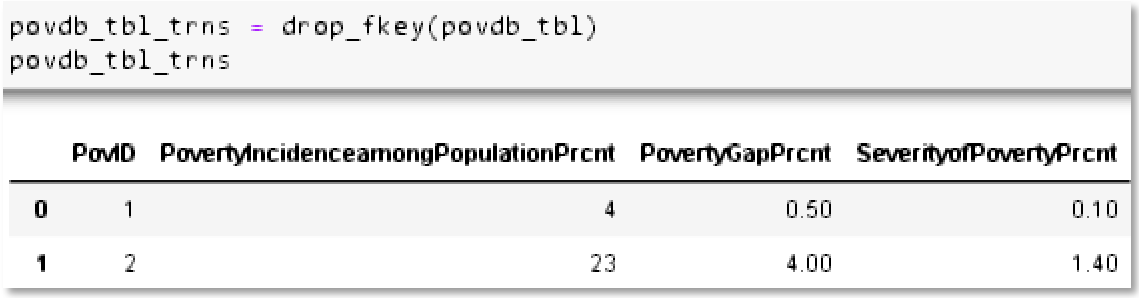
No transformation will be applied on region table as it is already conforming to the structure in data warehouse, however we need to assign the table into another variable for uniformity.



```
regdb_tbl_trns = regdb_tbl
regdb_tbl_trns
```

	RegID	RegionName
0	1	NATIONAL CAPITAL REGION
1	2	CORDILLERAADMINISTRATIVE REGION (CAR)

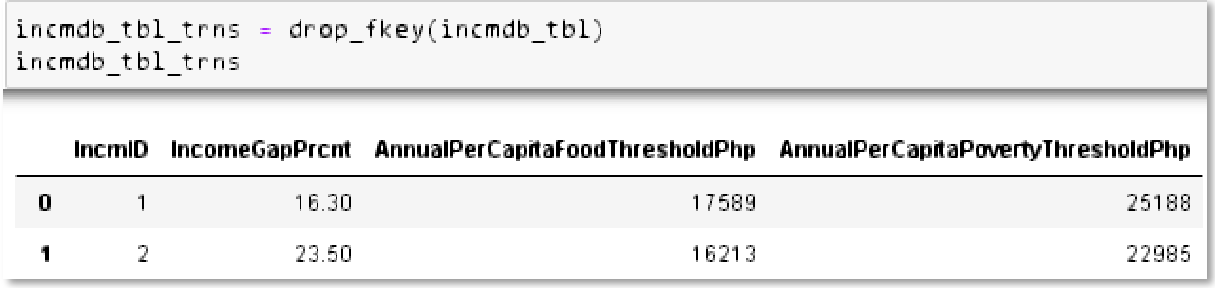
Poverty



```
povdb_tbl_trns = drop_fkey(povdb_tbl)
povdb_tbl_trns
```

	PovID	PovertyIncidenceamongPopulationPrcnt	PovertyGapPrcnt	SeverityofPovertyPrcnt
0	1	4	0.50	0.10
1	2	23	4.00	1.40

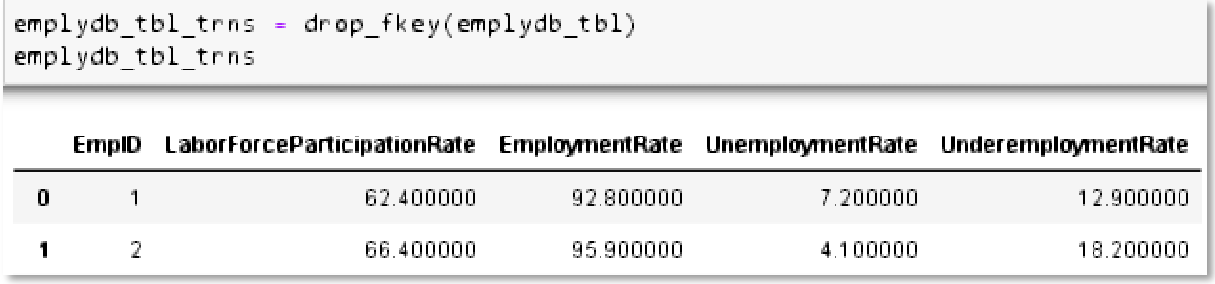
Income



```
incmdb_tbl_trns = drop_fkey(incmdb_tbl)
incmdb_tbl_trns
```

	IncmID	IncomeGapPrcnt	AnnualPerCapitaFoodThresholdPhp	AnnualPerCapitaPovertyThresholdPhp
0	1	16.30	17589	25188
1	2	23.50	16213	22985

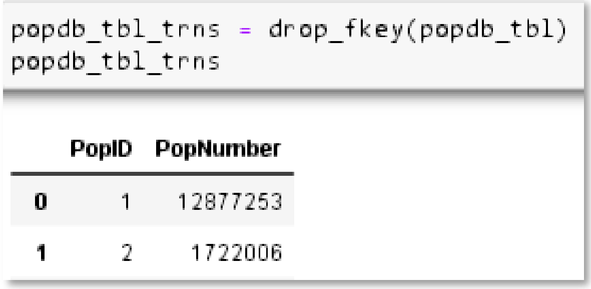
Employment



```
emplydb_tbl_trns = drop_fkey(emplydb_tbl)
emplydb_tbl_trns
```

	EmpID	LaborForceParticipationRate	EmploymentRate	UnemploymentRate	UnderemploymentRate
0	1	62.400000	92.800000	7.200000	12.900000
1	2	66.400000	95.900000	4.100000	18.200000

Population

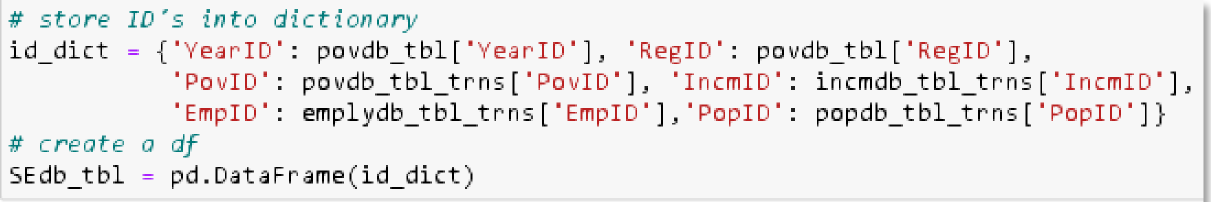


```
popdb_tbl_trns = drop_fkey(popdb_tbl)
popdb_tbl_trns
```

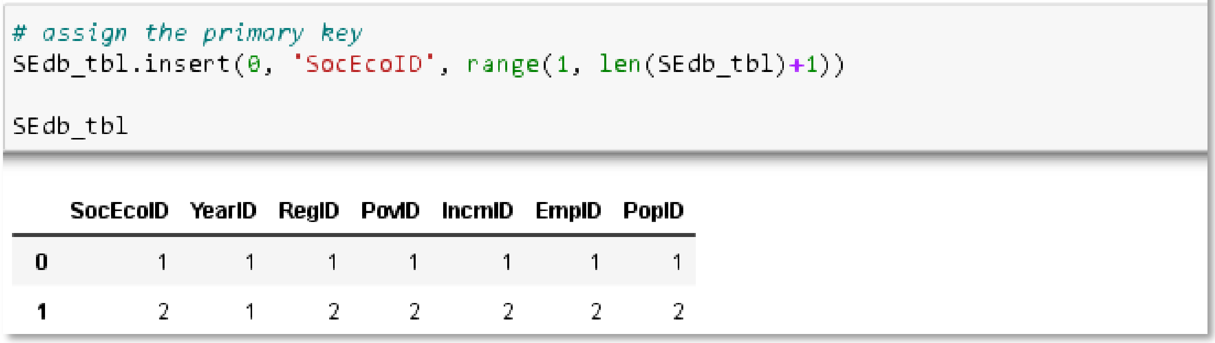
	PopID	PopNumber
0	1	12877253
1	2	1722006

Socio Economic

The Socio-Economic table will be loaded in the fact table. As mentioned previously on page 8, this fact table is called fact-less as it does not contain any calculated or aggregated data from the dimension tables, and this table will only serve as a bridge or connection.



```
# store ID's into dictionary
id_dict = {'YearID': povdb_tbl['YearID'], 'RegID': povdb_tbl['RegID'],
           'PovID': povdb_tbl_trns['PovID'], 'IncmID': incmdb_tbl_trns['IncmID'],
           'EmpID': emplydb_tbl_trns['EmpID'],'PopID': popdb_tbl_trns['PopID']}
# create a df
SEdb_tbl = pd.DataFrame(id_dict)
```



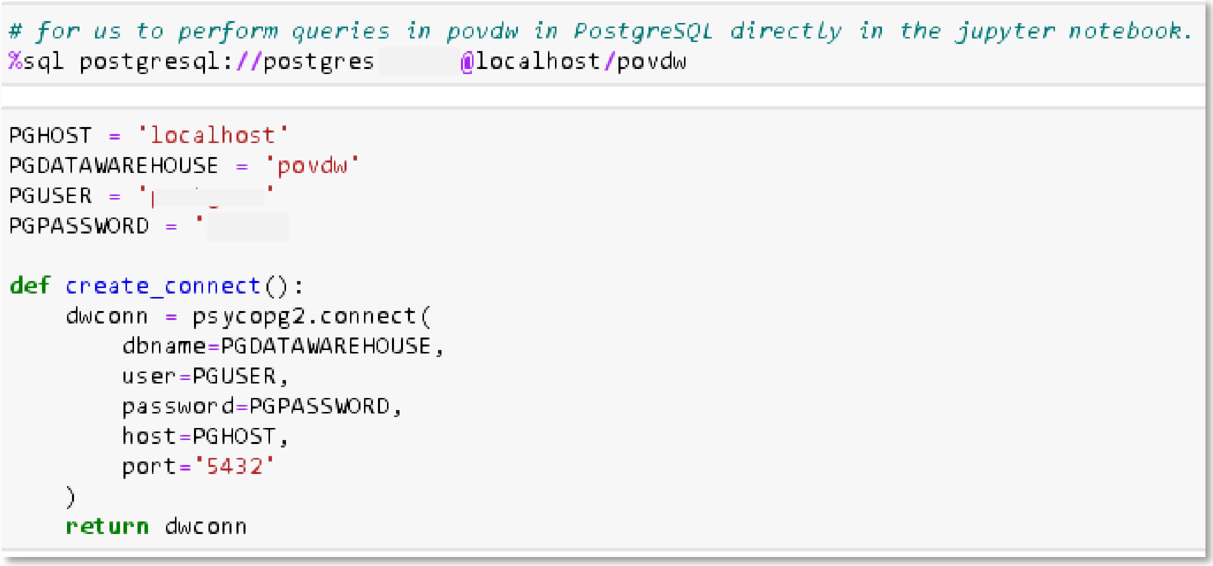
```
# assign the primary key
SEdb_tbl.insert(0, 'SocEcoID', range(1, len(SEdb_tbl)+1))

SEdb_tbl
```

	SocEcoID	YearID	RegID	PovID	IncmID	EmpID	PopID
0	1	1	1	1	1	1	1
1	2	1	2	2	2	2	2

II. 3 LOAD

After the transformation process, will now load the data into the povdw data warehouse.

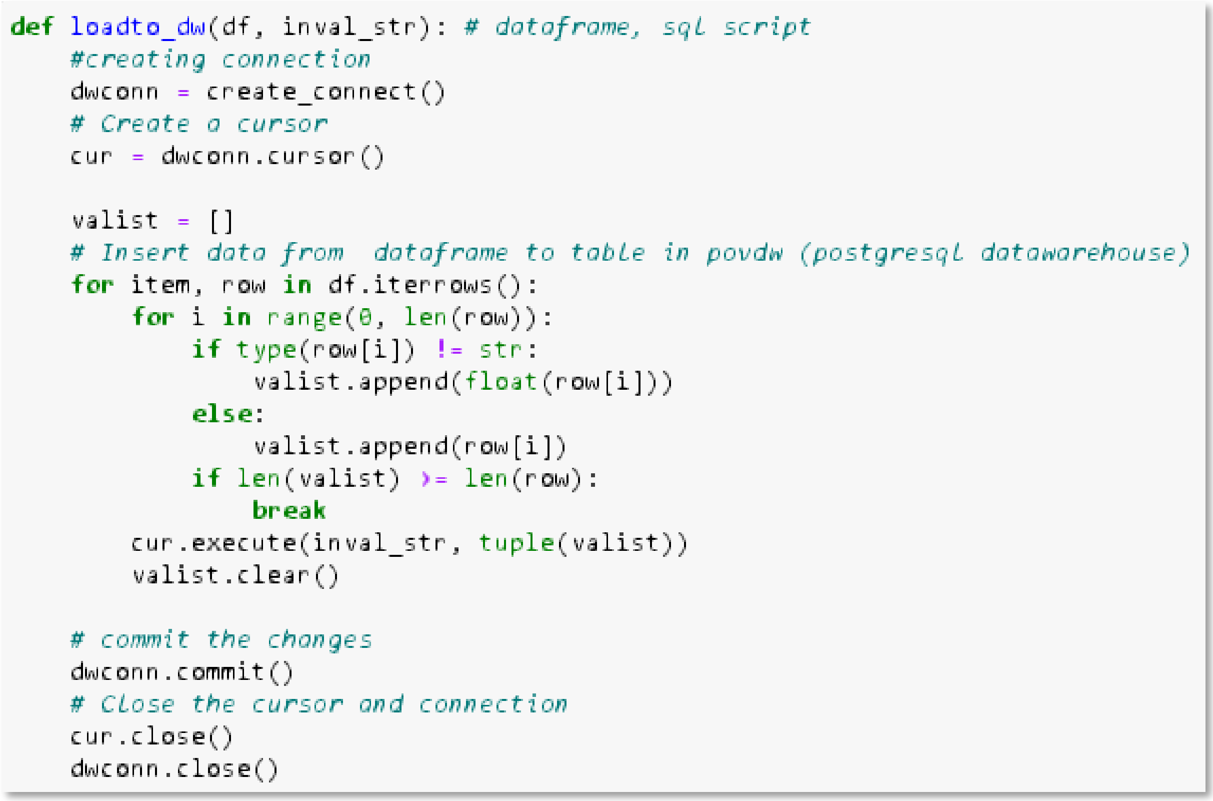
Creating connection to data warehouse.



```
# for us to perform queries in povdw in PostgreSQL directly in the jupyter notebook.
%sql postgresql://postgres[redacted]@localhost/povdw
```

```
PGHOST = 'localhost'
PGDATAWAREHOUSE = 'povdw'
PGUSER = '[redacted]'
PGPASSWORD = '[redacted]

def create_connect():
    dwconn = psycopg2.connect(
        dbname=PGDATAWAREHOUSE,
        user=PGUSER,
        password=PGPASSWORD,
        host=PGHOST,
        port='5432'
    )
    return dwconn
```

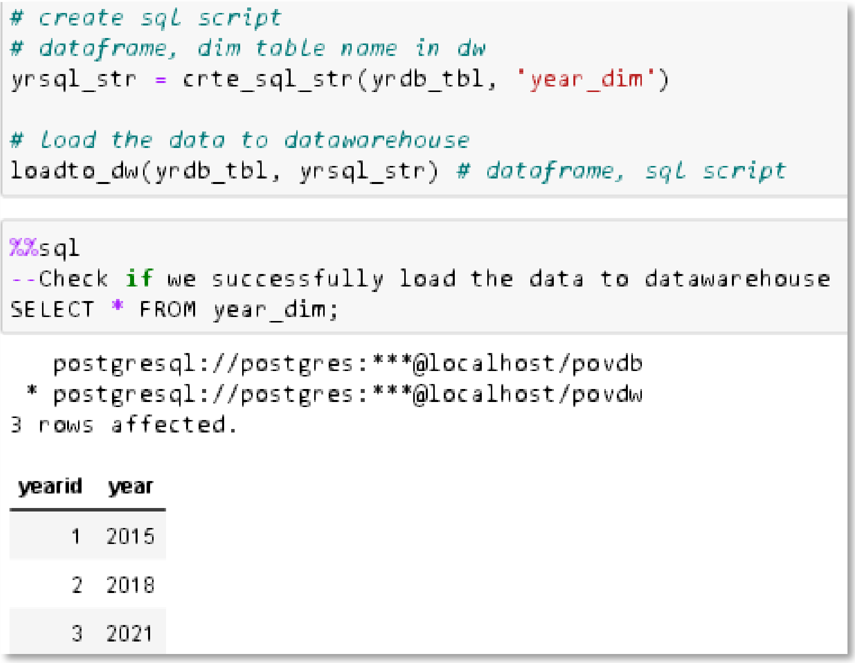
Load data to data warehouse (function)



```
def loadto_dw(df, inval_str): # dataframe, sql script
    #creating connection
    dwconn = create_connect()
    # Create a cursor
    cur = dwconn.cursor()

    valist = []
    # Insert data from  dataframe to table in povdw (postgresql datawarehouse)
    for item, row in df.iterrows():
        for i in range(0, len(row)):
            if type(row[i]) != str:
                valist.append(float(row[i]))
            else:
                valist.append(row[i])
            if len(valist) >= len(row):
                break
        cur.execute(inval_str, tuple(valist))
        valist.clear()

    # commit the changes
    dwconn.commit()
    # Close the cursor and connection
    cur.close()
    dwconn.close()
```

The function above will load the previously transformed data, but before using this function, will need to call again the previously created function named “crte_sql_string” to generate SQL string syntax.

Year_dim



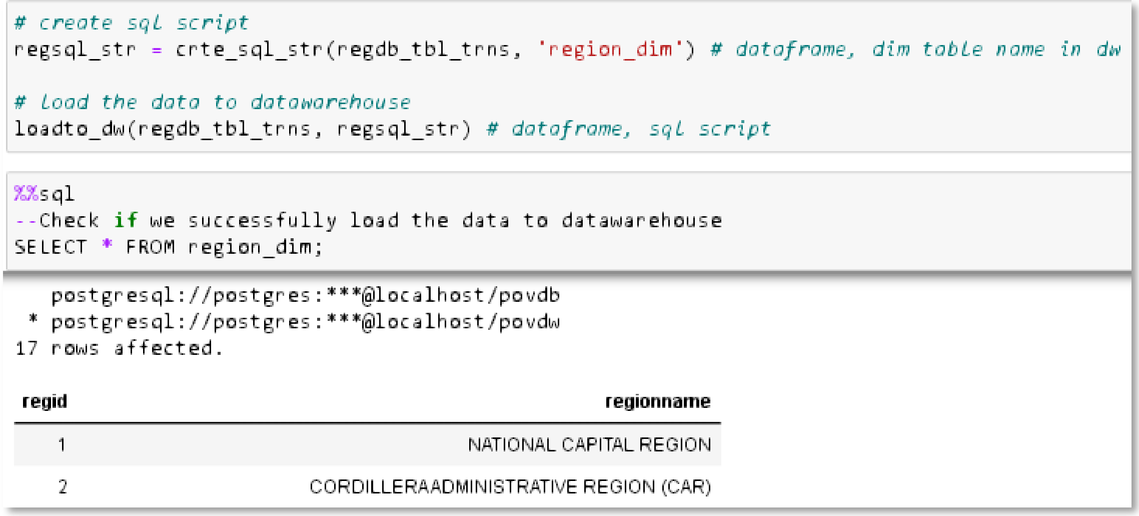
```
# create sql script
# dataframe, dim table name in dw
yrsql_str = crte_sql_str(yrdb_tbl, 'year_dim')

# load the data to datawarehouse
loadto_dw(yrdb_tbl, yrsql_str) # dataframe, sql script
```

```
%%sql
--Check if we successfully load the data to datawarehouse
SELECT * FROM year_dim;
```

```
   postgresql://postgres:***@localhost/povdb
 * postgresql://postgres:***@localhost/povdw
3 rows affected.
```

yearid	year
1	2015
2	2018
3	2021

Region_dim



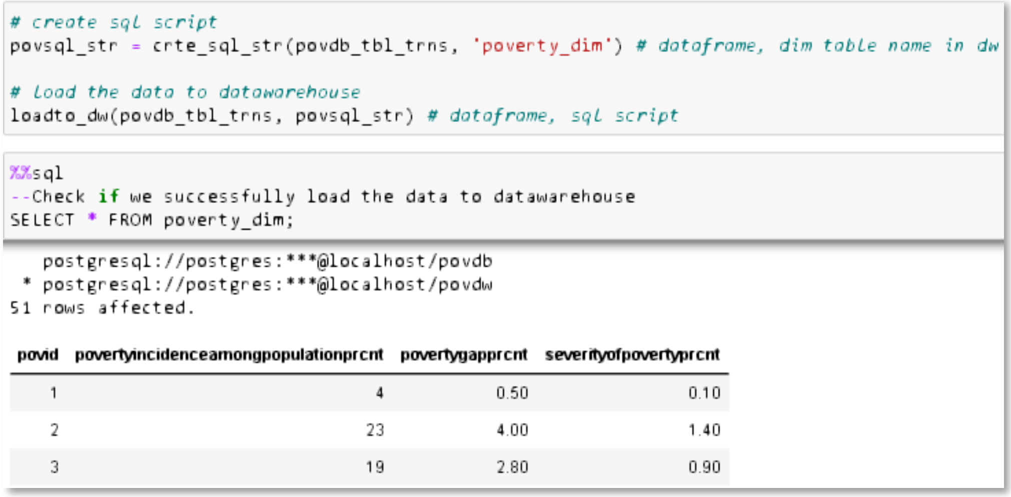
```
# create sql script
regsql_str = crte_sql_str(regdb_tbl_trns, 'region_dim') # dataframe, dim table name in dw

# load the data to datawarehouse
loadto_dw(regdb_tbl_trns, regsql_str) # dataframe, sql script
```

```
%%sql
--Check if we successfully load the data to datawarehouse
SELECT * FROM region_dim;
```

```
   postgresql://postgres:***@localhost/povdb
 * postgresql://postgres:***@localhost/povdw
17 rows affected.
```

regid	regionname
1	NATIONAL CAPITAL REGION
2	CORDILLERAADMINISTRATIVE REGION (CAR)

Poverty_dim



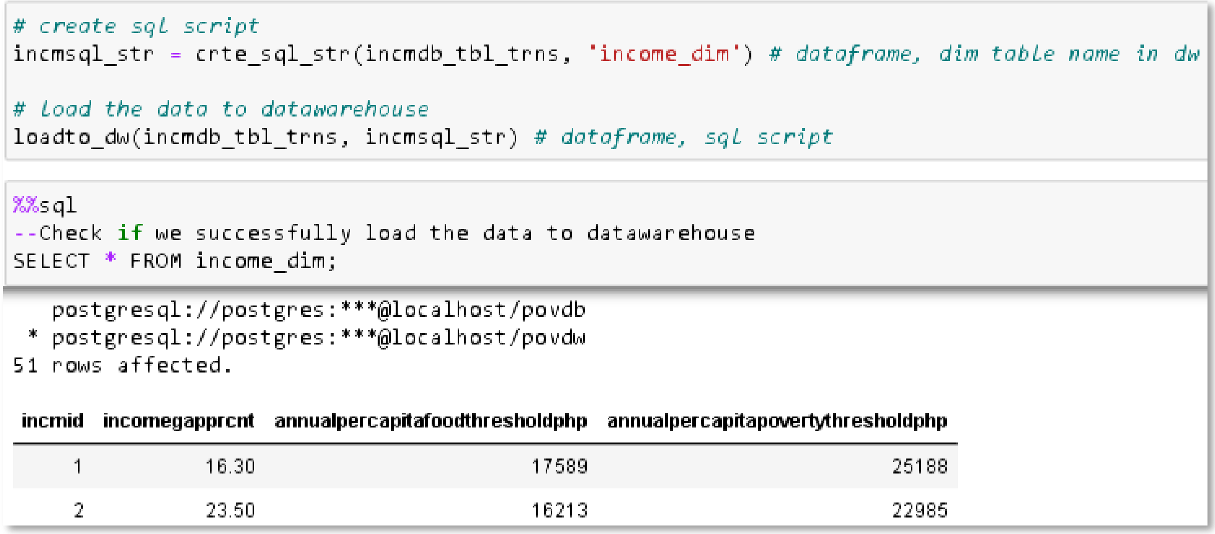
```
# create sql script
povsql_str = crte_sql_str(povdb_tbl_trns, 'poverty_dim') # dataframe, dim table name in dw

# load the data to datawarehouse
loadto_dw(povdb_tbl_trns, povsql_str) # dataframe, sql script
```

```
%%sql
--Check if we successfully load the data to datawarehouse
SELECT * FROM poverty_dim;
```

```
   postgresql://postgres:***@localhost/povdb
 * postgresql://postgres:***@localhost/povdw
51 rows affected.
```

povid	povertyincidenceamongpopulationprcnt	povertygapprcnt	severityofpovertyprcnt
1	4	0.50	0.10
2	23	4.00	1.40
3	19	2.80	0.90

Income_dim



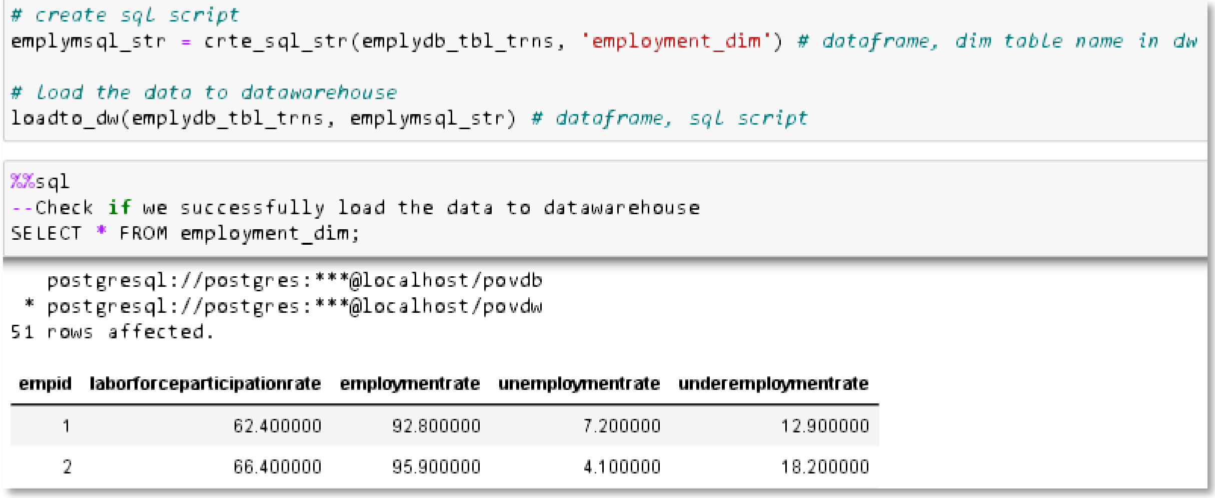
```
# create sql script
incmsql_str = crte_sql_str(incmdb_tbl_trns, 'income_dim') # dataframe, dim table name in dw

# load the data to datawarehouse
loadto_dw(incmdb_tbl_trns, incmsql_str) # dataframe, sql script
```

```
%%sql
--Check if we successfully load the data to datawarehouse
SELECT * FROM income_dim;
```

```
   postgresql://postgres:***@localhost/povdb
 * postgresql://postgres:***@localhost/povdw
51 rows affected.
```

incmid	incomegapprcnt	annualpercapitafoodthresholdphp	annualpercapitapovertythresholdphp
1	16.30	17589	25188
2	23.50	16213	22985

Employment_dim



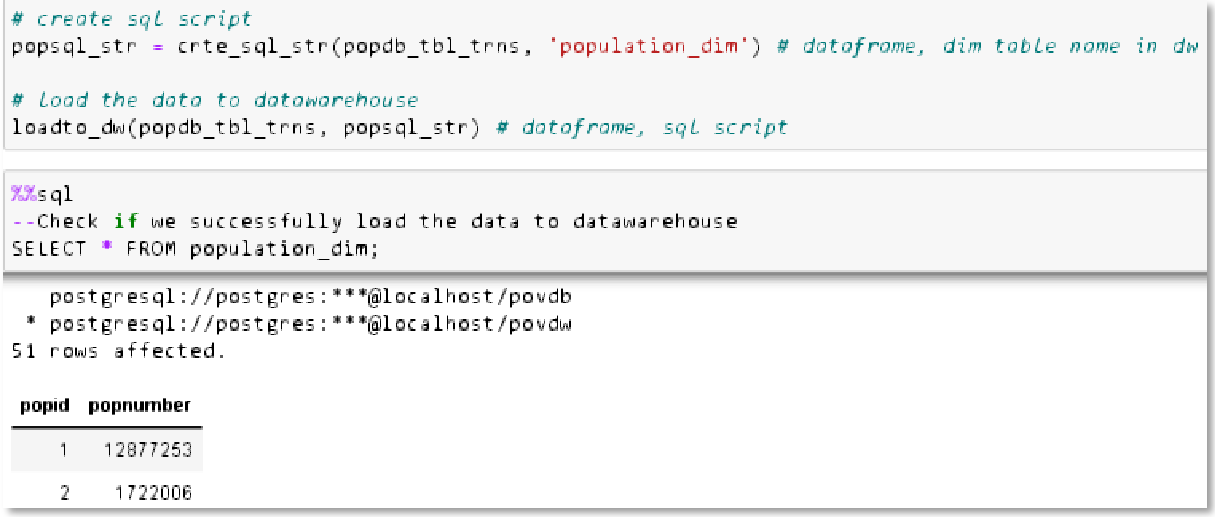
```
# create sql script
emplymsql_str = crte_sql_str(emplydb_tbl_trns, 'employment_dim') # dataframe, dim table name in dw

# load the data to datawarehouse
loadto_dw(emplydb_tbl_trns, emplymsql_str) # dataframe, sql script
```

```
%%sql
--Check if we successfully load the data to datawarehouse
SELECT * FROM employment_dim;
```

```
   postgresql://postgres:***@localhost/povdb
 * postgresql://postgres:***@localhost/povdw
51 rows affected.
```

empid	laborforceparticipationrate	employmentrate	unemploymentrate	underemploymentrate
1	62.400000	92.800000	7.200000	12.900000
2	66.400000	95.900000	4.100000	18.200000

Population_dim



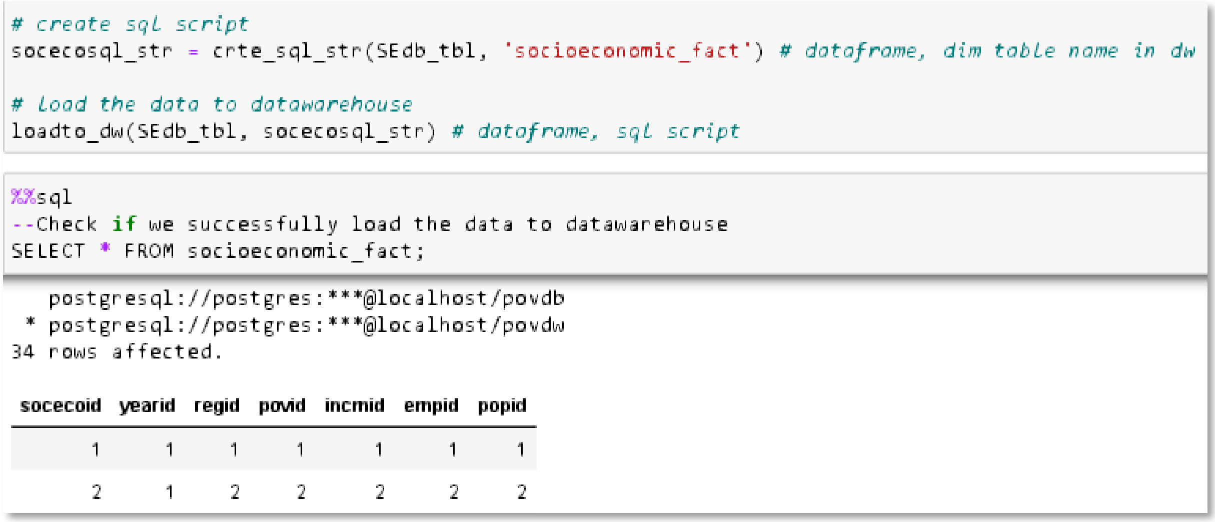
```
# create sql script
popsql_str = crte_sql_str(popdb_tbl_trns, 'population_dim') # dataframe, dim table name in dw

# load the data to datawarehouse
loadto_dw(popdb_tbl_trns, popsql_str) # dataframe, sql script
```

```
%%sql
--Check if we successfully load the data to datawarehouse
SELECT * FROM population_dim;
```

```
   postgresql://postgres:***@localhost/povdb
 * postgresql://postgres:***@localhost/povdw
51 rows affected.
```

popid	popnumber
1	12877253
2	1722006

Socio-Economic_fact



```
# create sql script
socecosql_str = crte_sql_str(SEdb_tbl, 'socioeconomic_fact') # dataframe, dim table name in dw

# Load the data to datawarehouse
loadto_dw(SEdb_tbl, socecosql_str) # dataframe, sql script
```

```
%%sql
--Check if we successfully load the data to datawarehouse
SELECT * FROM socioeconomic_fact;
```

```
   postgresql://postgres:***@localhost/povdb
 * postgresql://postgres:***@localhost/povdw
34 rows affected.
```

socecoid	yearid	regid	povid	incmid	empid	popid
1	1	1	1	1	1	1
2	1	2	2	2	2	2

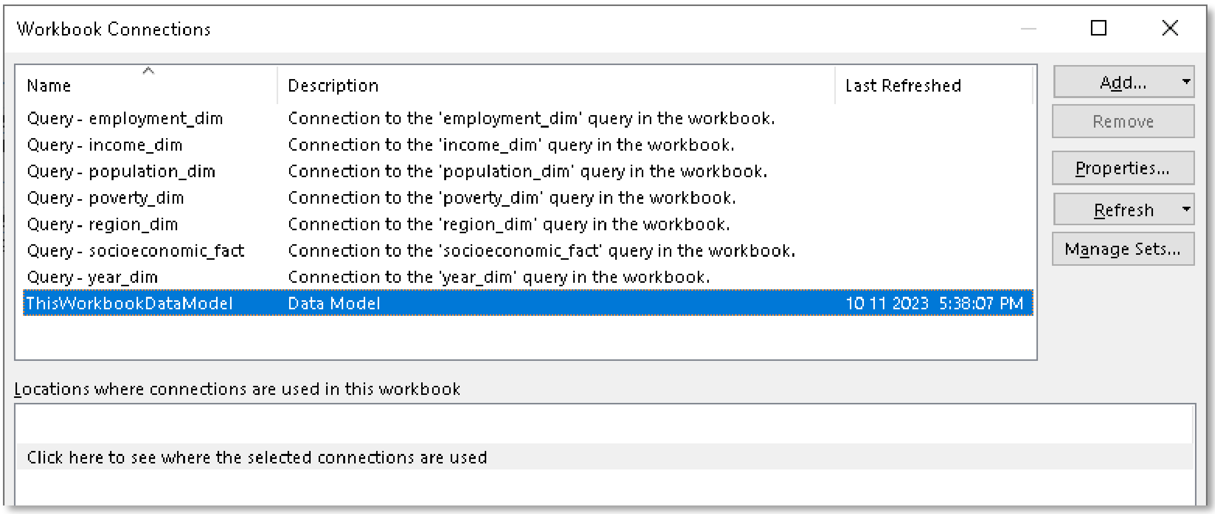
Summary table

Year	Regions	Number of rows	Primary Keys
2015	NCR, CAR, Reg I – XIII, BARMM	17	1 - 17
2018	NCR, CAR, Reg I – XIII, BARMM	17	18 - 34
2021p	NCR, CAR, Reg I – XIII, BARMM	17	35 - 51

At the time of creating those screen captures during both Phase I and Phase II ETL processes, the displayed data is limited to two – three columns only, if you wish to view all the rows of data, please refer to the Jupyter notebook.

Visualization (connecting PostgreSQL to Excel via ODBC)

The image below is the established connection after connecting PostgreSQL povdw data warehouse to Excel via ODBC direct connection.



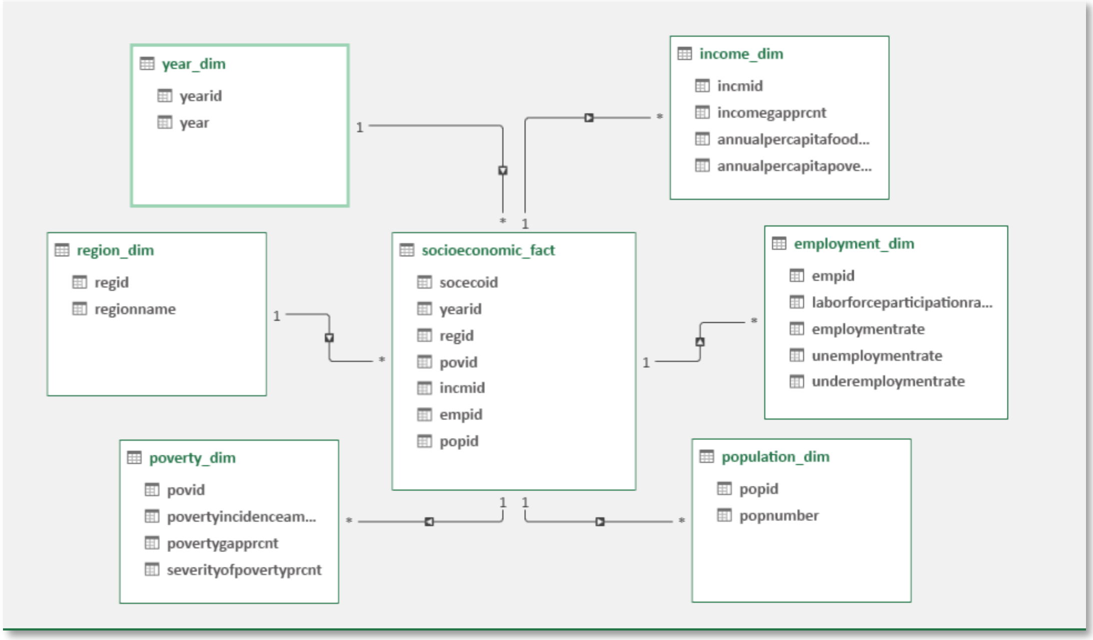
Workbook Connections

Name	Description	Last Refreshed
Query - employment_dim	Connection to the 'employment_dim' query in the workbook.	
Query - income_dim	Connection to the 'income_dim' query in the workbook.	
Query - population_dim	Connection to the 'population_dim' query in the workbook.	
Query - poverty_dim	Connection to the 'poverty_dim' query in the workbook.	
Query - region_dim	Connection to the 'region_dim' query in the workbook.	
Query - socioeconomic_fact	Connection to the 'socioeconomic_fact' query in the workbook.	
Query - year_dim	Connection to the 'year_dim' query in the workbook.	
ThisWorkbookDataModel	Data Model	10 11 2023 5:38:07 PM

Locations where connections are used in this workbook

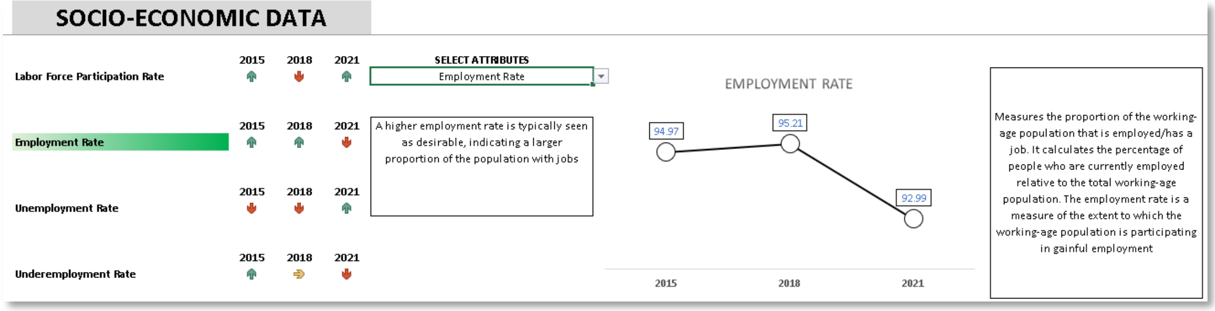
Click here to see where the selected connections are used

Pover Pivot Schema (Excel)



Creating a simple dynamic dashboard in excel to show the Socio-Economic data/status. **(see excel file for clearer view and more in-dept explanation)**

Overall Socio-Economic Status



SOCIO-ECONOMIC DATA

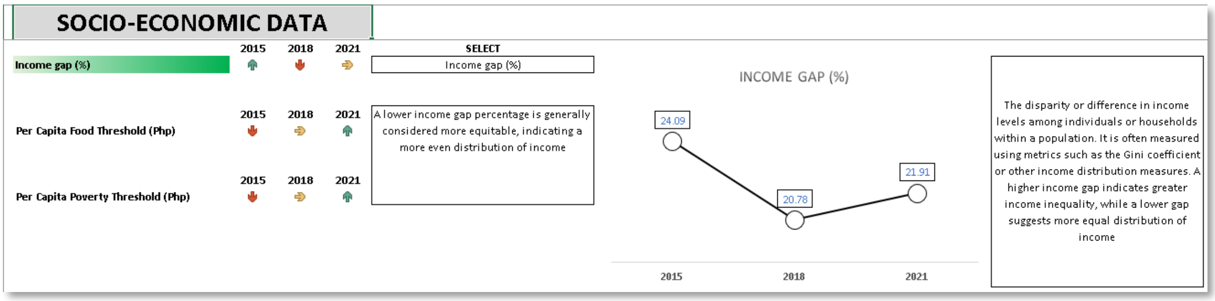
	2015	2018	2021
Labor Force Participation Rate	↑	↓	↑
Employment Rate	↑	↑	↓
Unemployment Rate	↓	↓	↑
Underemployment Rate	↑	→	↓

SELECT ATTRIBUTES

Employment Rate

A higher employment rate is typically seen as desirable, indicating a larger proportion of the population with jobs

Measures the proportion of the working-age population that is employed/has a job. It calculates the percentage of people who are currently employed relative to the total working-age population. The employment rate is a measure of the extent to which the working-age population is participating in gainful employment



SOCIO-ECONOMIC DATA

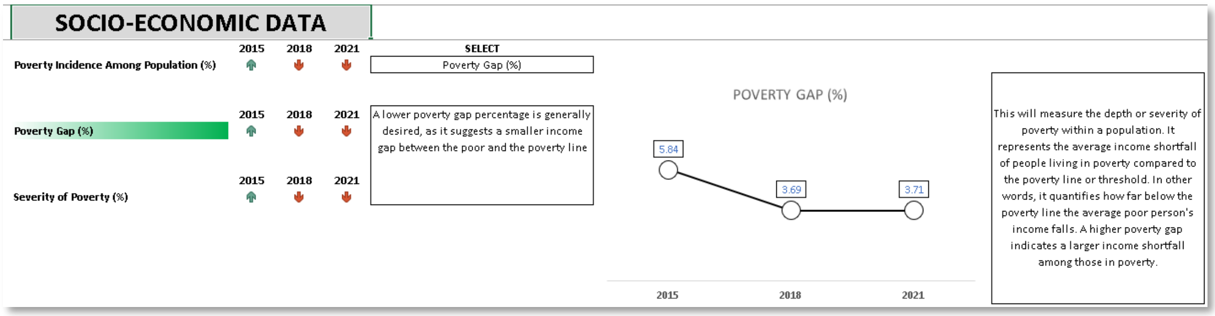
	2015	2018	2021
Income gap (%)	↑	↓	→
Per Capita Food Threshold (Php)	↓	→	↑
Per Capita Poverty Threshold (Php)	↓	→	↑

SELECT

Income gap (%)

A lower income gap percentage is generally considered more equitable, indicating a more even distribution of income

The disparity or difference in income levels among individuals or households within a population. It is often measured using metrics such as the Gini coefficient or other income distribution measures. A higher income gap indicates greater income inequality, while a lower gap suggests more equal distribution of income



SOCIO-ECONOMIC DATA

	2015	2018	2021
Poverty Incidence Among Population (%)	↑	↓	↓
Poverty Gap (%)	↑	↓	↓
Severity of Poverty (%)	↑	↓	↓

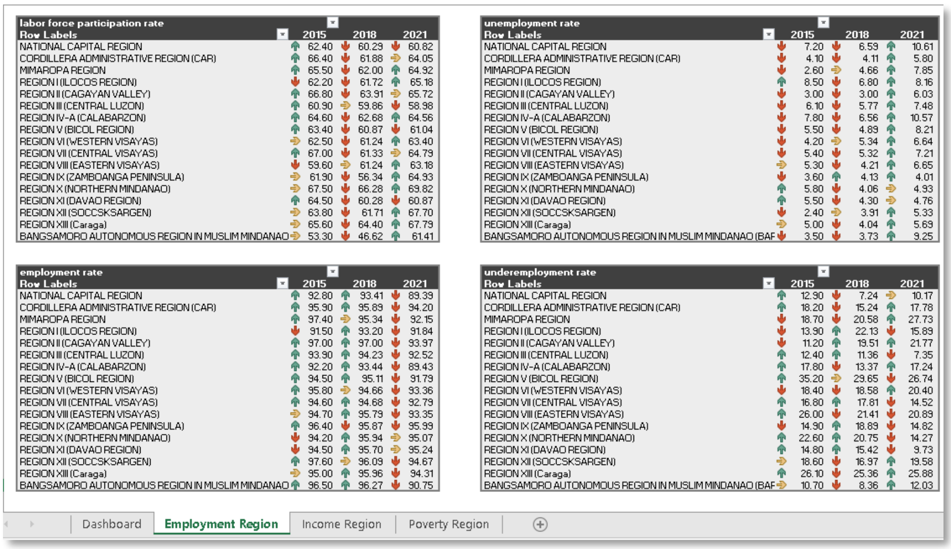
SELECT

Poverty Gap (%)

A lower poverty gap percentage is generally desired, as it suggests a smaller income gap between the poor and the poverty line

This will measure the depth or severity of poverty within a population. It represents the average income shortfall of people living in poverty compared to the poverty line or threshold. In other words, it quantifies how far below the poverty line the average poor person's income falls. A higher poverty gap indicates a larger income shortfall among those in poverty.

Employment by Region



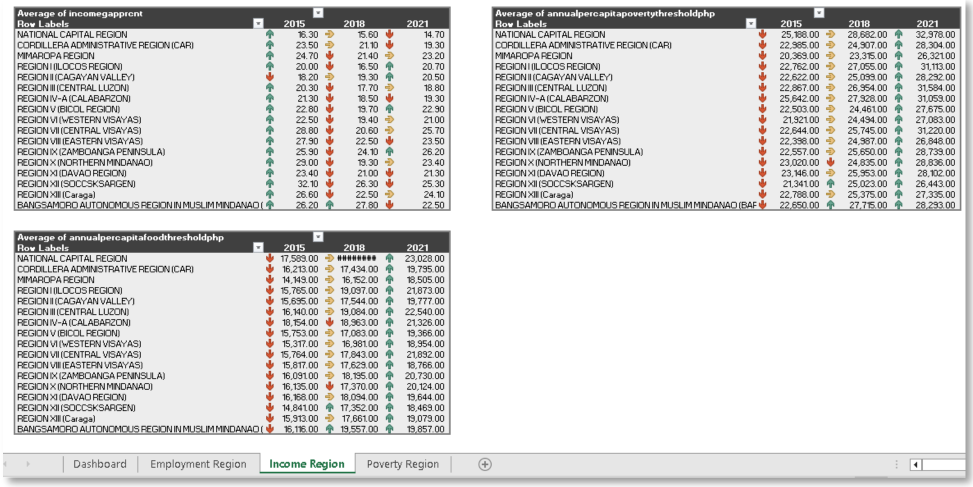
labor force participation rate			
Row Labels	2015	2018	2021
NATIONAL CAPITAL REGION	↑ 62.40	↓ 60.29	↓ 60.82
CORDILLERA ADMINISTRATIVE REGION (CAR)	↑ 66.40	↓ 61.88	→ 64.05
MIMAROPA REGION	↑ 65.50	↓ 62.00	↑ 64.92
REGION I (ILOCOS REGION)	↓ 62.20	↓ 61.72	↑ 65.18
REGION II (CAGAYAN VALLEY)	↑ 66.80	↓ 63.91	→ 65.72
REGION III (CENTRAL LUZON)	↑ 60.90	→ 59.86	↓ 58.98
REGION IV-A (CALABARZON)	↑ 64.60	↓ 62.68	↑ 64.56
REGION V (BICOL REGION)	↑ 63.40	↓ 60.87	↓ 61.04
REGION VI (WESTERN VISAYAS)	→ 62.50	↓ 61.24	↑ 63.40
REGION VII (CENTRAL VISAYAS)	↑ 67.00	↓ 61.33	→ 64.79
REGION VIII (EASTERN VISAYAS)	↓ 59.60	→ 61.24	↑ 63.18
REGION IX (ZAMBOANGA PENINSULA)	→ 61.90	↓ 56.34	↑ 64.93
REGION X (NORTHERN MINDANAO)	→ 67.50	↓ 66.28	↑ 69.82
REGION XI (DAVAO REGION)	↑ 64.50	↓ 60.28	↓ 60.87
REGION XII (SOCCSKSARGEN)	→ 63.80	↓ 61.71	↑ 67.70
REGION XIII (Caraga)	→ 65.60	↓ 64.40	↑ 67.79
BANGSAMORO AUTONOMOUS REGION IN MUSLIM MINDANAO	→ 53.30	↓ 46.62	↑ 61.41

unemployment rate			
Row Labels	2015	2018	2021
NATIONAL CAPITAL REGION	↓ 7.20	↓ 6.59	↑ 10.61
CORDILLERA ADMINISTRATIVE REGION (CAR)	↓ 4.10	↓ 4.11	↑ 5.80
MIMAROPA REGION	↓ 2.60	→ 4.66	↑ 7.85
REGION I (ILOCOS REGION)	↑ 8.50	↓ 6.80	↑ 8.16
REGION II (CAGAYAN VALLEY)	↓ 3.00	↓ 3.00	↑ 6.03
REGION III (CENTRAL LUZON)	↓ 6.10	↓ 5.77	↑ 7.48
REGION IV-A (CALABARZON)	↓ 7.80	↓ 6.56	↑ 10.57
REGION V (BICOL REGION)	↓ 5.50	↓ 4.89	↑ 8.21
REGION VI (WESTERN VISAYAS)	↓ 4.20	→ 5.34	↑ 6.64
REGION VII (CENTRAL VISAYAS)	↓ 5.40	↓ 5.32	↑ 7.21
REGION VIII (EASTERN VISAYAS)	→ 5.30	↓ 4.21	↑ 6.65
REGION IX (ZAMBOANGA PENINSULA)	↓ 3.60	↑ 4.13	↑ 4.01
REGION X (NORTHERN MINDANAO)	↑ 5.80	↓ 4.06	→ 4.93
REGION XI (DAVAO REGION)	↑ 5.50	↓ 4.30	→ 4.76
REGION XII (SOCCSKSARGEN)	↓ 2.40	→ 3.91	↑ 5.33
REGION XIII (Caraga)	→ 5.00	↓ 4.04	↑ 5.69
BANGSAMORO AUTONOMOUS REGION IN MUSLIM MINDANAO (BAF	↓ 3.50	↓ 3.73	↑ 9.25

employment rate			
Row Labels	2015	2018	2021
NATIONAL CAPITAL REGION	↑ 92.80	↑ 93.41	↓ 89.39
CORDILLERA ADMINISTRATIVE REGION (CAR)	↑ 95.90	↑ 95.89	↓ 94.20
MIMAROPA REGION	↑ 97.40	→ 95.34	↓ 92.15
REGION I (ILOCOS REGION)	↓ 91.50	↑ 93.20	↓ 91.84
REGION II (CAGAYAN VALLEY)	↑ 97.00	↑ 97.00	↓ 93.97
REGION III (CENTRAL LUZON)	↑ 93.90	↑ 94.23	↓ 92.52
REGION IV-A (CALABARZON)	↑ 92.20	↑ 93.44	↓ 89.43
REGION V (BICOL REGION)	↑ 94.50	↑ 95.11	↓ 91.79
REGION VI (WESTERN VISAYAS)	↑ 95.80	→ 94.66	↓ 93.36
REGION VII (CENTRAL VISAYAS)	↑ 94.60	↑ 94.68	↓ 92.79
REGION VIII (EASTERN VISAYAS)	→ 94.70	↑ 95.79	↓ 93.35
REGION IX (ZAMBOANGA PENINSULA)	↑ 96.40	↓ 95.87	↓ 95.99
REGION X (NORTHERN MINDANAO)	↓ 94.20	↑ 95.94	→ 95.07
REGION XI (DAVAO REGION)	↓ 94.50	↑ 95.70	→ 95.24
REGION XII (SOCCSKSARGEN)	↑ 97.60	→ 96.09	↓ 94.67
REGION XIII (Caraga)	→ 95.00	↑ 95.96	↓ 94.31
BANGSAMORO AUTONOMOUS REGION IN MUSLIM MINDANAO	↑ 96.50	↑ 96.27	↓ 90.75

underemployment rate			
Row Labels	2015	2018	2021
NATIONAL CAPITAL REGION	↑ 12.90	↓ 7.24	→ 10.17
CORDILLERA ADMINISTRATIVE REGION (CAR)	↑ 18.20	↓ 15.24	↑ 17.78
MIMAROPA REGION	↓ 18.70	↓ 20.58	↑ 27.73
REGION I (ILOCOS REGION)	↓ 13.90	↑ 22.13	↓ 15.89
REGION II (CAGAYAN VALLEY)	↓ 11.20	↑ 19.51	↑ 21.77
REGION III (CENTRAL LUZON)	↑ 12.40	↑ 11.36	↓ 7.35
REGION IV-A (CALABARZON)	↑ 17.80	↓ 13.37	↑ 17.24
REGION V (BICOL REGION)	↑ 35.20	→ 29.65	↓ 26.74
REGION VI (WESTERN VISAYAS)	↓ 18.40	↓ 18.58	↑ 20.40
REGION VII (CENTRAL VISAYAS)	↑ 16.80	↑ 17.81	↓ 14.52
REGION VIII (EASTERN VISAYAS)	↑ 26.00	↓ 21.41	↓ 20.89
REGION IX (ZAMBOANGA PENINSULA)	↓ 14.90	↑ 18.89	↓ 14.82
REGION X (NORTHERN MINDANAO)	↑ 22.60	↑ 20.75	↓ 14.27
REGION XI (DAVAO REGION)	↑ 14.80	↑ 15.42	↓ 9.73
REGION XII (SOCCSKSARGEN)	→ 18.60	↓ 16.97	↑ 19.58
REGION XIII (Caraga)	↑ 26.10	↓ 25.36	↑ 25.88
BANGSAMORO AUTONOMOUS REGION IN MUSLIM MINDANAO (BAF	→ 10.70	↓ 8.36	↑ 12.03

Income by Region

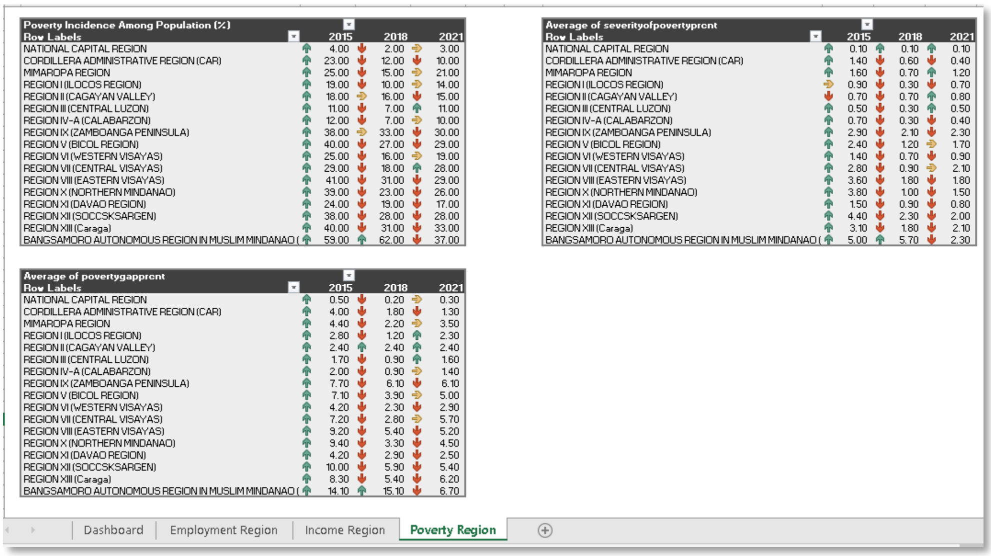


Average of incomegapprcnt			
Row Labels	2015	2018	2021
NATIONAL CAPITAL REGION	↑ 16.30	→ 15.60	↓ 14.70
CORDILLERA ADMINISTRATIVE REGION (CAR)	↑ 23.50	→ 21.10	↓ 19.30
MIMAROPA REGION	↑ 24.70	↓ 21.40	→ 23.20
REGION I (ILOCOS REGION)	↑ 20.00	↓ 16.50	↑ 20.70
REGION II (CAGAYAN VALLEY)	↓ 18.20	→ 19.30	↑ 20.50
REGION III (CENTRAL LUZON)	↑ 20.30	↓ 17.70	→ 18.80
REGION IV-A (CALABARZON)	↑ 21.30	↓ 18.50	↓ 19.30
REGION V (BICOL REGION)	↑ 22.80	↓ 19.70	↑ 22.90
REGION VI (WESTERN VISAYAS)	↑ 22.50	↓ 19.40	→ 21.00
REGION VII (CENTRAL VISAYAS)	↑ 28.80	↓ 20.60	→ 25.70
REGION VIII (EASTERN VISAYAS)	↑ 27.90	↓ 22.50	↓ 23.50
REGION IX (ZAMBOANGA PENINSULA)	↑ 25.90	↓ 24.10	↑ 26.20
REGION X (NORTHERN MINDANAO)	↑ 29.00	↓ 19.30	→ 23.40
REGION XI (DAVAO REGION)	↑ 23.40	↓ 21.00	↓ 21.30
REGION XII (SOCCSKSARGEN)	↑ 32.10	↓ 26.30	↓ 25.30
REGION XIII (Caraga)	↑ 26.60	↓ 22.50	→ 24.10
BANGSAMORO AUTONOMOUS REGION IN MUSLIM MINDANAO (	↑ 26.20	↑ 27.80	↓ 22.50

Average of annualpercapitapovertythresholdphp			
Row Labels	2015	2018	2021
NATIONAL CAPITAL REGION	↓ 25,188.00	→ 28,682.00	↑ 32,978.00
CORDILLERA ADMINISTRATIVE REGION (CAR)	↓ 22,985.00	→ 24,907.00	↑ 28,304.00
MIMAROPA REGION	↓ 20,369.00	→ 23,315.00	↑ 26,321.00
REGION I (ILOCOS REGION)	↓ 22,762.00	→ 27,055.00	↑ 31,113.00
REGION II (CAGAYAN VALLEY)	↓ 22,622.00	→ 25,099.00	↑ 28,292.00
REGION III (CENTRAL LUZON)	↓ 22,867.00	→ 26,954.00	↑ 31,584.00
REGION IV-A (CALABARZON)	↓ 25,642.00	→ 27,928.00	↑ 31,059.00
REGION V (BICOL REGION)	↓ 22,503.00	→ 24,461.00	↑ 27,675.00
REGION VI (WESTERN VISAYAS)	↓ 21,921.00	→ 24,494.00	↑ 27,083.00
REGION VII (CENTRAL VISAYAS)	↓ 22,644.00	→ 25,745.00	↑ 31,220.00
REGION VIII (EASTERN VISAYAS)	↓ 22,398.00	→ 24,987.00	↑ 26,848.00
REGION IX (ZAMBOANGA PENINSULA)	↓ 22,557.00	→ 25,650.00	↑ 28,739.00
REGION X (NORTHERN MINDANAO)	↓ 23,020.00	↓ 24,835.00	↑ 28,836.00
REGION XI (DAVAO REGION)	↓ 23,146.00	→ 25,953.00	↑ 28,102.00
REGION XII (SOCCSKSARGEN)	↓ 21,341.00	↑ 25,023.00	↑ 26,443.00
REGION XIII (Caraga)	↓ 22,788.00	→ 25,375.00	↑ 27,335.00
BANGSAMORO AUTONOMOUS REGION IN MUSLIM MINDANAO (BAF	↓ 22,650.00	↑ 27,715.00	↑ 28,293.00

Average of annualpercapitafoodthresholdphp			
Row Labels	2015	2018	2021
NATIONAL CAPITAL REGION	↓ 17,589.00	→ ########	↑ 23,028.00
CORDILLERA ADMINISTRATIVE REGION (CAR)	↓ 16,213.00	→ 17,434.00	↑ 19,795.00
MIMAROPA REGION	↓ 14,149.00	→ 16,152.00	↑ 18,505.00
REGION I (ILOCOS REGION)	↓ 15,765.00	→ 19,097.00	↑ 21,873.00
REGION II (CAGAYAN VALLEY)	↓ 15,695.00	→ 17,544.00	↑ 19,777.00
REGION III (CENTRAL LUZON)	↓ 16,140.00	→ 19,084.00	↑ 22,540.00
REGION IV-A (CALABARZON)	↓ 18,154.00	↓ 18,963.00	↑ 21,326.00
REGION V (BICOL REGION)	↓ 15,753.00	→ 17,083.00	↑ 19,366.00
REGION VI (WESTERN VISAYAS)	↓ 15,317.00	→ 16,981.00	↑ 18,954.00
REGION VII (CENTRAL VISAYAS)	↓ 15,764.00	→ 17,843.00	↑ 21,892.00
REGION VIII (EASTERN VISAYAS)	↓ 15,817.00	→ 17,629.00	↑ 18,766.00
REGION IX (ZAMBOANGA PENINSULA)	↓ 16,091.00	→ 18,195.00	↑ 20,730.00
REGION X (NORTHERN MINDANAO)	↓ 16,135.00	↓ 17,370.00	↑ 20,124.00
REGION XI (DAVAO REGION)	↓ 16,168.00	→ 18,094.00	↑ 19,644.00
REGION XII (SOCCSKSARGEN)	↓ 14,841.00	↑ 17,352.00	↑ 18,469.00
REGION XIII (Caraga)	↓ 15,913.00	→ 17,661.00	↑ 19,079.00
BANGSAMORO AUTONOMOUS REGION IN MUSLIM MINDANAO (	↓ 16,116.00	↑ 19,557.00	↑ 19,857.00

Poverty by Region



Poverty Incidence Among Population (%)			
Row Labels	2015	2018	2021
NATIONAL CAPITAL REGION	↑ 4.00	↓ 2.00	→ 3.00
CORDILLERA ADMINISTRATIVE REGION (CAR)	↑ 23.00	↓ 12.00	↓ 10.00
MIMAROPA REGION	↑ 25.00	↓ 15.00	→ 21.00
REGION I (ILOCOS REGION)	↑ 19.00	↓ 10.00	→ 14.00
REGION II (CAGAYAN VALLEY)	↑ 18.00	→ 16.00	↓ 15.00
REGION III (CENTRAL LUZON)	↑ 11.00	↓ 7.00	↑ 11.00
REGION IV-A (CALABARZON)	↑ 12.00	↓ 7.00	→ 10.00
REGION IX (ZAMBOANGA PENINSULA)	↑ 38.00	→ 33.00	↓ 30.00
REGION V (BICOL REGION)	↑ 40.00	↓ 27.00	↓ 29.00
REGION VI (WESTERN VISAYAS)	↑ 25.00	↓ 16.00	→ 19.00
REGION VII (CENTRAL VISAYAS)	↑ 29.00	↓ 18.00	↑ 28.00
REGION VIII (EASTERN VISAYAS)	↑ 41.00	↓ 31.00	↓ 29.00
REGION X (NORTHERN MINDANAO)	↑ 39.00	↓ 23.00	↓ 26.00
REGION XI (DAVAO REGION)	↑ 24.00	↓ 19.00	↓ 17.00
REGION XII (SOCCSKSARGEN)	↑ 38.00	↓ 28.00	↓ 28.00
REGION XIII (Caraga)	↑ 40.00	↓ 31.00	↓ 33.00
BANGSAMORO AUTONOMOUS REGION IN MUSLIM MINDANAO (	↑ 59.00	↑ 62.00	↓ 37.00

Average of severityofpovertyprcnt			
Row Labels	2015	2018	2021
NATIONAL CAPITAL REGION	↑ 0.10	↑ 0.10	↑ 0.10
CORDILLERA ADMINISTRATIVE REGION (CAR)	↑ 1.40	↓ 0.60	↓ 0.40
MIMAROPA REGION	↑ 1.60	↓ 0.70	↑ 1.20
REGION I (ILOCOS REGION)	→ 0.90	↓ 0.30	↓ 0.70
REGION II (CAGAYAN VALLEY)	↓ 0.70	↓ 0.70	↑ 0.80
REGION III (CENTRAL LUZON)	↑ 0.50	↓ 0.30	↑ 0.50
REGION IV-A (CALABARZON)	↑ 0.70	↓ 0.30	↓ 0.40
REGION IX (ZAMBOANGA PENINSULA)	↑ 2.90	↓ 2.10	↓ 2.30
REGION V (BICOL REGION)	↑ 2.40	↓ 1.20	→ 1.70
REGION VI (WESTERN VISAYAS)	↑ 1.40	↓ 0.70	↓ 0.90
REGION VII (CENTRAL VISAYAS)	↑ 2.80	↓ 0.90	→ 2.10
REGION VIII (EASTERN VISAYAS)	↑ 3.60	↓ 1.80	↓ 1.80
REGION X (NORTHERN MINDANAO)	↑ 3.80	↓ 1.00	↓ 1.50
REGION XI (DAVAO REGION)	↑ 1.50	↓ 0.90	↓ 0.80
REGION XII (SOCCSKSARGEN)	↑ 4.40	↓ 2.30	↓ 2.00
REGION XIII (Caraga)	↑ 3.10	↓ 1.80	↓ 2.10
BANGSAMORO AUTONOMOUS REGION IN MUSLIM MINDANAO (	↑ 5.00	↑ 5.70	↓ 2.30

Average of povertygapprcnt			
Row Labels	2015	2018	2021
NATIONAL CAPITAL REGION	↑ 0.50	↓ 0.20	→ 0.30
CORDILLERA ADMINISTRATIVE REGION (CAR)	↑ 4.00	↓ 1.80	↓ 1.30
MIMAROPA REGION	↑ 4.40	↓ 2.20	→ 3.50
REGION I (ILOCOS REGION)	↑ 2.80	↓ 1.20	↑ 2.30
REGION II (CAGAYAN VALLEY)	↑ 2.40	↑ 2.40	↑ 2.40
REGION III (CENTRAL LUZON)	↑ 1.70	↓ 0.90	↑ 1.60
REGION IV-A (CALABARZON)	↑ 2.00	↓ 0.90	→ 1.40
REGION IX (ZAMBOANGA PENINSULA)	↑ 7.70	↓ 6.10	↓ 6.10
REGION V (BICOL REGION)	↑ 7.10	↓ 3.90	→ 5.00
REGION VI (WESTERN VISAYAS)	↑ 4.20	↓ 2.30	↓ 2.90
REGION VII (CENTRAL VISAYAS)	↑ 7.20	↓ 2.80	→ 5.70
REGION VIII (EASTERN VISAYAS)	↑ 9.20	↓ 5.40	↓ 5.20
REGION X (NORTHERN MINDANAO)	↑ 9.40	↓ 3.30	↓ 4.50
REGION XI (DAVAO REGION)	↑ 4.20	↓ 2.90	↓ 2.50
REGION XII (SOCCSKSARGEN)	↑ 10.00	↓ 5.90	↓ 5.40
REGION XIII (Caraga)	↑ 8.30	↓ 5.40	↓ 6.20
BANGSAMORO AUTONOMOUS REGION IN MUSLIM MINDANAO (	↑ 14.10	↑ 15.10	↓ 6.70

D. Conclusion

Based on the table below, the government should focus on employment and, more importantly, on income attributes, namely 'per capita food threshold' and 'per capita poverty threshold' solutions, as they have been in a negative state since 2018.

Indicator	Attributes	2015	2018	2021	Conclusion for recent year
Employment	Labor Force Participation Rate	+	-	+	Maintain
	Employment Rate	+	+	-	The Government should focus on
	Unemployment Rate	+	+	-	The Government should focus on
	Employment Rate	-	+s	+	Maintain
Income	Income gap (%)	-	+	+s	For Improvement
	Per Capita Food Threshold (Php)	+	-s	-	The Government should focus on
	Per Capita Poverty Threshold (Php)	+	-s	-	The Government should focus on
Poverty	Poverty Incidence Among Population (%)	-	+	+	Maintain
	Poverty Gap (%)	-	+	+	Maintain
	Severity of Poverty (%)	-	+	+	Maintain

(+) in positive state, the government should maintain this.

(-) in negative state, the government should take some intervention or action to solve the problem.

(S) slightly towards negative (-s) or positive (+s) state.

E. References

Kraak, M., Ricker, B., & Engelhardt, Y. (2018). Abstract. Challenges of Mapping Sustainable Development Goals Indicators Data. ISPRS International Journal of Geo-Information, 7(12), 482. https://doi.org/10.3390/ijgi7120482

Annual Labor and Employment Status https://www.psa.gov.ph/content/2018-annual-labor-and-employment-status

Annual Per Capita Food Threshold and Subsistence Incidence Among Population https://openstat.psa.gov.ph/PXWeb/pxweb/en/DB/DB__1E__FY/0071E3DF040.px/?rxid=ca0a2b00-1d5a-412d-8a08-be01e009d80d.

Annual Per Capita Poverty Threshold and Poverty Incidence Among Population https://openstat.psa.gov.ph/PXWeb/pxweb/en/DB/DB__1E__FY/0031E3DF020.px/?rxid=aa90cec7-9eb7-43b8-b32c-0a563c706645

Updated Projected Mid-Year Population for the Philippines Based on the 2015 POPCEN Results: 2020-2025 https://psa.gov.ph/statistics/census/projected-population

2021 Annual Labor Market Statistics (Preliminary Results) https://psa.gov.ph/content/2021-annual-labor-market-statistics-preliminary-results

2015 Annual Labor Survey https://psa.gov.ph/sites/default/files/iesd/2022-12/TABLE%25204%2520Total%2520Population%252015%2520Years%2520Old%2520and%2520Over%2520and%2520Rates%2520of%2520Labor%2520Force%2520Participation%252C%2520Employment%2520Unemployment%2520%2520and%2520Underemployment%252C%2520by%2520Region%2520October%25202015.pdf

Highlights of the Philippine Population 2020 Census of Population and Housing (2020 CPH) https://psa.gov.ph/content/highlights-philippine-population-2020-census-population-and-housing-2020-cph