Electric Fuel Generation and Consumption

Milestone: Report Submission

GROUP 3

Group Members:
Tomoki Kyotani (002986248)
Sai Bhavana Atluri (001594085)
Jignasuben Rohitbhai Vekariya (002981797)
Bhavya Maheshwari (002116459)

DAMG 7290: Data Warehousing and Business Intelligence
Spring 2022, Northeastern University

Instructor:
Prof. Vincent Lattuada

Table of Contents:

Introduction:

Nowadays the population is increasing day by day, use of energy sources and pollution is also increasing rapidly which affects weather as well. We decided to build a warehouse with respective fields for more effective analysis on how energy source generation and consumption is affected by population, pollution, and weather in the U.S.

For our project, we selected four datasets named Electric power data, Population data, Pollution data, and Weather data from four different sources. The Electric Power dataset contains details of electricity production in the entire U.S. from every energy generation plant by month and having data up to 2021. The air pollution dataset contains data of different pollutants like Carbon Monoxide, Nitrogen Dioxide etc. being released in the air in the U.S. The population dataset contains the population estimates of the citizens and immigrants in the U.S. The weather dataset has the data about average, minimum and maximum temperature of everyday of 2015 for every state in the U.S.

By aggregating and joining these 4 datasets by state and year variables, we're trying to analyze how each of these factors affect the other.

Software Requirement:

We are planning to implement our project on:

- SSIS for loading and transforming data into Staging and Data Warehouse (SQL Server)
- SSAS for building an OLAP Cube
- Tableau for visualizations

Data Sources:

1. *ELECTRIC POWER DATA (2001-2021)*:

   Data source (Excel): https://www.eia.gov/electricity/data/eia923/
   File Name: 2010: EIA-923, 2011: EIA-923, 2012: EIA-923, 2013: EIA-923, 2014: EIA-923, 2015: EIA-923 and 2016: EIA-923
   The whole dataset consists of a few sheets which have Electricity Generation & Fuel Data, Stock Data at state & census region levels, Boiler Fuel data, Generator data, Fuel Receipts data. Stock data have coal, petroleum liquids and petroleum coke stocks.
   Data is collected from every plant from each state in the United States with their respective plant operator name, sector name, fuel types and units label. Some absent values were stored using statistical regression techniques.

Column Name	**Data Type**	**Description**
Facility ID	integer	EIA plant identification number.
Facility Name	varchar (100)	Facility Name
State	varchar (2)	State the facility is located in. (e.g. AL)
Operator ID	varchar (50)	The EIA operator identification number
Operator Name	varchar (100)	The name of the entity which operates the plant.
NAICS Code	varchar (10)	The facility's North American Industrial Classification System (NAICS) code.
Reported Fuel Type Code	varchar (30)	The fuel code reported to EIA.
Net Generation (megawatt-hours)	float	Net electricity generation, year to date in megawatt-hours (MWh)
Year	varchar (4)	Year the data is from

2. *AIR POLLUTION (2000-2016)*:

Data Source (csv): https://www.kaggle.com/datasets/sogun3/uspollution
Filename: pollution_us_2000_2016
This dataset is about air pollution in the United States. This data is scraped from the United States Environmental Protection Agency (link) into kaggle. This dataset contains 28 columns and has the information regarding the location i.e., the state, county,city and address, date of observation, the pollutants i.e., Nitrogen Dioxide (NO_2), Carbon Monoxide (CO), Sulfur Dioxide (SO_2), and Ozone (O_3) each of which have 5 specific columns indicating units, mean, first max value, first max hour and Air Quality Index (AQI). The data is available from the year 2000 to 2016.

Column Name	**Data Type**	**Description**
State	varchar (30)	State in which the air pollution is recorded. (e.g. Arizona)
Date Local	date	Date on which the air pollution is recorded.
NO2 Mean	float	Average of the Nitrogen Dioxide values logged.
NO2 Air Quality Index	float	Air Quality Index (AQI) for Nitrogen Dioxide.
O3 Mean	float	Average of the Ozone values logged.
O3 Air Quality Index	float	Air Quality Index (AQI) of Ozone
SO2 Mean	float	Average of the Sulfur Dioxide values logged.
SO2 Air Quality Index	float	Air Quality Index (AQI) of Sulfur Dioxide
CO Mean	float	Average of the Carbon Monoxide values logged.
CO Air Quality Index	float	Air Quality Index (AQI) of Carbon Monoxide

3. *POPULATION (2010-2019)*:

   Data Source (csv):
   https://www.census.gov/data/datasets/time-series/demo/popest/2010s-state-total.html#par_textimage_500989927
   Filename: Population, Population Change, and Estimated Components of Population Change: April 1, 2010 to July 1, 2019 (NST-EST2019-alldata)
   This dataset contains various population data for each state. We are interested in the columns such as NAME (state name), POPESTIMATE201x (resident total population estimate), NPOPCHG_201x (numeric change in resident total population), and NETMIG201x (Net migration).

Column Name	**Data Type**	**Description**
Name	varchar (30)	Name of state. (e.g. Arizona)
POPESTIMATE 201X	bigint	Resident total population estimate for the state. There are 8 columns for 2010 to 2017.
NPOPCHG201X	bigint	Numeric change in resident total population. There are 8 columns for 2010 to 2017.
BIRTHS201X	bigint	Births in period 4/1/201X to 6/30/201X. There are 8 columns for 2010 to 2017.
DEATHS201X	bigint	Deaths in period 4/1/201X to 6/30/201X. There are 8 columns for 2010 to 2017.
NETMIG201X	bigint	Net migration from each state. There are 8 columns for 2010 to 2017.

During the data transformation, we pivoted the source table from wide format to long format in order to bring the year as a separate column. This transformation will help in making the joins easier as the tables would be in the same format.

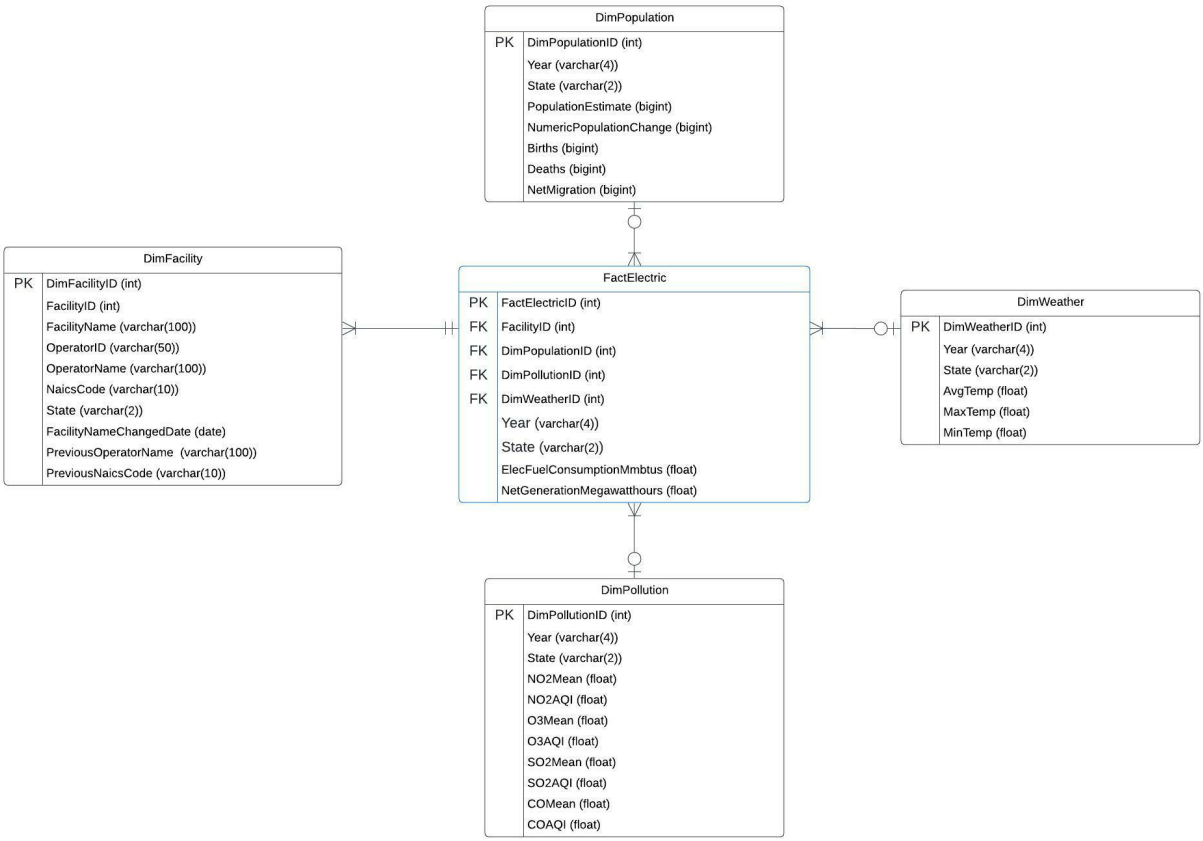
4. *WEATHER DATA (2015)*:

Data Source (csv): https://data.world/mattwinter225/2015-usa-weather-avg-max-min
Filename: 2015 USA Weather Data FINAL.csv
This dataset contains weather data captured at weather stations in all 50 states during 2015. The data is recorded daily and includes AvgTemp, MaxTemp, and MinTemp for each day/station. Since this contains data only for 2015, we will make up data for 2010-2014 and 2016.

Column Name	**Data Type**	**Description**
STATION	varchar (50)	Station Code
STATION_NAME	varchar (80)	Station Name
AvgTemp	float	Average temperature for the day at the weather station
MaxTemp	float	Maximum temperature for the day at the weather station
MinTemp	float	Minimum temperature for the day at the weather station
State Name	varchar (30)	Name of state. (e.g. Arizona)
Date	date	Date when the weather data is logged.

Data Model:

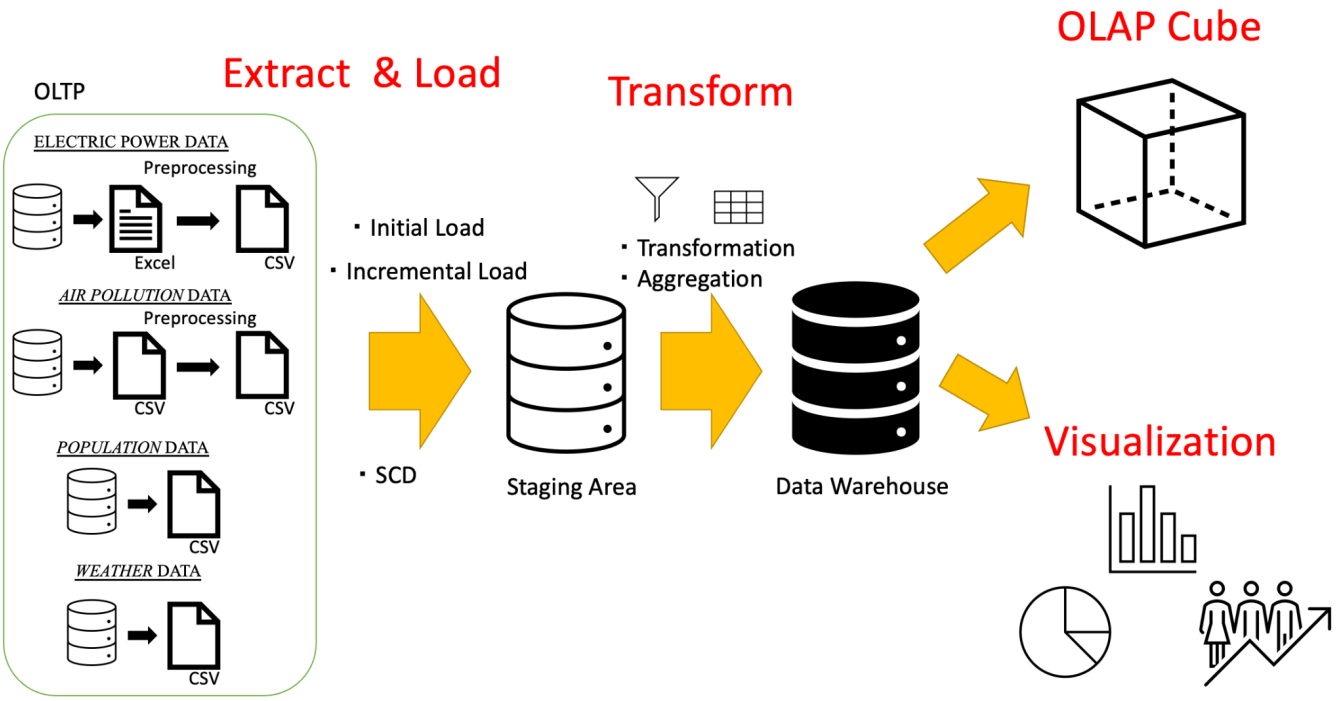
Our current data model is shown below:



In our data warehouse, the fact table is “FactElectric" which stores facts about electric power consumption and generation. The dimension tables are “DimFacility", “DimPopulation”, “DimPollution”, and “DimWeather”. We choose state and year as our dimensions for the OLAP Cube and NO2Mean, NO2AQI, O3Mean, O3AQi, COMean, COAQL, SO2Mean, SO2AQI, AvgTemp, MaxTemp, MinTemp, ElecFuelConsumptionMmbtus, NetGenerationMegawatthours etc as our measures. This was all done to see the trends in the data we used.

Data Flow:

Below is the overview of our data flow:



Staging Area and Data Warehouse are hosted on SQL Server, and we perform the ELT and other processes above using SSIS, SSAS, and Tableau

After extracting the data from the OLTP system, we first load them into the staging area to keep raw data for the purpose of troubleshooting and auditing and to avoid unnecessary workload on the OLTP system in case we need to load the same data again.

Next, we will perform the data transformations described below:

- Electric Power Data
  - Aggregate (sum) the values of "ElecFuelConsumptionMmbtus" and "NetGenerationMegawatthours" columns by "FacilityID", "Year", and "State"
  - Generate primary keys in DimFacility and FactElectric tables as we load data
- Air Pollution Data
  - Convert state names to two-letter state code by using (joining) a new table StateLookup (e.g. Arizona -> AZ)
    - Reference: https://gist.github.com/esfand/9443427
  - Extract year from the "Data Local" column and create a new columns "Year"

    - Aggregate (average) “NO2Mean”, ”NO2AQI”, “O3Mean”, “O3AQI”, “SO2Mean”, “SO2AQI”, “COMean”, and “COAQI” values by “Year” and “State”
    - Generate primary key in DimPollution table as we load data
- Population Data
    - Convert state names to two-letter state code by using (joining) a new table StateLookup (e.g. Arizona -> AZ)
    - Extract year from the population statistics columns and convert the wide format to the long format
        - For instance, the POPESTIMATE201X columns will be formatted as:

Year	POPESTIMATE
2010	123,456,789
2011	223,456,789
2012	323,456,789
2013	423,456,789
2014	523,456,789
2015	623,456,789
2016	723,456,789
2017	823,456,789

    - Generate primary key in DimPopulation table as we load data
- Weather Data
    - Convert state names to two-letter state code by using (joining) a new table StateLookup (e.g. Arizona -> AZ)
    - Extract year from the “Data” column and create a new columns “Year”
    - Aggregate “AvgTemp” (average), “MaxTemp” (max), and “MinTemp” (minimum) values by “Year” and “State”
    - Generate primary key in DimWeather table as we load data

We store these transformed data into a data warehouse, create an OLAP cube, and visualize the data. The datasets from 2010 to 2016 are initially loaded, and the 2017 dataset is used to perform the incremental load.

Slowly Changing Dimensions (SCD) are set on the “FacilityName“, “OperatorID”, “OperatorName”, and “NaicsCode” columns in the Facility dimension (DimFacility). Types of SCD are as follows:

- "*FacilityName*" : Type 2 - Creating a new additional record
    - The "FacilityNameChangedDate" column is default to Null, and when the "FacilityName" change happens, the value of "FacilityNameChangedDate" is set to the date of the change.
- "OperatorID": Type 1 - Overwriting the old value
- "OperatorName": Type 3 - Adding a new column "PreviousOperatorName" (default to Null)
- "NaicsCode": Type 3 - Adding a new column "PreviousNaicsCode" (default to Null)

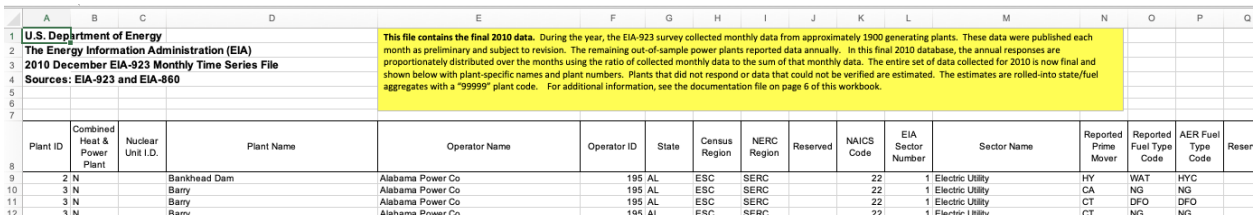
If "OperatorID", "OperatorName" and/or "NaicsCode" changes at the same time as "FacilityName", change(s) on "OperatorID", "OperatorName" and/or "NaicsCode" will take place on a new record created due to the Type 2 SCD on "FacilityName".

Data Preprocessing:

Electric Power Data:

Before loading the Electric Power Data using SSIS, we followed the steps below and worked on cleaning the Electric Power dataset using Python:

1. Remove the unnecessary lines at the beginning of each file
    a. There are 7 ".xls" and ".xlsx" files (2010-2017), and all of the files contain the 5 to 7 unnecessary lines for the file metadata as shown below: (The number of unnecessary rows varies each file.)
    b. When reading these files into panda's DataFrame, we skipped these unnecessary lines.



U.S. Department of Energy
The Energy Information Administration (EIA)
2010 December EIA-923 Monthly Time Series File
Sources: EIA-923 and EIA-860

This file contains the final 2010 data. During the year, the EIA-923 survey collected monthly data from approximately 1900 generating plants. These data were published each month as preliminary and subject to revision. The remaining out-of-sample power plants reported data annually. In this final 2010 database, the annual responses are proportionately distributed over the months using the ratio of collected monthly data to the sum of that monthly data. The entire set of data collected for 2010 is now final and shown below with plant-specific names and plant numbers. Plants that did not respond or data that could not be verified are estimated. The estimates are rolled-into state/fuel aggregates with a "99999" plant code. For additional information, see the documentation file on page 6 of this workbook.

Plant ID	Combined Heat & Power Plant	Nuclear Unit I.D.	Plant Name	Operator Name	Operator ID	State	Census Region	NERC Region	Reserved	NAICS Code	EIA Sector Number	Sector Name	Reported Prime Mover	Reported Fuel Type Code	AER Fuel Type Code	Reser
2	N		Bankhead Dam	Alabama Power Co	195	AL	ESC	SERC		22	1	Electric Utility	HY	WAT	HYC	
3	N		Barry	Alabama Power Co	195	AL	ESC	SERC		22	1	Electric Utility	CA	NG	NG	
3	N		Barry	Alabama Power Co	195	AL	ESC	SERC		22	1	Electric Utility	CT	DFO	DFO	

2. Remove all records for the estimated data (FacilityName="State-Fuel Level Increment", FacilityID=99999), which are presented as aggregated records at the state/fuel type level
3. Some of the files have different column names from the other files. Thus, we changed the column names we will load into our Data Warehouse so that these required columns are the same across all files.
4. Some records in some of the files have "." for the "Year" column. We replaced "." with the appropriate Year values.

5. After these cleaning tasks are done, we export them into csv files while putting quotes around non-numeric columns.

Since the number of rows to remove from the beginning of each file and column names are different, data preprocessing is performed differently on each file. Please refer to "ElectricDataPreprocessing.ipynb" for the detailed implementation. For example, the Python code for the 2014 data file is as follows:

```
# Import module
import pandas as pd
import csv

# Read the original 2014 data file
df = pd.read_excel('./original/ELECTRIC_2014.xlsx', skiprows=5)

# Remove rows for the "State-Fuel Level Increment" facility
df.drop(df[df['Plant Id'] == 99999].index, axis='index', inplace=True)

# Format columns
df.rename(columns={'Plant Id': 'Plant Id', 'Plant Name': 'Plant Name', 'Operator Name': 'Operator Name',
                   'Operator Id': 'Operator Id', 'Plant State': 'State',
                   'Reported\nFuel Type Code': 'Reported Fuel Type Code',
                   'AER\nFuel Type Code': 'AER Fuel Type Code',
                   'Net Generation\n(Megawatthours)': 'Net Generation (Megawatthours)',
                   'Elec Fuel Consumption\nMMBtu': 'Elec Fuel Consumption MMBtu',
                   'YEAR': 'Year'}, inplace=True)

# Replace "." values for the 'Year' column with appropriate year values
df['Year'] = df['Year'].apply(lambda x: 2014 if x == '.' else x)

# Export to csv
df.to_csv('ele_2014.csv', index = False, quoting=csv.QUOTE_NONNUMERIC)
```

Air Pollution Data

The "Address" column in the original csv file contains a comma within some of its values. However, since values in the "Address" column are not surrounded by quotes, SSIS reads these values as two separate columns. To avoid this issue, we used Python, stored the csv file in pandas DataFrame, and exported to a new csv file with double quotes around non-numeric columns.

```
# Import modules
import pandas as pd
import csv
!pip install --upgrade xlrd
```

```
df = pd.read_csv('./pollution_us_2000_2016.csv')

df.to_csv('pollution.csv', index = False, quoting=csv.QUOTE_NONNUMERIC)
```

Data Generation:

Weather Data for the Initial and Incremental Load:

Since we were able to find the weather dataset for all U.S. states only from 2015, we manually generated weather data for 2010-2014 and 2016 for the initial load and 2017 for the incremental load. Using Python and its modules, we added random numbers between -3 and 3 to 2015's AvgTemp, MaxTemp, and MinTemp while keeping the values in other columns such as "STATION" or "STATION_NAME".

The Python code used for this data generation is as follows:

```
# Import modules
import pandas as pd
from google.colab import files
import random
import numpy as np
import csv
```

```
# Upload the 2015 data file
file = files.upload()
df_2015 = pd.read_csv('Weather_2015.csv', delimiter=';')
df_2015.head()
```

```
years = ['10', '11', '12', '13', '14', '16', '17']
quotedColumns = ['STATION', 'STATION_NAME', 'StateName', 'State', 'Date']

# Create a data file for missing years one by one
for year in years:
  df_new = df_2015.copy()
  temps = df_new[['AvgTemp', 'MaxTemp', 'MinTemp']].to_numpy()

  # add a rondom number between -3 and 3 to 'AvgTemp', 'MaxTemp', 'MinTemp' columns
  for i in range(len(temps)):
    temps[i][0] = temps[i][0] + random.uniform(-3, 3)
    temps[i][1] = temps[i][1] + random.uniform(-3, 3)
    temps[i][2] = temps[i][2] + random.uniform(-3, 3)

  df_new[['AvgTemp', 'MaxTemp', 'MinTemp']] = np.round(temps, 2)

  # Replace the year 2015 with an appropriate year
  new_value = '/' + year
  df_new['Date'] = df_new['Date'].replace(regex=r'/15', value=new_value)

  # Put double quotes around values for 'STATION', 'STATION_NAME', 'StateName', 'State', 'Date' as in the 2015 data file
  for c in quotedColumns:
    df_new[c] = '"' + df_new[c] + '"'

  # Put double quotes around column names as in the 2015 data file
  df_new.columns = '"' + df_new.columns + '"'

  # Export a generate data to csv
  df_new.to_csv('Weather_20' + year +'.csv', sep=';', index=False, quoting=csv.QUOTE_NONE)
```

2017 Data for the Incremental Load:

- Weather 2017 Data
    - Generated using Python as described above.
- Air Pollution 2017 Data
    - Generated using Python by adding random values between 0 and 1 to the 'NO2 Mean', 'NO2 AQI', 'O3 Mean', 'O3 AQI', 'SO2 Mean', 'SO2 AQI', 'CO Mean', 'CO AQI' columns in the 2016 data
    - Python code is as follows:

```
# Import modules
import pandas as pd
import csv
import random
!pip install --upgrade xlrd
```

```
# Read original data file for 2010 to 2016
df = pd.read_csv('./pollution_us_2000_2016.csv')

# Extract year from 'Data Local' column and add it as a new column
df['Year'] = pd.DatetimeIndex(df['Date Local']).year

# Filter only 2016 data
df_2017 = df[df['Year'] == 2016]

# Add random values between 0 and 1 to the numeric columns from 2016
DW_columns = ['NO2 Mean','NO2 AQI','O3 Mean','O3 AQI','SO2 Mean','SO2 AQI','CO Mean','CO AQI']
df_2017[DW_columns] = df_2017[DW_columns] + random.uniform(0, 1)

# Replace the year part 2016 with 2017
df_2017['Date Local'] = pd.DatetimeIndex(df_2017['Date Local'])  + pd.offsets.DateOffset(years=1)

# Drop 'Year' column
df_2017.drop(columns=['Year'], inplace=True)

# Export to csv
df_2017.to_csv('pollution2017.csv', index = False, quoting=csv.QUOTE_NONNUMERIC)
```

Slowly Changing Dimensions (SCD):

SCDs are set on the “FacilityName“, “OperatorID”, “OperatorName”, and “NaicsCode” columns in the Facility dimension (DimFacility) with its types as described below:

- “*FacilityName*“ : Type 2 - Creating a new additional record
    - The “FacilityNameChangedDate” column is default to Null, and when the “FacilityName” change happens, the value of “FacilityNameChangedDate” is set to the date of the change.
- “OperatorID”: Type 1 - Overwriting the old value
- “OperatorName”: Type 3 - Adding a new column “PreviousOperatorName” (default to Null)

- “NaicsCode”: Type 3 - Adding a new column “PreviousNaicsCode” (default to Null)

We select 5 records to change their values from the 2017 dataset and create a new csv file that contains only records with modified values.

The Python code used for this data generation is as follows:

```
# Read data file used for Electric 2017 incremental load
df_2017 = pd.read_csv('/Users/tomokikyotani/Downloads/DAMG_data/incremental/Electric/ele_2017.csv')

# Select 5 records to change
df_scd = df_2017.iloc[[0,1000,1500,5000,10000]].reset_index(drop=True)

# Change values of the selected 5 records
df_scd.loc[0, 'Plant Name'] = 'Boston SCD Facility'
df_scd.loc[1, 'Plant Name'] = 'Seattle SCD Facility'
df_scd.loc[2, 'Operator Id'] = 77777
df_scd.loc[2, 'Operator Name'] = 'Northeastern SCD Operator'
df_scd.loc[3, 'NAICS Code'] = 'NUSCD'
df_scd.loc[4, 'Plant Name'] = 'DAMG Facility'
df_scd.loc[4, 'NAICS Code'] = 'DAMG'

# Format column names
df_scd.rename(columns={'Plant Id': 'Plant Id', 'Plant Name': 'Plant Name', 'Operator Name': 'Operator Name',
                       'Operator Id': 'Operator Id', 'Plant State': 'State',
                       'Reported\nFuel Type Code': 'Reported Fuel Type Code',
                       'AER\nFuel Type Code': 'AER Fuel Type Code',
                       'Net Generation\n(Megawatthours)': 'Net Generation (Megawatthours)',
                       'Elec Fuel Consumption\nMMBtu': 'Elec Fuel Consumption MMBtu',
                       'YEAR': 'Year'}, inplace=True)

# Export to csv
df_scd.to_csv('ele_SCD.csv', index = False, quoting=csv.QUOTE_NONNUMERIC)
```

Initial Data Load to Staging Tables:

For the initial data load, we take the data files from all data sources for 2010 to 2016 and load them into our staging tables. During this process, all columns are loaded as “varchar”, and data type changes and data transformation will be performed when moving data from staging tables to Data Warehouse. (The implementation of the initial data load using SSIS is explained in the later “SSIS Implementation of Initial/Incremental Load and SCD” section.)

Electric Power Data:

1. Place all data files (2010 to 2016) in one folder.
2. Specify the path to the folder above in the variable “SourceFolder_Electric”
3. After adding a Flat File Source and specifying one of the data file in the folder, go to the “Expressions” property of the Flat File Source’s Connection Manager, set the “ConnectionString” property to a new empty variable “Filepath_Electric”
4. Using the Foreach Loop Container, Flat File Source, and the variables above, load all files in the folder (one file at a time) while selecting only the required columns out of 264 total columns.
5. Using OLE DB Destination, load the data into a staging table.

6. Repeat the steps above until it reads all files in the folder

Air Pollution Data:

1. All the data for 2010 to 2016 are in one csv file, so create one variable "Filepath_Pollution" with a value of the path to this data file.
2. For the "Expressions" property of the Flat File Source's Connection Manager, set the "ConnectionString" property to the variable "Filepath_Pollution"
3. Read them all using Flat File Source and load them into a staging table using OLE DB Destination.

Population Data:

1. All the data for 2010 to 2016 for the initial load and 2017 for the incremental load are in one csv file, so create one variable "Filepath_Population" with a value of the path to this data file.
2. For the "Expressions" property of the Flat File Source's Connection Manager, set the "ConnectionString" property to the variable "Filepath_Population"
3. Read them all using Flat File Source and load them into a staging table using OLE DB Destination.
4. Since this data is currently in the wide format, during the data transformation step, we will convert from the wide format to long format.

Weather Data:

1. Place all data files (2010 to 2016) in one folder.
2. Specify the path to the folder above in the variable "SourceFolder_Weather"
3. After adding a Flat File Source and specifying one of the data file in the folder, go to the "Expressions" property of the Flat File Source's Connection Manager, set the "ConnectionString" property to a new empty variable "SourceFolder_Weather"
4. Using the Foreach Loop Container, Flat File Source, and the variables above, load all files in the folder (one file at a time) while selecting only the required columns
5. Using OLE DB Destination, load the data into a staging table.
6. Repeat the steps above until it reads all files in the folder

Data Transformation and Loading to DataWarehouse:

As described in the "Data Flow" section above, the high-level overview of the data transformation for each dataset is as follows:

Electric Power Data:

- Aggregate (sum) the values of "ElecFuelConsumptionMmbtus*"* and "NetGenerationMegawatthours" columns by "FacilityID", "Year", and "State"

- Generate primary keys in DimFacility and FactElectric tables as we load data

Air Pollution Data:

- Convert state names to two-letter state code by using (joining) a newly created table StateLookup (e.g. Arizona -> AZ)
- Extract year from the "Data Local" column and create a new columns "Year"
- Aggregate (average) "NO2Mean", "NO2AQI", "O3Mean", "O3AQI", "SO2Mean", "SO2AQI", "COMean", and "COAQI" values by "Year" and "State"
- Generate primary key in DimPollution table as we load data

Population Data:

- Convert state names to two-letter state code by using (joining) a new table StateLookup (e.g. Arizona -> AZ)
- Extract year from the population statistics columns and convert the wide format to the long format
- Generate primary key in DimPopulation table as we load data

Weather Data:

- Convert state names to two-letter state code by using (joining) a new table StateLookup (e.g. Arizona -> AZ)
- Extract year from the "Data" column and create a new columns "Year"
- Aggregate "AvgTemp" (average), "MaxTemp" (max), and "MinTemp" (minimum) values by "Year" and "State"
- Generate primary key in DimWeather table as we load data

Other notes about this transformation:

- State name conversions are done using SQL by joining each staging table with a newly created table "StateLookup". The "StateLookup" table has two columns:
    - The "StateName" column which contains full state names
    - The "StateAbbrev" column which contains two-letter abbreviations of state names
- Primary key values in each dimension table and the fact table are generated as new records are inserted due to their IDENTITY column property.
- In the "FactElectric" table, "FacilityID", "DimPopulationID", "DimPollutionID", and "DimWeatherID" are assigned when joining all dimension tables by State and Year.

The implementation of data transformation and data load to Data Warehouse using SSIS is explained in the later "SSIS Implementation of Initial/Incremental Load and SCD" section.

Incremental Data Load:

After the initial load, the incremental 2017 data for the Electric Power Data, Weather Data, and Air Pollution Data are loaded. (The 2017 Population data is already loaded in the wide-format initial load dataset.)

The Electric Power data is given in the original data source website, and we manually created the 2017 data for Weather Data and Air Pollution Data as described in the "Data Generation" section.

The implementation of incremental data load using SSIS is explained in the later "SSIS Implementation of Initial/Incremental Load and SCD" section.

Slowly Changing Dimensions (SCD):

SCDs are set on the "*FacilityName*", "OperatorID", "OperatorName", and "NaicsCode" columns in the facility dimension table (DimFacility).

Detailed explanations on these SCDs are as follows:

"*FacilityName*" :

- This column will have the Type 2 SCD.
- Upon the change of "FacilityName", we first set the "FacilityNameChangedDate" (default to Null) to the date of change. Next, a new record is created with the new FacilityName and all other values same as the old record.
- If a change on "OperatorID", "OperatorName" and/or "NaicsCode" happen at the same time as "FacilityName" change, all changes are reflected only on a new record.

"OperatorID":

- This column will have the Type 1 SCD.
- Upon the change of the value in the "OperatorID" column, we simply overwrite the old value.

"OperatorName":

- This column will have the Type 3 SCD.

- We have a column”PreviousOperatorName”, and upon the change of the value in the “OperatorName” column, we store the old value to the ”PreviousOperatorName” column and change the value in the “OperatorName” column to a new value.

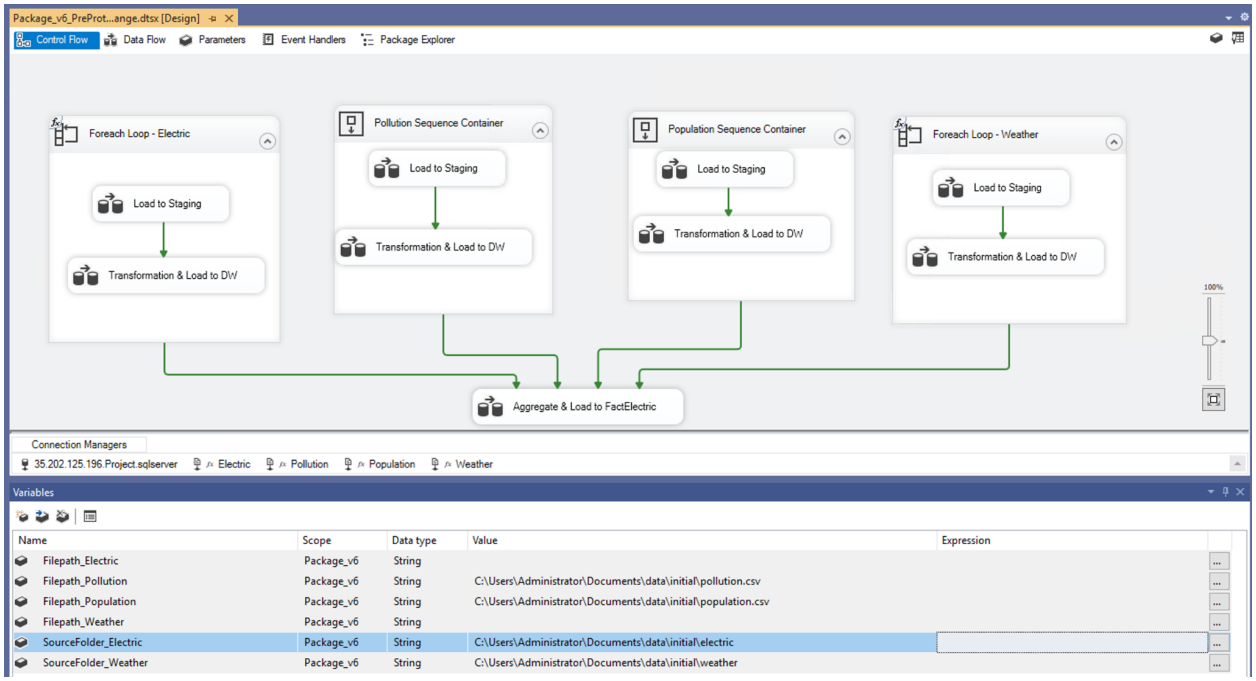
“NaicsCode”:

- This column will have the Type 3 SCD.
- We have a column”PreviousNaicsCode”, and upon the change of the value in the “NaicsCode” column, we store the old value to the ”PreviousNaicsCode” column and change the value in the “NaicsCode” column to a new value.

The implementation of SCD using SSIS is explained in the next “SSIS Implementation of Initial/Incremental Load and SCD” section.

SSIS Implementation of Initial/Incremental Load and SCD:

1. Control Flow and Variables Setting

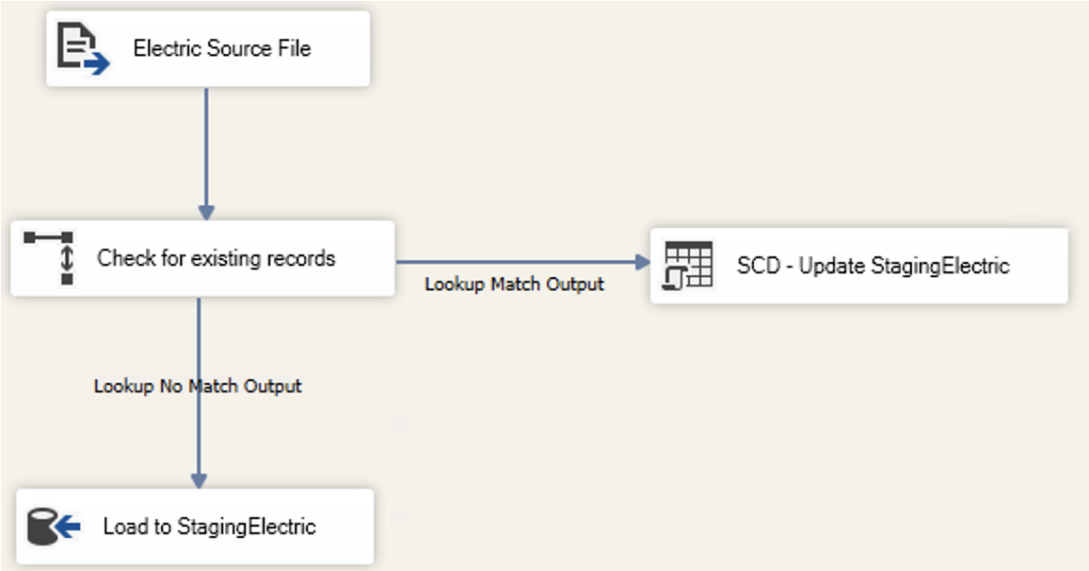


- Using variables, specify a path to the folder that contains data files or a path to a data file to load.
- For Electric Power Data and Weather Data, the Foreach Loop Container allows the “Load to Staging” Data Flow to read data files in the folder specified above one by one.
- The “Load to Staging” Data Flow Task reads data in a source file and loads them into a staging table.

- The “Transformation & Load to DW” Data Flow Task reads data from a staging table, performs data transformation such as data type conversion and aggregation, combines or splits data as necessary using Lookup or Conditional Split, and loads transformed data into each dimension tables.
- After all data is loaded into the dimension tables, the “Aggregate & Load to FactElectric” Data Flow merge-joins all necessary tables and loads data into the fact table.

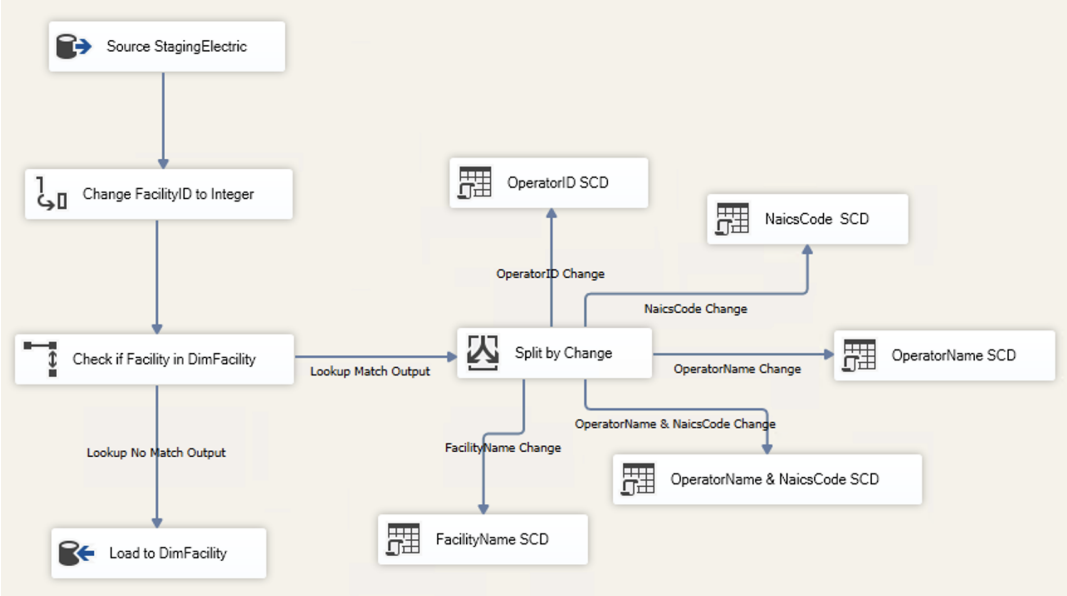
2. Electric Power Data Load

- “Load to Staging” Data Flow



  - Read data from a source file using Flat File Source
  - Since some of the columns read from this source file are set as SCD, using Lookup, route new data to the OLE DB Destination to insert into the staging table (StagingElectric) and data for SCD to the OLE DB Command to update an existing record in StagingElectric.

- “Transformation & Load to DW” Data Flow



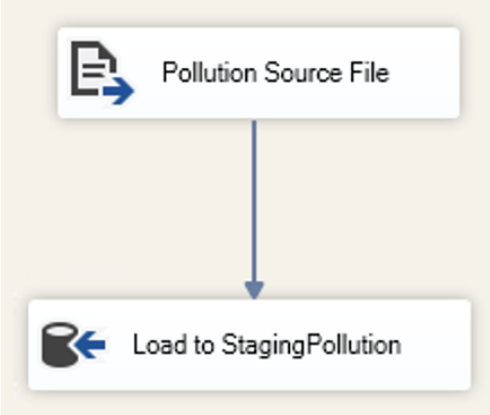
- Using the OLE DB Source and SELECT query below, fetch records from StagingElectric that have been updated due to SCD or records that are new and not in the facility dimension table (DimFacility).

```
select
    distinct a.FacilityID
    , a.FacilityName
    , a.OperatorID
    , a.OperatorName
    , a.NaicsCode
    , a.[State]
    , df.FacilityName as [FacilityName_currentSource]
    , df.FacilityNameChangedDate as [FacilityNameChangedDate_currentSource]
from StagingElectric a
join DimFacility df
    on a.FacilityID = df.FacilityID
where a.[Year] = (select max(Year(b.[Year])) from StagingElectric b)
    and df.FacilityNameChangedDate is Null
    and (a.FacilityName != df.FacilityName
        or a.OperatorID != df.OperatorID
        or a.OperatorName != df.OperatorName
        or a.NaicsCode != df.NaicsCode
    )
union
select
    distinct c.FacilityID
    , c.FacilityName
    , c.OperatorID
    , c.OperatorName
    , c.NaicsCode
    , c.[State]
    , Null as [FacilityName_currentSource]
    , Null as [FacilityNameChangedDate_currentSource]
from StagingElectric c
where c.FacilityID not in (select distinct d.FacilityID from DimFacility d);
```

- Change the data type of Facility ID to “Four byte signed integer”
- Using Lookup, route new facility data to the OLE DB Destination to insert into DimFacility and data for SCD to the further conditional split.
- When Lookup is used above, the current values in DimFacility are obtained. By comparing the current values in DimFacility to the updated values in StagingElectric, split data to the proper OLE DB Command to perform appropriate updates for SCD.

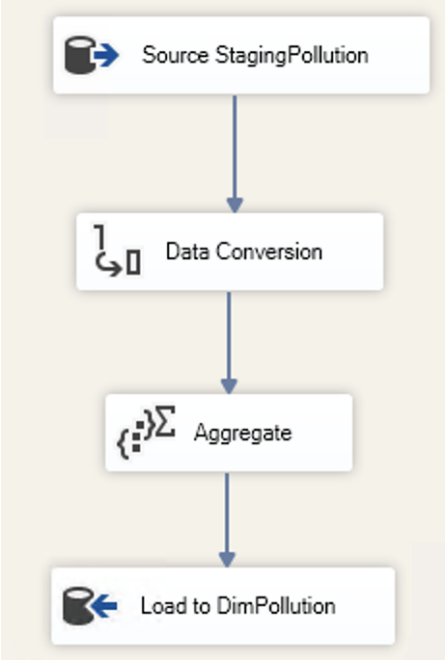
3.Air Pollution Data Load

- “Load to Staging” Data Flow



- Read data from a source file using Flat File Source
- Load to the staging table (StagingPollution)

- “Transformation & Load to DW” Data Flow



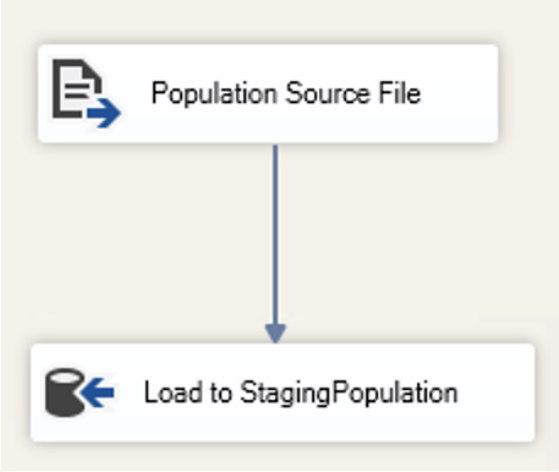
- Using the OLE DB Source and SELECT query below, fetch records from StagingPollution that are new and not in the facility dimension table (DimPollution). Since the data load happens yearly, the WHERE Clause checks for Year values in both StagingPollution and DimPollution to determine if it is a new record or not. In addition, by joining the StateLookup table, which contains records about each state's full state name and two-letter abbreviation, convert state full name to two-letter abbreviation.

```
select
    Year(a.Date) as [Year]
    , s.StateAbbrev as [State]
    , NO2Mean
    , NO2AQI
    , O3Mean
    , O3AQI
    , SO2Mean
    , SO2AQI
    , COMean
    , COAQI
from StagingPollution a
join StateLookup s
    on a.State = s.StateName
where Year(a.[Date]) not in (select distinct [Year] from DimPollution)
```

- Using the Data Conversion, convert data types of input columns to match the data types in DimPollution.
- Aggregate (average) the data (e.g. NO2Mean, NO2AQI, etc..) by State and Year
- Load aggregated data into DimPollution

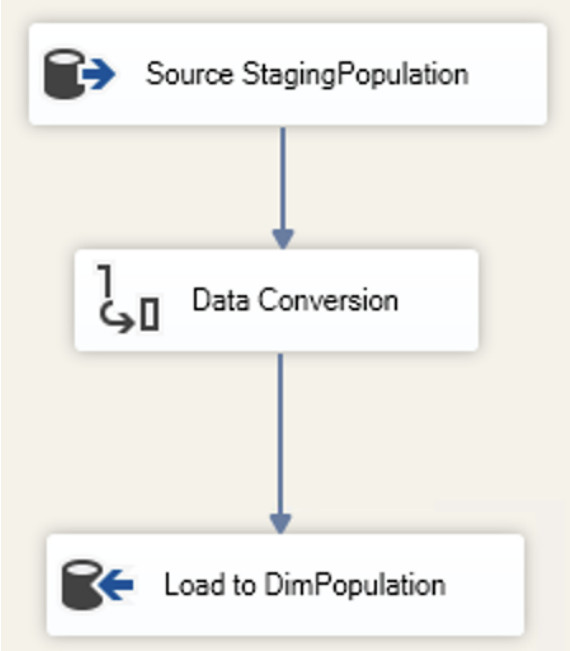
4. Population Data Load

- "Load to Staging" Data Flow



- Read data from a source file using Flat File Source
- Load to the staging table (StagingPopulation)

- “Transformation & Load to DW” Data Flow



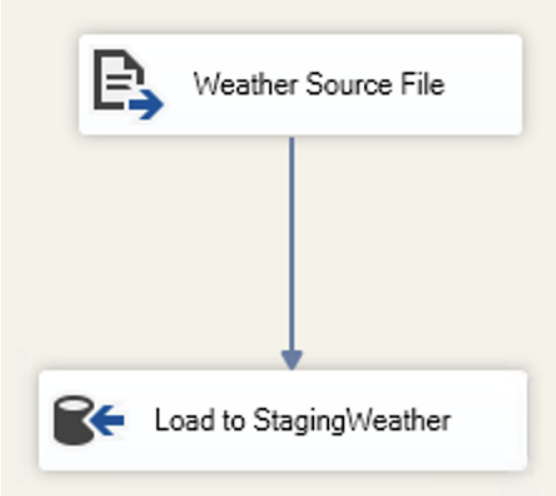
  - Using the OLE DB Source and SELECT query below, fetch records from StagingPopulation. After the initial load, StagingPopulation contains all data from 2010 to 2017, which we are interested in for this project, and all data from 2010 to 2017 are loaded at once to the population dimension table (DimPopulation). In addition, by joining the StateLookup table, which contains records about each state’s full state name and two-letter abbreviation, convert state full name to two-letter abbreviation.

```
with temp1 as (
    SELECT [State], [DescYear], [Count]
    FROM (
        select * from StagingPopulation
    ) p
    UNPIVOT
    ([Count] FOR [DescYear] IN (PopulationEstimate2010, PopulationEstimate2011, PopulationEstimate2012, PopulationEstimate2013, PopulationEstimate2014, PopulationEstimate2015,
                                PopulationEstimate2016, PopulationEstimate2017, NumericPopulationChange2010, NumericPopulationChange2011, NumericPopulationChange2012,
                                NumericPopulationChange2013, NumericPopulationChange2014, NumericPopulationChange2015, NumericPopulationChange2016, NumericPopulationChange2017,
                                Births2010, Births2011, Births2012, Births2013, Births2014, Births2015, Births2016, Births2017, Deaths2010, Deaths2011, Deaths2012, Deaths2013,
                                Deaths2014, Deaths2015, Deaths2016, Deaths2017, NetMigration2010, NetMigration2011, NetMigration2012, NetMigration2013, NetMigration2014,
                                NetMigration2015, NetMigration2016, NetMigration2017)) as unpvt
),
temp2 as (
    select
        right(DescYear, 4) as [Year]
        , s.StateAbbrev as [State]
        , substring(DescYear, 0, len(DescYear)-3) as [Description]
        , [Count]
    from temp1 a
    join StateLookup s
        on a.State = s.StateName
    where Year(right(DescYear, 4)) not in (select distinct [Year] from DimPopulation)
    and s.StateAbbrev not in (select distinct [State] from DimPopulation)
)
select
    [Year]
    , [State]
    , [PopulationEstimate]
    , [NumericPopulationChange]
    , [Births]
    , [Deaths]
    , [NetMigration]
from temp2
pivot(
    max([Count]) for [Description] in ([PopulationEstimate], [NumericPopulationChange], [Births], [Deaths], [NetMigration])
) as PivotTable;
```

- Using the Data Conversion, convert data types of input columns to match the data types in DimPopulation.
- Load data into DimPopulation

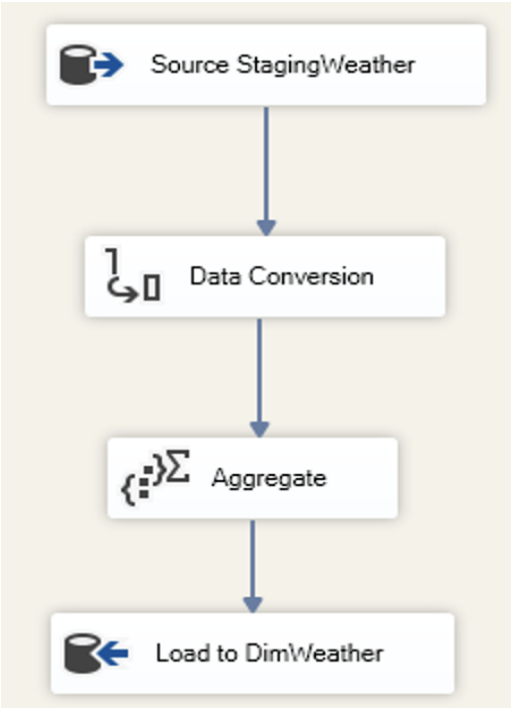
5. Weather Data Load

- "Load to Staging" Data Flow



- Read data from a source file using Flat File Source
- Load to the staging table (StagingWeather)

- "Transformation & Load to DW" Data Flow

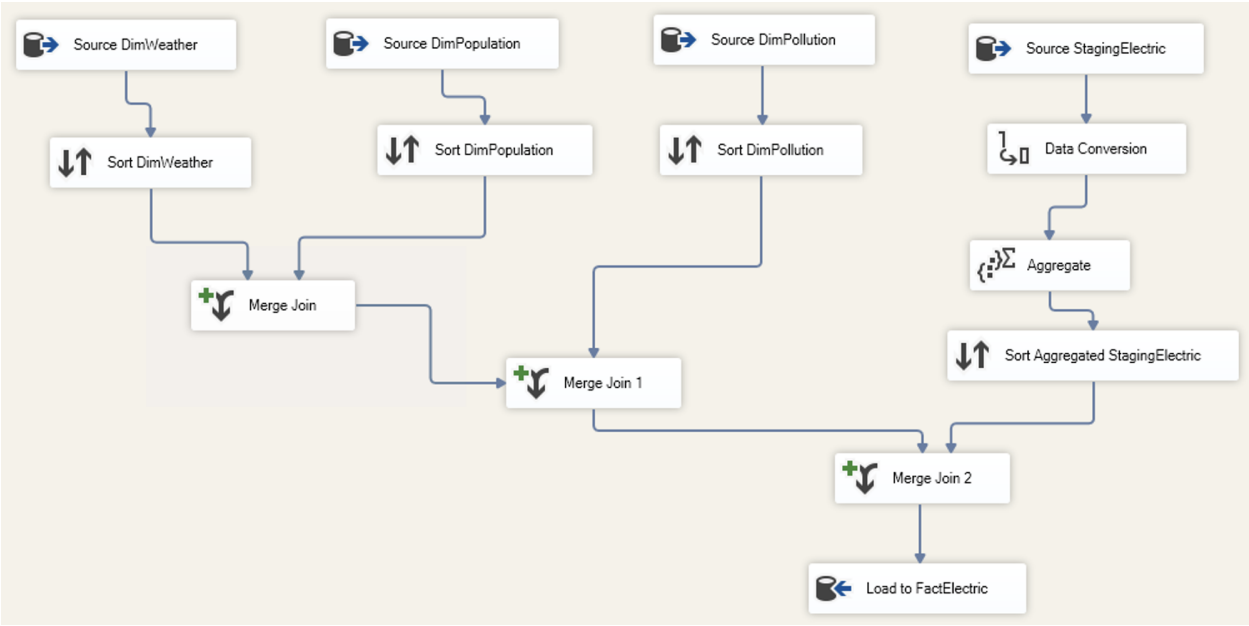


- Using the OLE DB Source and SELECT query below, fetch records from StagingWeather that are new and not in the weather dimension table (DimWeather). Since the data load happens yearly, the WHERE Clause checks for Year values in both StagingWeather and DimWeather to determine if it is a new record or not. In addition, by joining the StateLookup table, which contains records about each state's full state name and two-letter abbreviation, convert state full name to two-letter abbreviation.

```
select
    Year(a.Date) as [Year]
    , s.StateAbbrev as [State]
    , AvgTemp
    , MaxTemp
    , MinTemp
from StagingWeather a
join StateLookup s
    on a.State = s.StateName
where Year(a.[Date]) not in (select distinct [Year] from DimWeather)
```

- Using the Data Conversion, convert data types of input columns to match the data types in DimWeather.
- Aggregate the data (Average for AvgTemp, Maximum for MaxTemp, Minimum for MinTemp) by State and Year
- Load aggregated data into DimWeather

6. Aggregate & Load to FactElectric



- Merge Joins (Full Outer Join) data in DimWeather, DimPopulation, and DimPollution by State and Year…(1)
- Fetch data from StagingElectric, convert data types from varchar to float for ElecFuelConsumptionMmbtus and NetGenerationMegawatthours, and aggregate them by FacilityID, State, and Year… (2)
- Merge Joins (Left Outer Join - Left: (2) & Right: (1)) data (1) and (2) by State and Year
- Load aggregated and merged data into the fact table (FactElectric)

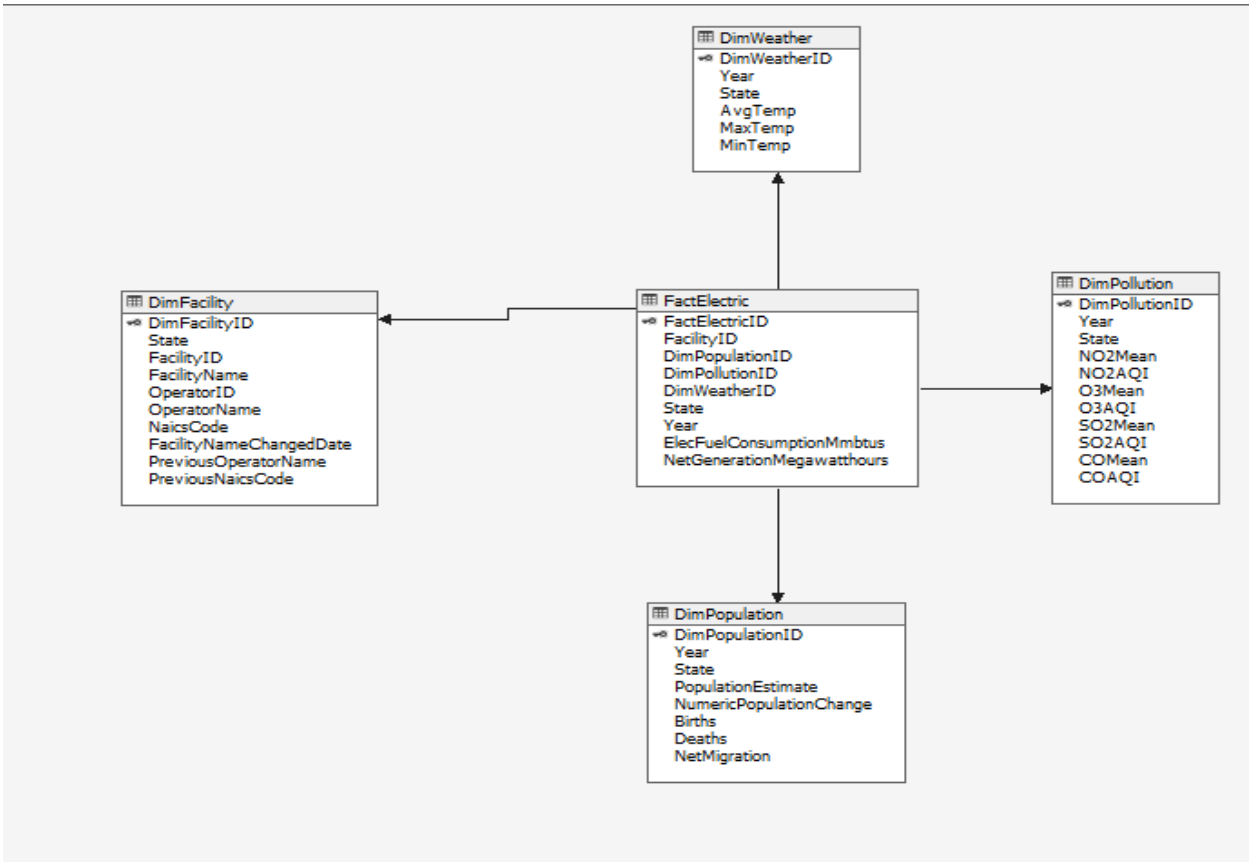
OLAP Cube:

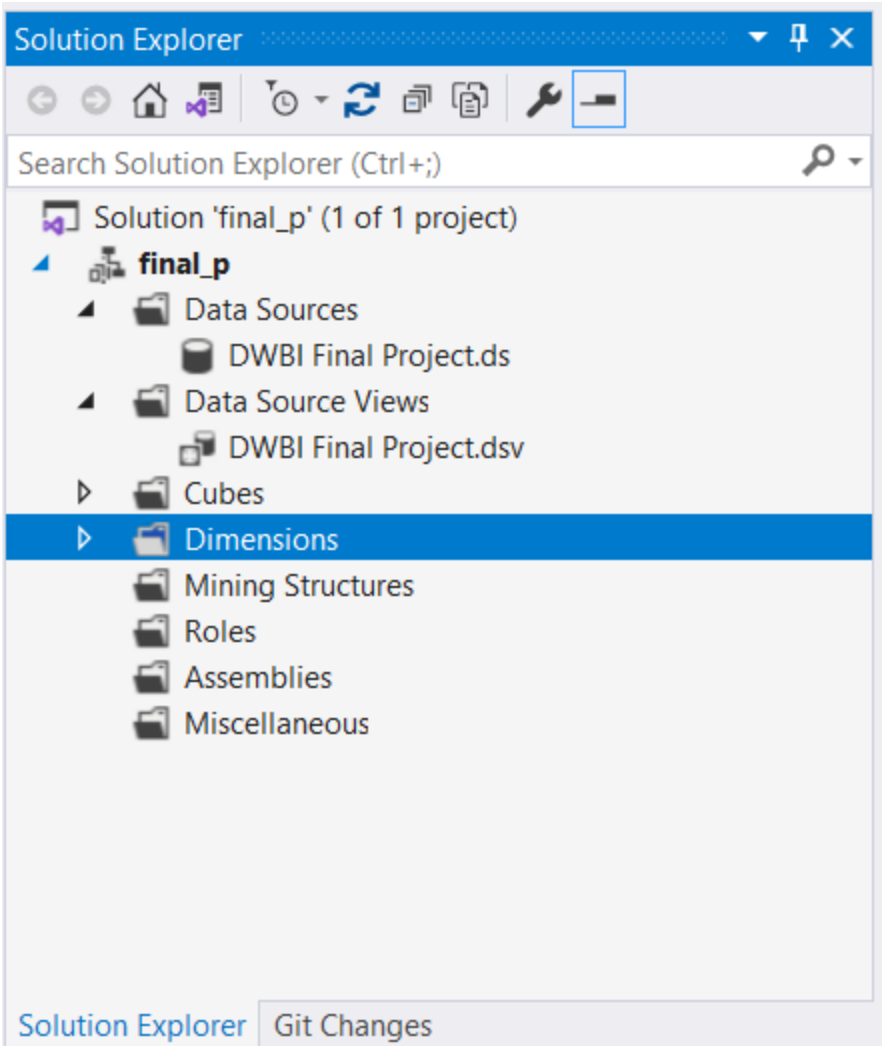
An OLAP cube is a type of data structure that allows faster analysis on the dimensions that are used to define a business problem. We built an OLAP cube to get better insights on the data when combined from all the tables and also helped us seeing the general trends based on Year, State, etc.

SSAS Implementation of OLAP Cube:

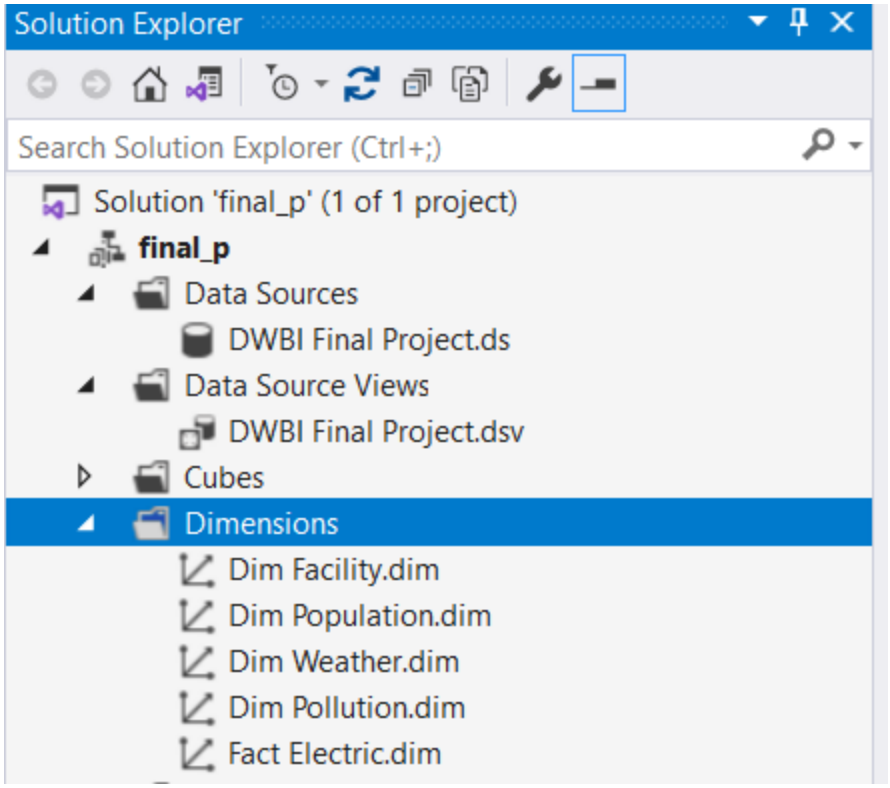
We used SSAS (SQL Server Analysis Services) to implement the cube:

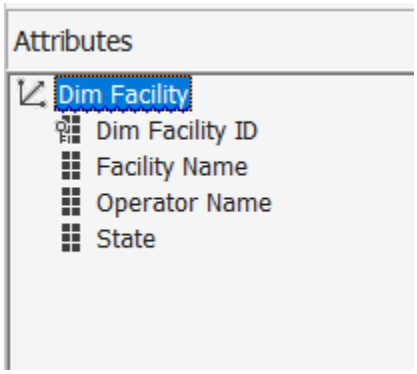
- Created a Data Source with tables (dimPopulation, dimFacility, dimPollution, FactElectric) by connecting to the SQL Server instance and created a data source view using the data source.

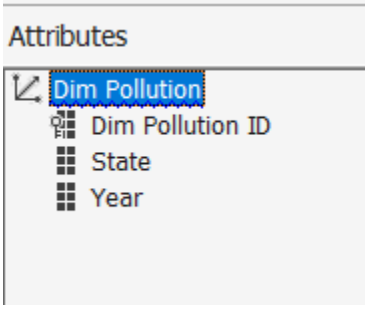

DimWeather
DimWeatherID
Year
State
AvgTemp
MaxTemp
MinTemp
DimFacility
DimFacilityID
State
FacilityID
FacilityName
OperatorID
OperatorName
NaicsCode
FacilityNameChangedDate
PreviousOperatorName
PreviousNaicsCode
FactElectric
FactElectricID
FacilityID
DimPopulationID
DimPollutionID
DimWeatherID
State
Year
ElecFuelConsumptionMmbtus
NetGenerationMegawatthours
DimPollution
DimPollutionID
Year
State
NO2Mean
NO2AQI
O3Mean
O3AQI
SO2Mean
SO2AQI
COMean
COAQI
DimPopulation
DimPopulationID
Year
State
PopulationEstimate
NumericPopulationChange
Births
Deaths
NetMigration

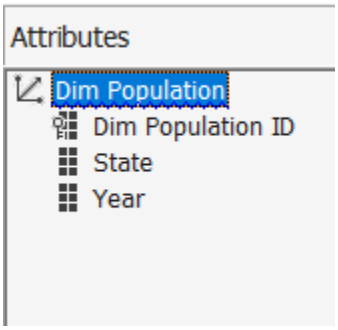


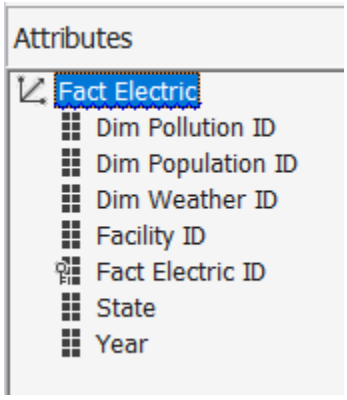
- Added the dimensions for the cube.
  Dimensions are basically objects used to group by the data in the cube.





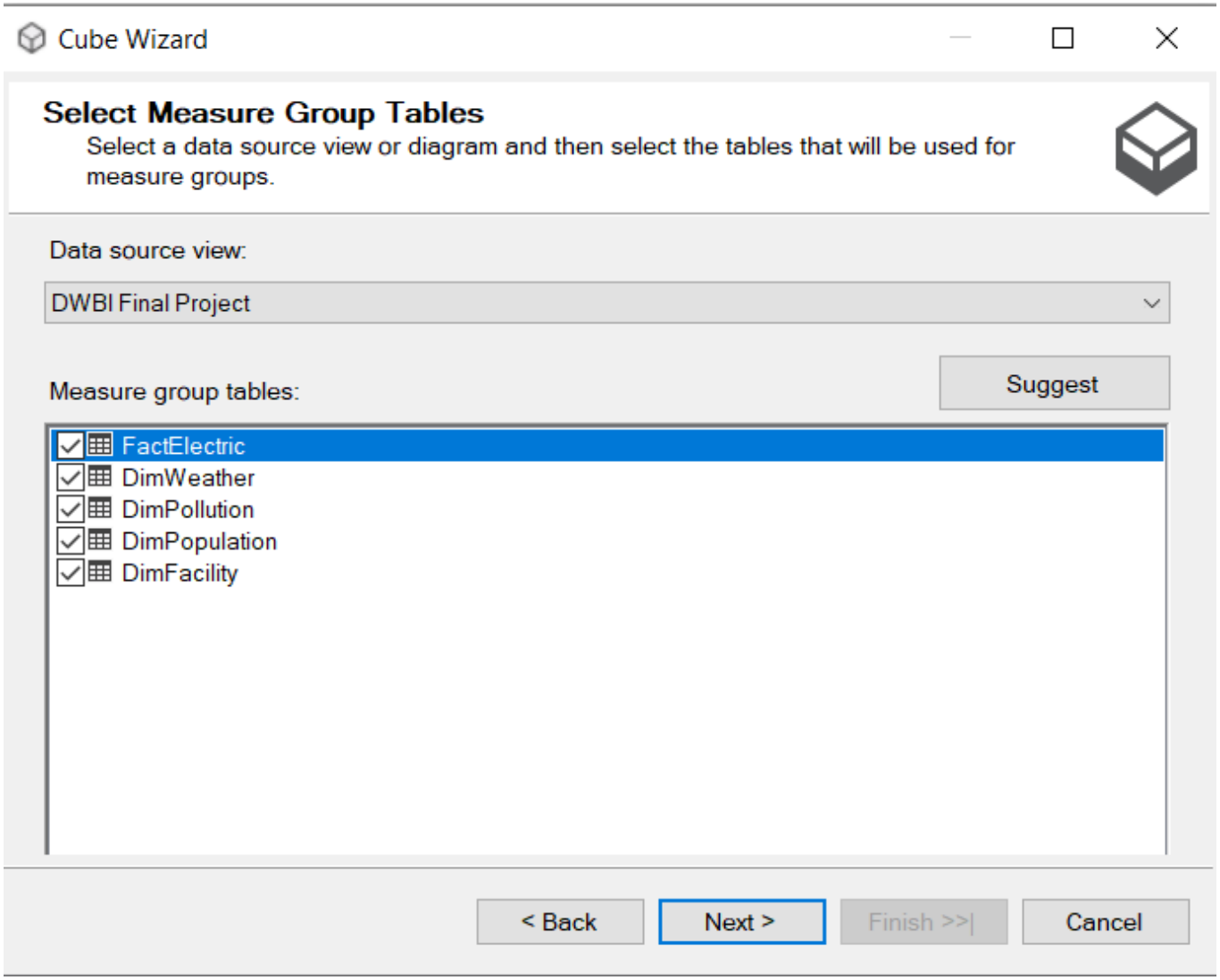


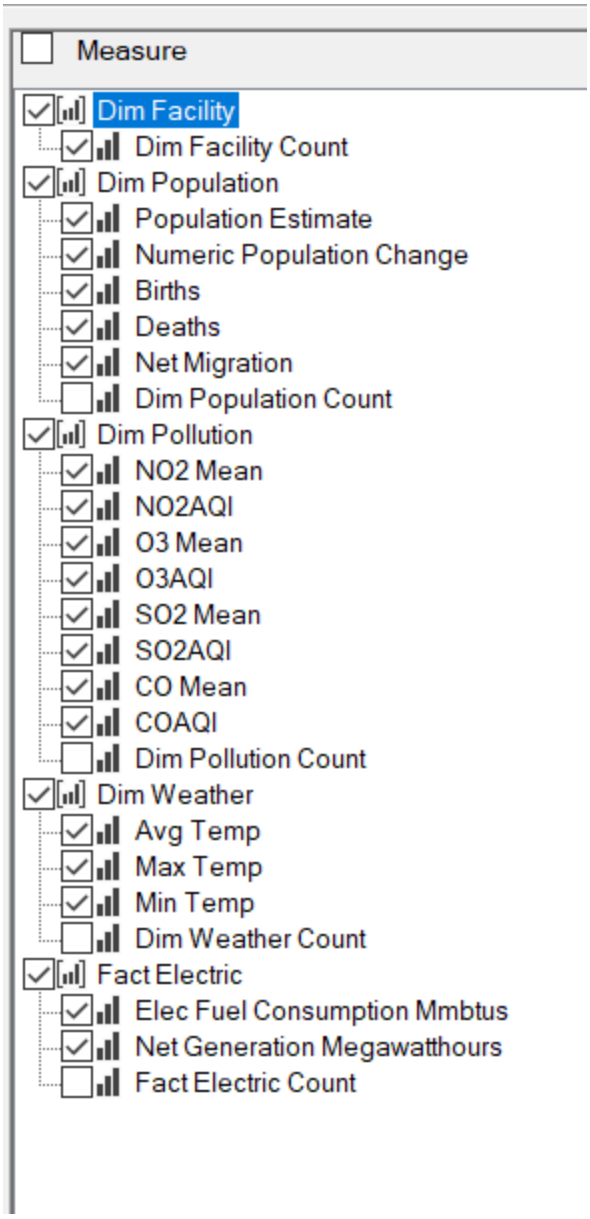


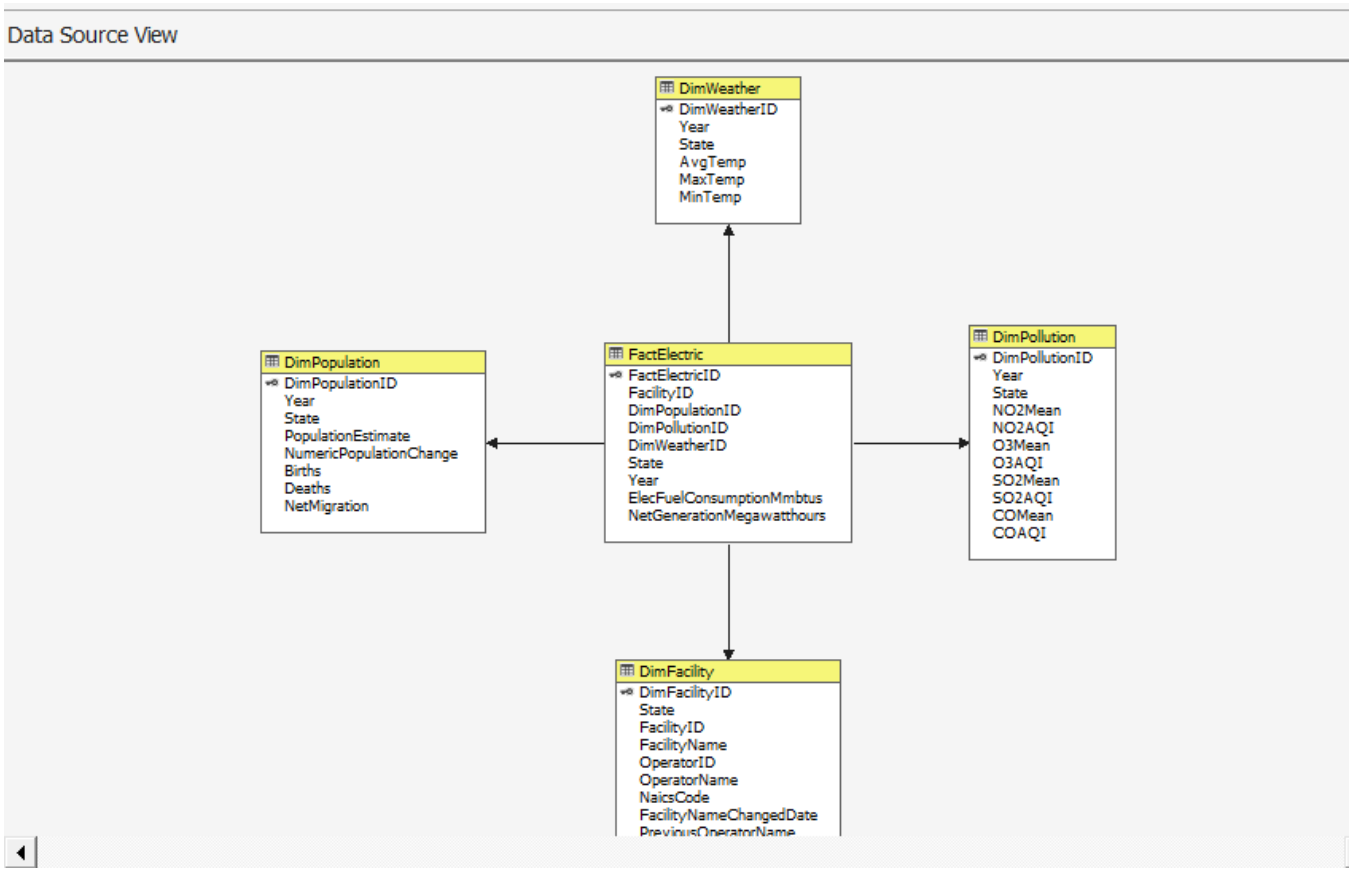


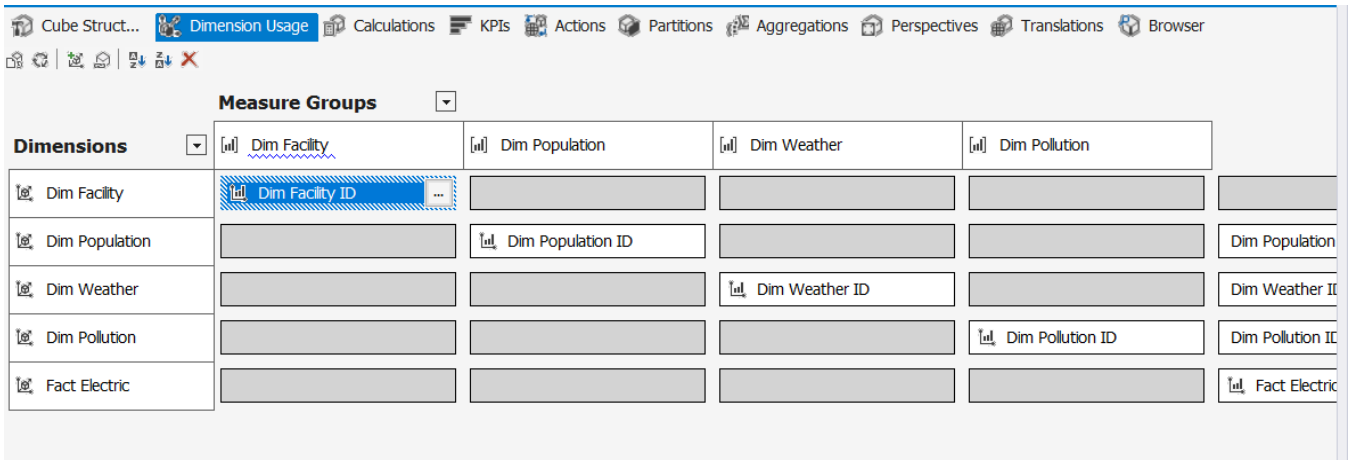
Above are the images of the dimensions used in our cube from various tables.

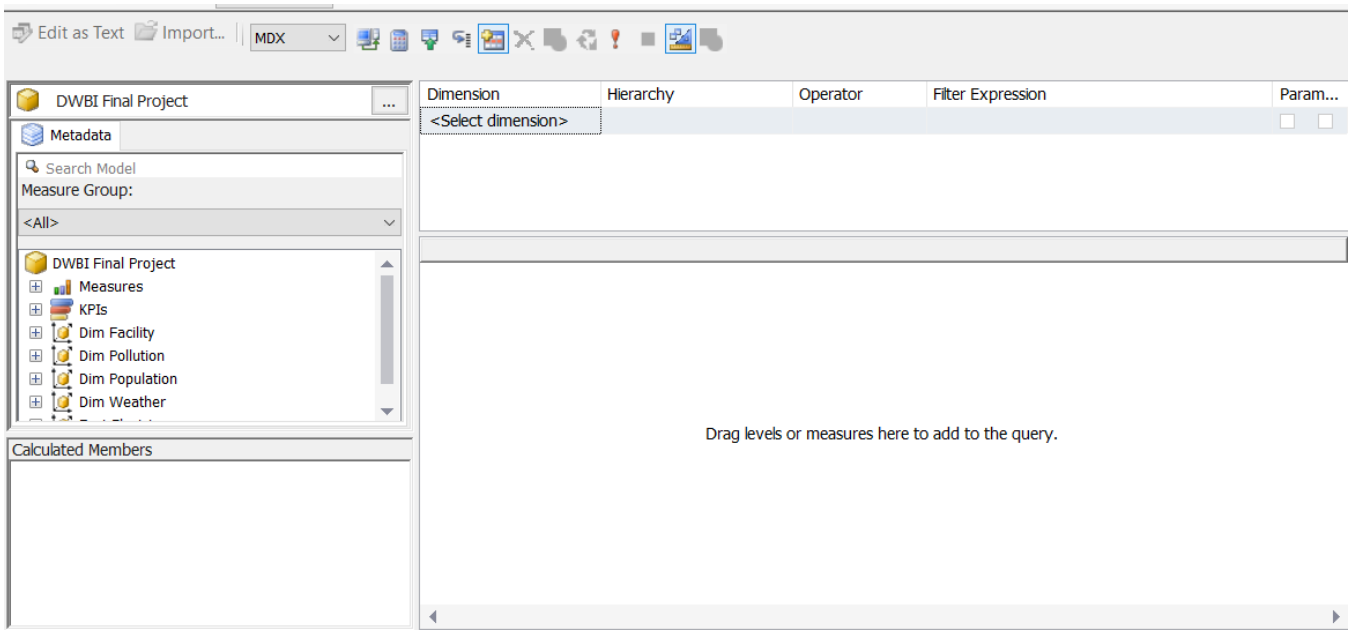
- Built a cube and then processed it with the above given dimensions and measures as shown below in order to deploy the cube.


Cube Wizard
Select Measure Group Tables
Select a data source view or diagram and then select the tables that will be used for measure groups.
Data source view:
DWBI Final Project
Measure group tables:
Suggest
FactElectric
DimWeather
DimPollution
DimPopulation
DimFacility
< Back
Next >
Finish >>|
Cancel

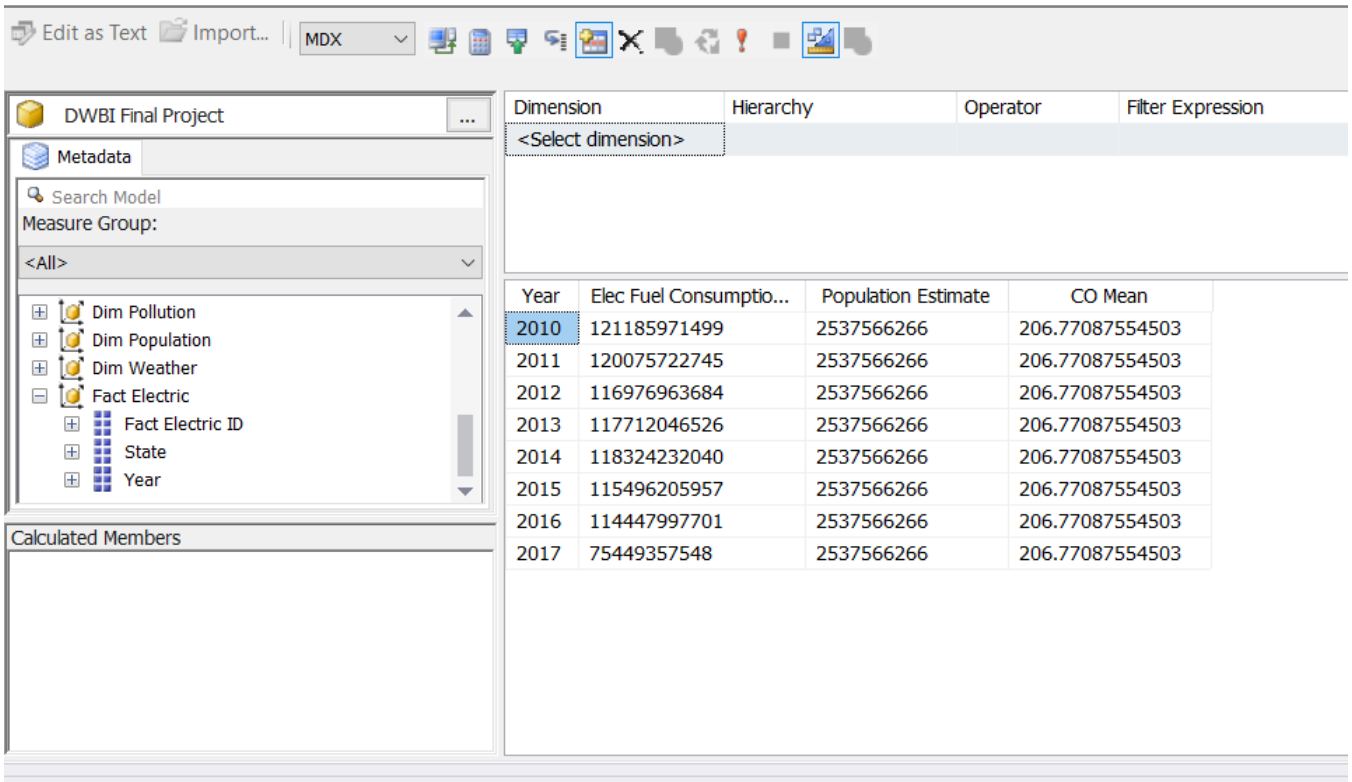

Measure
Dim Facility
Dim Facility Count
Dim Population
Population Estimate
Numeric Population Change
Births
Deaths
Net Migration
Dim Population Count
Dim Pollution
NO2 Mean
NO2AQI
O3 Mean
O3AQI
SO2 Mean
SO2AQI
CO Mean
COAQI
Dim Pollution Count
Dim Weather
Avg Temp
Max Temp
Min Temp
Dim Weather Count
Fact Electric
Elec Fuel Consumption Mmbtus
Net Generation Megawatthours
Fact Electric Count


Data Source View
DimWeather
DimWeatherID
Year
State
AvgTemp
MaxTemp
MinTemp
DimPopulation
DimPopulationID
Year
State
PopulationEstimate
NumericPopulationChange
Births
Deaths
NetMigration
FactElectric
FactElectricID
FacilityID
DimPopulationID
DimPollutionID
DimWeatherID
State
Year
ElecFuelConsumptionMmbtus
NetGenerationMegawatthours
DimPollution
DimPollutionID
Year
State
NO2Mean
NO2AQI
O3Mean
O3AQI
SO2Mean
SO2AQI
COMean
COAQI
DimFacility
DimFacilityID
State
FacilityID
FacilityName
OperatorID
OperatorName
NaicsCode
FacilityNameChangedDate
PreviousOperatorName


Cube Struct...
Dimension Usage
Calculations
KPIs
Actions
Partitions
Aggregations
Perspectives
Translations
Browser
Measure Groups
Dimensions
Dim Facility
Dim Population
Dim Weather
Dim Pollution
Dim Facility
Dim Facility ID
Dim Population
Dim Population ID
Dim Population
Dim Weather
Dim Weather ID
Dim Weather ID
Dim Pollution
Dim Pollution ID
Dim Pollution ID
Fact Electric
Fact Electric



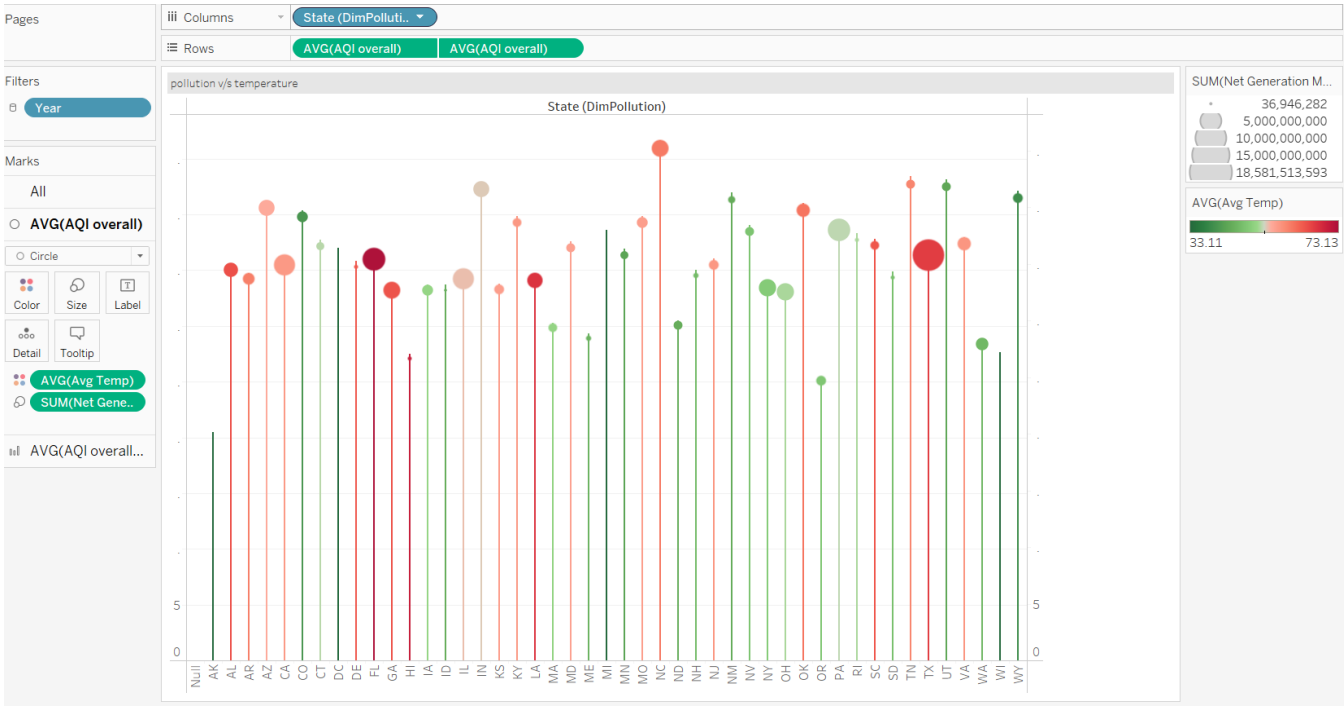
- Below is an example of Population Estimate and Electric fuel consumption based on year.



Questions to Answer:

1. What kind of relationship/impact does Electric Power generation and fuel consumption have with various pollutions?

   This question can be answered with the help of the graph below.



   The above lollipop chart represents the US states in the x-axis, air quality index in the y-axis. The color represents the temperature (lower temperature in green and higher temperature in red) and the size of the lollipop head represents the net electricity generation.

   Through this graph, we can observe that as the electricity generation increases, the average temperature for each state also increasesand the AQI is also high.

2. What kind of relationship/impact does Population have on Electric Power generation and fuel consumption?

3. What kind of relationship/impact does Temperature have on Electric Power generation and fuel consumption?

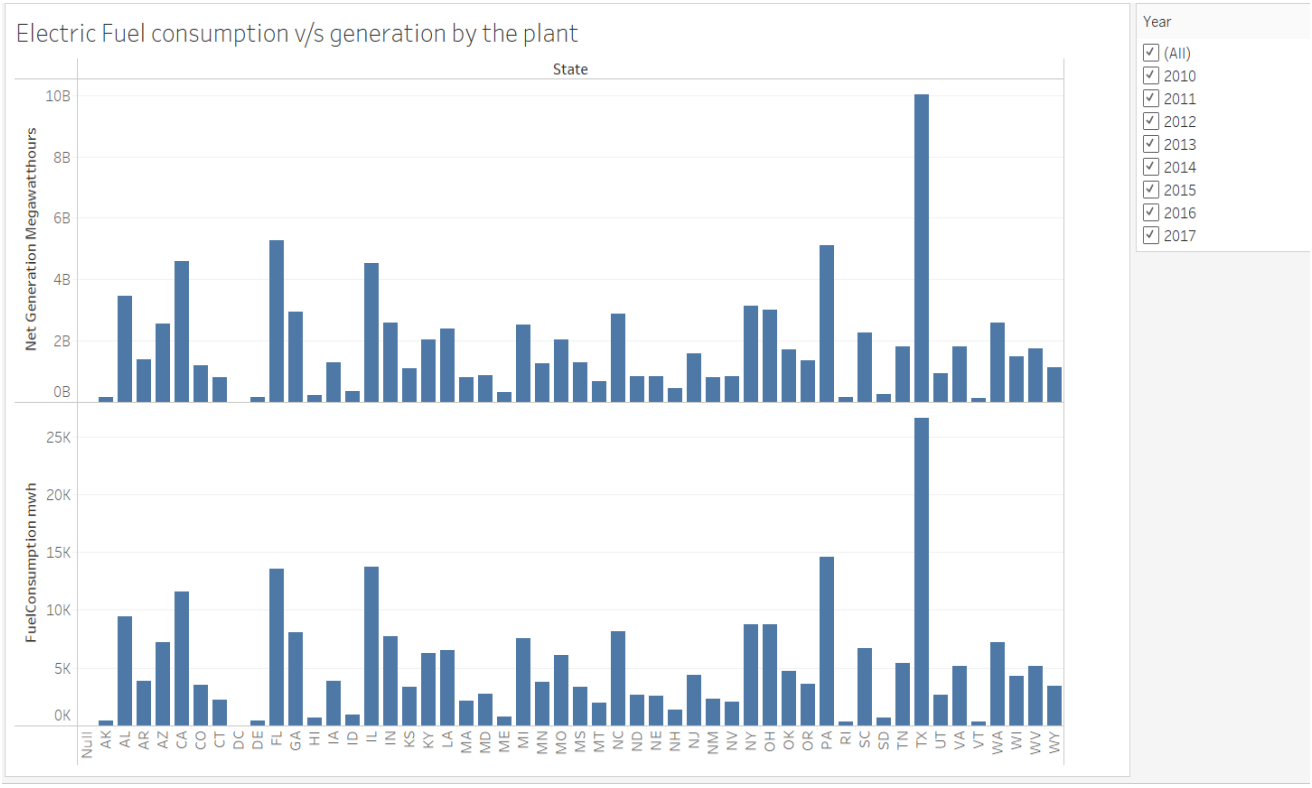
Questions 2 and 3 are answered below:
From year 2010 to 2016, we analyzed that there is not much change in population. As the population slightly changed, the electricity consumption reduced slightly(the changes were very minute). So, we

tried to analyse why it happened. This part is out of the scope of the project, but we found that in between these years, a few new technologies came up, example - computers started using less electricity etc. these small changes played a significant role in electricity utilization.

Same is the case observed with temperature, since there was a very slight change in population, it led to less pollution, and due to this the temperature remained mostly unchanged.

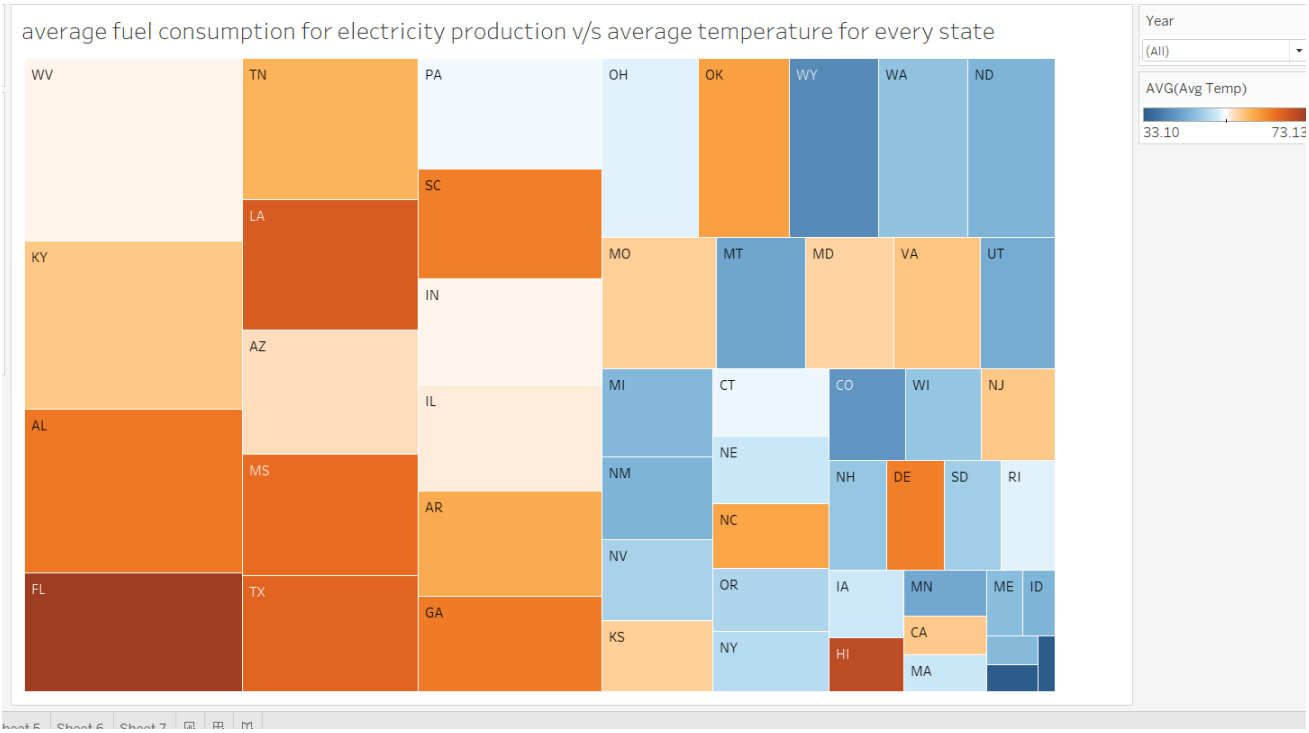
Visualizations:

1. Electricity generation v/s Consumption



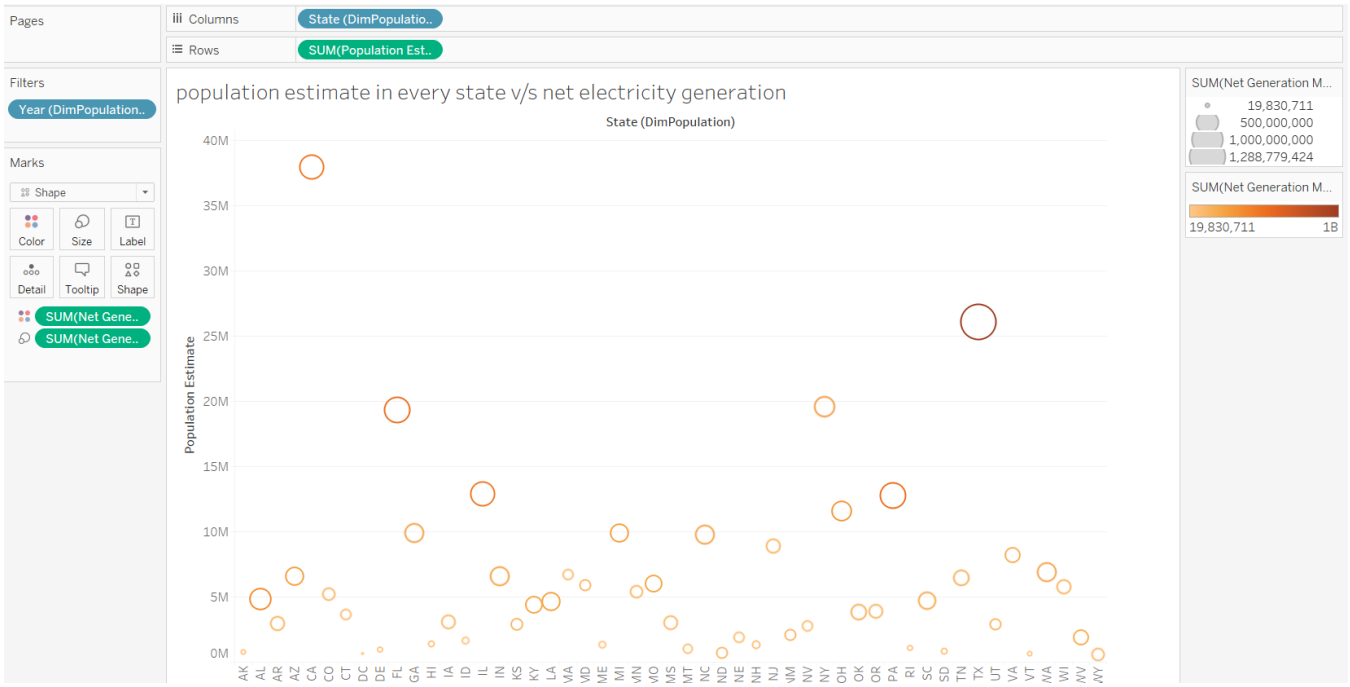
From the above graph about net annual generation and consumption, we can see that fuel consumption for electricity generation is in the range of thousands and we're getting generated electricity in very large amounts (billions). We noticed that both are directly proportional and are approximately in the same ratio.

2. Average fuel consumption for electricity production V/S Average temperature for every state



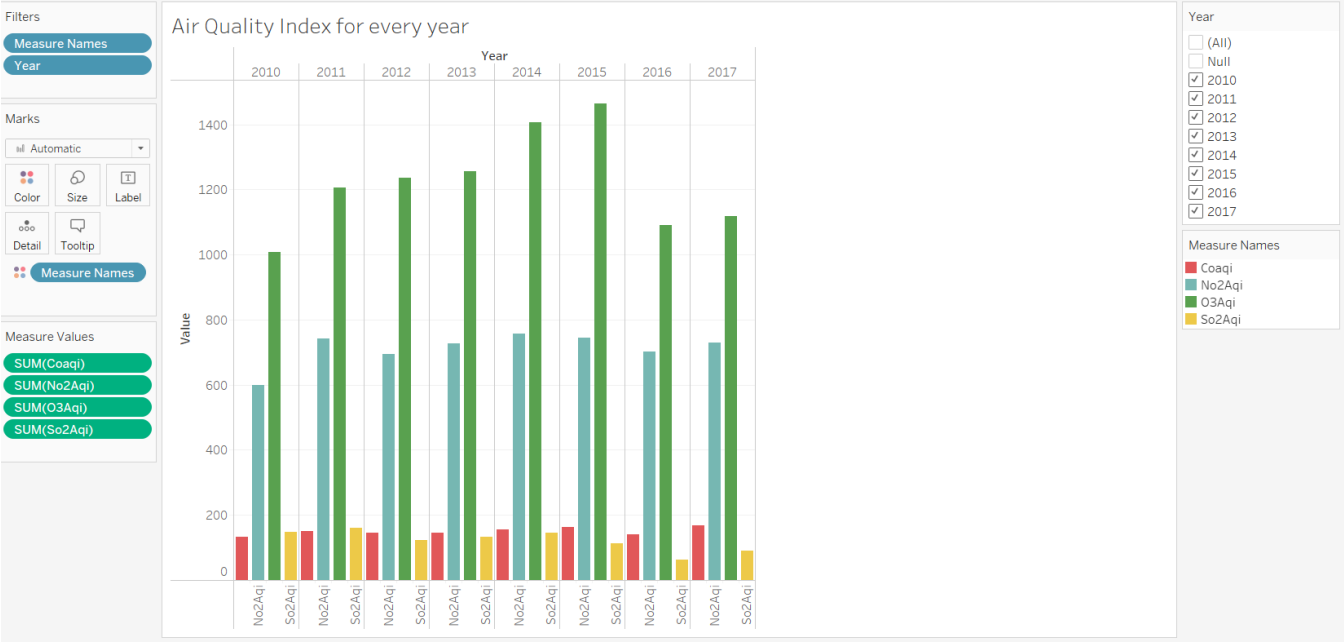
From the above plot, we can analyze average temperature by monitoring color shades in state blocks and size of the blocks specifies fuel consumption of every state. According to the graph, we infer that in which ever state the average fuel consumption is more, that state has a relatively high temperature and as consumption of fuel is decreasing.

3. Population vs Net Generation



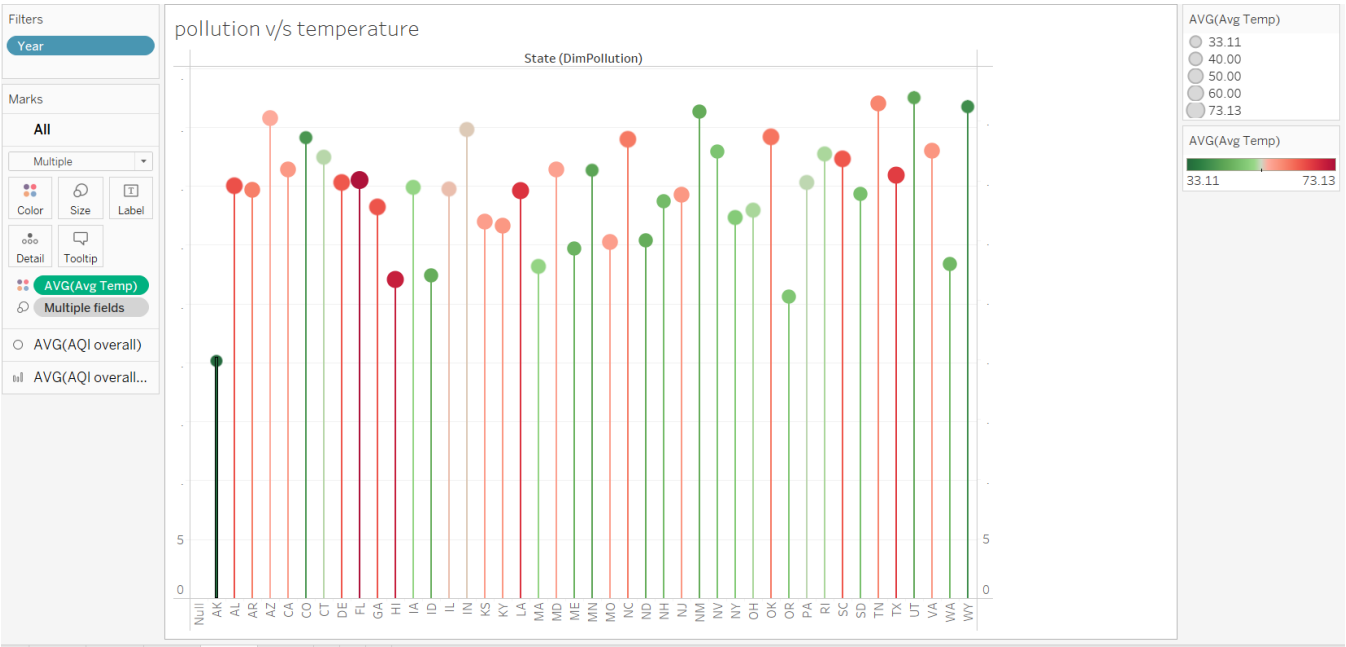
Above bubble chart shows how the population affects the electricity generation. In the graph, size and color of bubbles indicate the net electricity generation of each state. We can see that as the population in california is higher, electricity use is high and electricity generation ratio goes high because of high need.

4. Air quality analysis from every year



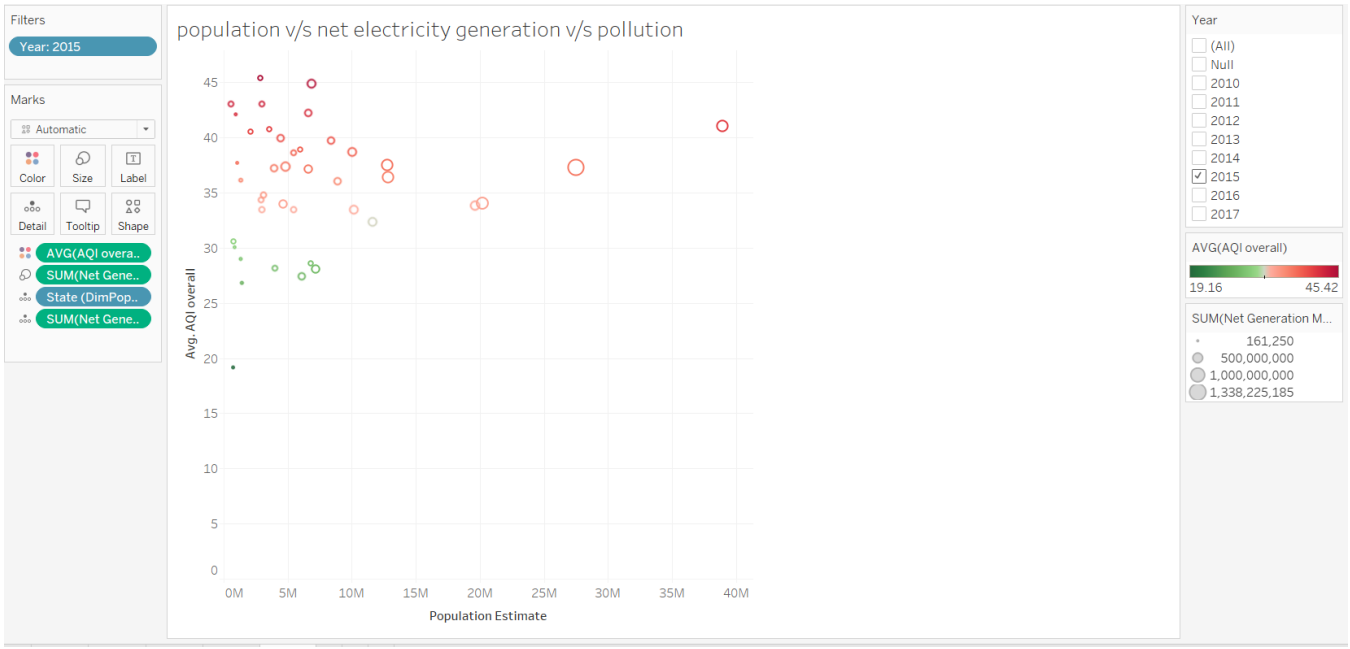
Above bar graph is presenting the average logged gasses released every year from the United states. We can notice that every year the portion of O3 is larger than the other three.

5. Pollution vs temperature



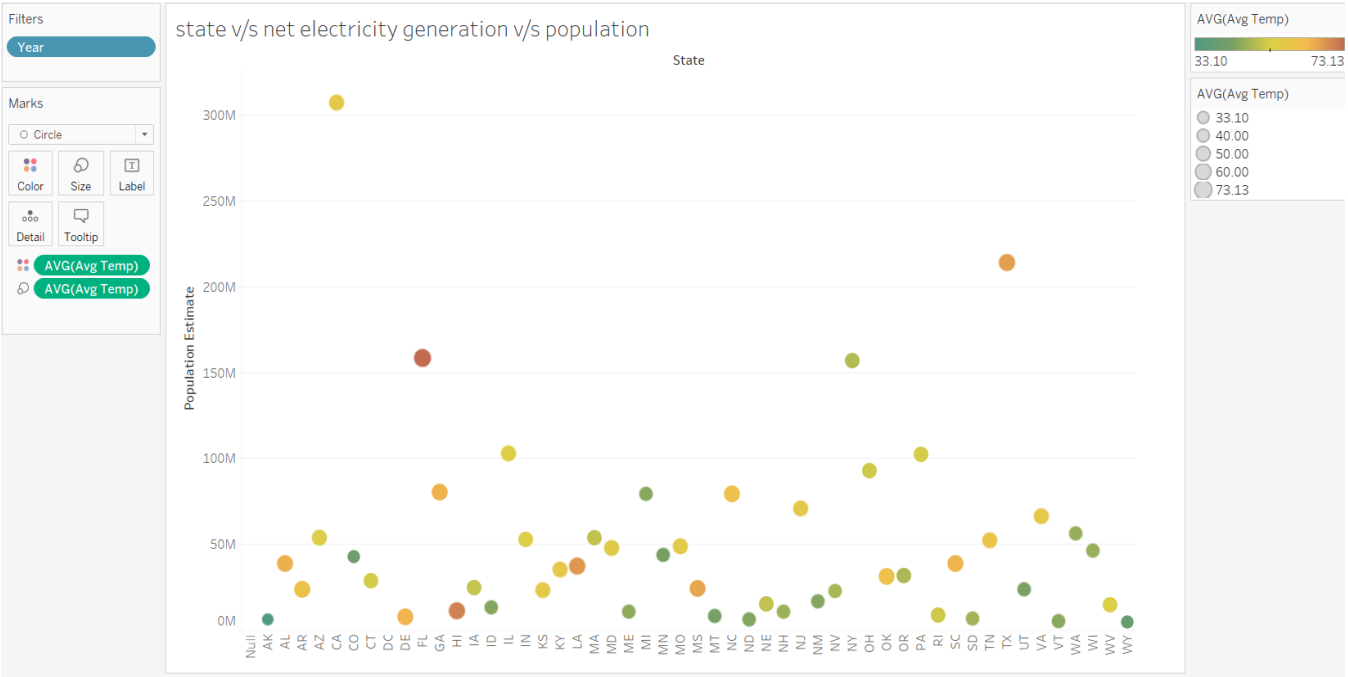
Above graph presents the analysis about how increasing pollution affects temperature as use of electricity, vehicles, and other gadgets increase and which directly and indirectly affect temperature. We can see that

6. Correlation between temperature, pollution,fuel consumption.



Above bubble graph is presenting the correlation between pollution,population and generation of electricity. We can analyze that whien the electricity generation is high, then the AQI is also high.

7. Population V/S Electricity Generation



From the above chart, we can visualize how population affects electricity generation and temperature. We can observe that as the population is more that 80M, the temperature is recorded above the average for each state.

Appendix:

Modification History:

Version	Date	Author	Description
1	03-23-2022	Team	Project Proposal
2	04-06-2022	Team	First Draft - Add Title Page, Table of Contents, Introduction, Data Sources, High Level Data Model, High Level Data Flow, Questions to Answer, Visualizations, and Appendix
3	04-20-2022	Team	Change Software Requirement, Add weather data as another source, Update Data Model, Data Transformation, and Questions to answer to reflect new weather data, Add Data Generation/Preparation section
4	04-21-2022	Team	Add Initial Load section, Update Introduction and Visualization to reflect new weather data, Listed other sections we need to work on with "To be continued…",
5	04-23-2022	Tomoki	Modify column names to have the same format (Camel case) and update the SSIS package accordingly, fix an issue with the Address column when loading the pollution dataset
6	04-24-2022	Tomoki	Update the SSIS package to transform and load data in staging tables into Data Warehouse for Weather, Population, and Pollution dimensions, Change the SCD setting,
7	04-26-2022	Tomoki	Update the SSIS package to transform and load data from staging tables into Data Warehouse for Facility dimension and Electric fact table, Implement SCD and incremental load, Add another data preprocessing step for Electric dataset, Generate 2017 Pollution and Weather data for incremental load
8	04-28-2022	Tomoki Bhavya	Update the report to reflect changes made from 04-23, 04-24, and 04-26 Work on building OLAP Cube

9	04-29-2022	Tomoki Bhavya	Update the report to reflect changes made from 04-23, 04-24, and 04-26 Work on building OLAP Cube
10	04-30-2022	Tomoki Bhavya Jigna & Bhavana	Update the report to reflect changes made from 04-23, 04-24, and 04-26 Work on building OLAP Cube Work on visualizations
11	05-01-2022	Tomok Bhavya Jigna & Bhavana	Update the "SSIS Implementation of Initial/Incremental Load and SCD" section in the report Work on building OLAP Cube Work on visualizations

Instructor Notes:

Date	Instructor Comment	Reponse
03-25-2022	good including weather might be interesting too	Since we already have three data sources and each of them requires multiple transformation tasks, we decided to stick to three data sources and not to include weather data.
04-09-2022	Use a lookup to convert the state, do not do it with a huge case statement. You don't have much data you should add a weather data set.	We have added a weather dataset as another source.
04-22-2022	Add lengths for data types Try to keep names consistent ... mixed case vs all caps Sometime underscore / camelcase Opetator_ID ...	We updated column names, data types, and dimensional models accordingly. We added explanations to screen shots.

	Column names with spaces Data types on dimensional model Cool - data creation Little more info with screen shots TBD to be filled.	