Commercial Building Stock Assessment (CBSA) 2025 User Guide

PREPARED BY:

Westat Inc.

August 2026

Northwest Energy Efficiency Alliance
700 NE Multnomah Street, Suite 1300
Portland, Oregon 97232
p 503.688.5400
neea.org
info@neea.org

Table of Contents

Introduction

The Commercial Building Stock Assessment (CBSA) collects, analyzes, and publishes information on the energy use of commercial buildings in the Northwest, including their characteristics and installed equipment. This information is used to advance strategies for a more efficient and resilient energy system in the Northwest by the Northwest Energy Efficiency Alliance (NEEA), its funders, and other regional stakeholders.

In 2024 and 2025, Westat and Michaels Energy, along with its subcontractor Driftless Energy, conducted data collection for the 2025 CBSA on behalf of NEEA. Using a sampling frame of 12,513 commercial buildings, this comprehensive regional study sampled 5,654 commercial and multi-family residential buildings across Idaho, Montana, Oregon, and Washington, encompassing 25 distinct primary building activity types. Each participating building was asked to complete a web survey, facilitate an in-person site visit for direct assessment of building systems and features, and authorize access to energy usage data from its utility provider(s). Over the course of data collection, 676 buildings completed web surveys, 568 buildings completed site visits, and utility providers gave energy usage data for 521 buildings. Buildings that submitted at least one type of data are included in the data tables for analysis. All buildings with estimated square footage of 1,000 square feet or greater are included in the Building Sampling Frame table.

This user guide provides support in using the 2025 CBSA database of 26 distinct data tables for analysis. It outlines the structure of the database, explains how to aggregate data, and provides guidance on applying statistical weights to generate representative estimates.

Utility of the 2025 CBSA Data

The 2025 CBSA data tables provide a comprehensive foundation for analyzing commercial and multi-family residential building stock across the Northwest at both regional and subregional levels. These data tables offer a snapshot of energy-related building characteristics and energy usage data, enabling users to

- **Estimate building stock characteristics** for diverse commercial spaces.
- **Assess energy-using equipment,** including lighting, appliances, and mechanical systems.

- **Analyze equipment prevalence and attributes,** such as fuel type, vintage, efficiency, and size.
- **Evaluate building envelope features,** including windows, walls, and insulation.
- **Assess energy consumption data** to support benchmarking, efficiency planning, and policy development.

By leveraging these capabilities, users can generate a wide variety of insights into the composition and performance of commercial buildings, informing energy efficiency strategies and regional planning efforts. For example, these data tables can be used to understand the distribution of refrigeration equipment by building size, heating and cooling equipment type by state, and energy usage by building activity.

When combined with data from prior CBSA studies, the data tables support the analysis of trends in building characteristics and energy usage. Evaluating changes over time may highlight market transformation successes or areas for opportunity in commercial and multi-family residential building energy usage. Some building types included in the 2025 CBSA were not part of the eligible sample for the 2019 CBSA; data users can focus on 2025 buildings that would also have been eligible in 2019 by using the variable *building_activity_eligible_in_cbsa2019* in the A_Summary_Building table. Additionally, the data dictionary maps common variables between the 2025 and 2019 CBSA studies to support the analysis of trends over this period.

The data tables include final weights as well as variables for variance estimation strata and variance estimation PSUs to reflect the complex sample design and the various weighting adjustments on standard errors (*varstrat* and *varpsu* for tables with participating buildings; *frm_varstrat* and *frm_varpsu* for the Building Sampling Frame table). In this document, the section titled Applying Weights and Estimating Variance provides example R and Python program code for generating common statistics.

2025 CBSA Database Overview

This section presents the overall structure of the 2025 CBSA database. It describes the unique data tables and how they are connected, with figures that illustrate these relationships. The section continues with information on how the data elements in each table are defined, including a description of the missing value codes applied when necessary. It ends with an example of how analysts can aggregate sub-building data to the building level, which is necessary to apply the building-level weights to produce weighted estimates.

The 2025 CBSA data are provided as a set of flat files with comma-separated values (CSV). Each CSV file corresponds to a single table, where columns contain data for attributes of a particular object, for example, the HVAC system. The tables together form a relational database that is the full set of 2025 CBSA data.

Each table in the 2025 CBSA database contains details for objects of the same type (e.g., water heaters, lighting fixtures), with columns within each table representing the attributes for that type of object. These unique tables are organized into groups of thematically related data, as summarized in Table 1.

Table 1. Groups of tables in database groups

Table group	Data tables in group
A_Summary	Building, HVAC, Lighting, Fans, Pumps
B_Building	Characteristics, Weights, Billing data
C_Building_Repeated_Rooms	Lodging
D_Envelope	Floors, Roofs, Walls, Windows
E_HVAC (including Hydronic Systems)	System, Ventilation, Boilers, Chillers, Heat rejections, Water heaters, Pumps
F_Refrigeration	Reach ins, Walk ins
G_Lighting	Fixtures
Building Sampling Frame	Building Sampling Frame

Within a building, collected data are stored in each of the relevant thematic tables accordingly. For example, characteristics of a building, such as windows, and particular objects, such as lighting fixtures, are each stored in separate tables. Tables can be linked using unique identifiers contained in each row. Each table includes a primary key for that table and foreign keys that can link to other tables, such as different types of light fixtures to buildings. Using lighting fixtures as an example, one can use the primary and foreign keys to compare lighting in several spaces in addition to an overall evaluation of the lighting in the building.

Note that data in a table may show inconsistencies between related variables, and the data has been left as collected in most instances. For example, on the topic of a building's energy sources, some questions were asked during the web survey and other data were collected in the site visit; a web survey respondent could select "No" for natural gas as an energy source, while the site visit data lists natural gas as heating fuel for HVAC. Data users can make independent decisions for how to handle inconsistencies in analysis, though if available, data from the more detailed site visit may be more reliable than web survey data reported by a building respondent.

Inconsistencies can arise between variables within the site visit data as well, for example, a lighting application that does not typically occur with the selected lamp type. Lighting application options were not limited by the prior lamp type selection. In this instance, the lamp type is likely to be the more accurate response because different lamp types are more visually distinct than lighting applications. For other examples of inconsistencies, data users may need to make their own judgments about which response type is more likely to be accurate.

A data dictionary in Microsoft Excel format provides additional information for each variable in the database, including the variable type, data type, associated question text or description, response options for closed-ended questions, and mapping to the 2019 CBSA data, where applicable.

Analytical Tables

The five analytical tables in table group A (A_Summary_Building, A_Summary_HVAC, A_Summary_Lighting, A_Summary_Fans, A_Summary_Pumps) provide foundational analyses and aggregated data for each building, highlighting key variables of interest. These files are in both CSV and XLSX format, providing flexibility for data users. The data dictionary offers more details for each column in these summary tables.

The A_Summary_Building table includes one record for each building in the study. This table provides overall information about the building, including the state and urbanicity category of its location, its use, and its overall features, such as the energy sources it uses. It also calculates numerical totals and percentages for building-level attributes such as area, perimeter, and energy consumption (both raw and weather normalized) by summing all underlying components. For HVAC systems (ventilation, heating, and cooling), both the predominant and secondary systems are described when applicable. The A_Summary_Building table can be linked to other tables using the *building_id* field.

The A_Summary_HVAC table lists one record for each HVAC system, meaning a building may have one row, multiple rows, or no rows, depending on the system(s) present. Many columns match those in the E_HVAC_System data table, and others provide aggregated information from related tables or calculated values based on HVAC data. The HVAC summary table can be linked to the E_HVAC_System table using the *hvac_system_id* field that appears on both tables.

The A_Summary_Lighting table includes one row for each fixture type within every indoor or outdoor lighting group. If a fixture type appears in two indoor groups and one outdoor group, it

will be listed as three separate entries, each with the corresponding details for that group. Similarly, if an indoor group contains two fixture types, the table will have one row for each type. The lighting summary table includes key variables from the G_Lighting_Fixtures table, sums of collected information, and building-level information from the building summary table. The lighting summary table can be joined to the G_Lighting_Fixtures table using the *lighting_fixtures_id* field.

The A_Summary_Fans table lists one record for each fan system identified in a participating building, meaning a building may have one row, multiple rows, or no rows, depending on the system(s) present. This table consolidates data across different system types (e.g., HVAC return, heat rejection, refrigerator walk-in) to allow for summarizing fan data across system types, including fan source, fan controls, motor type, and drive type. It also provides a summary assessment of the presence of a variable frequency drive (VFD) based on other available equipment information. The A_Summary_Fans table can be linked to other tables using the *building_id* field. Each fan can be linked to its original source file using the *fan_consolidated_source_id* field.

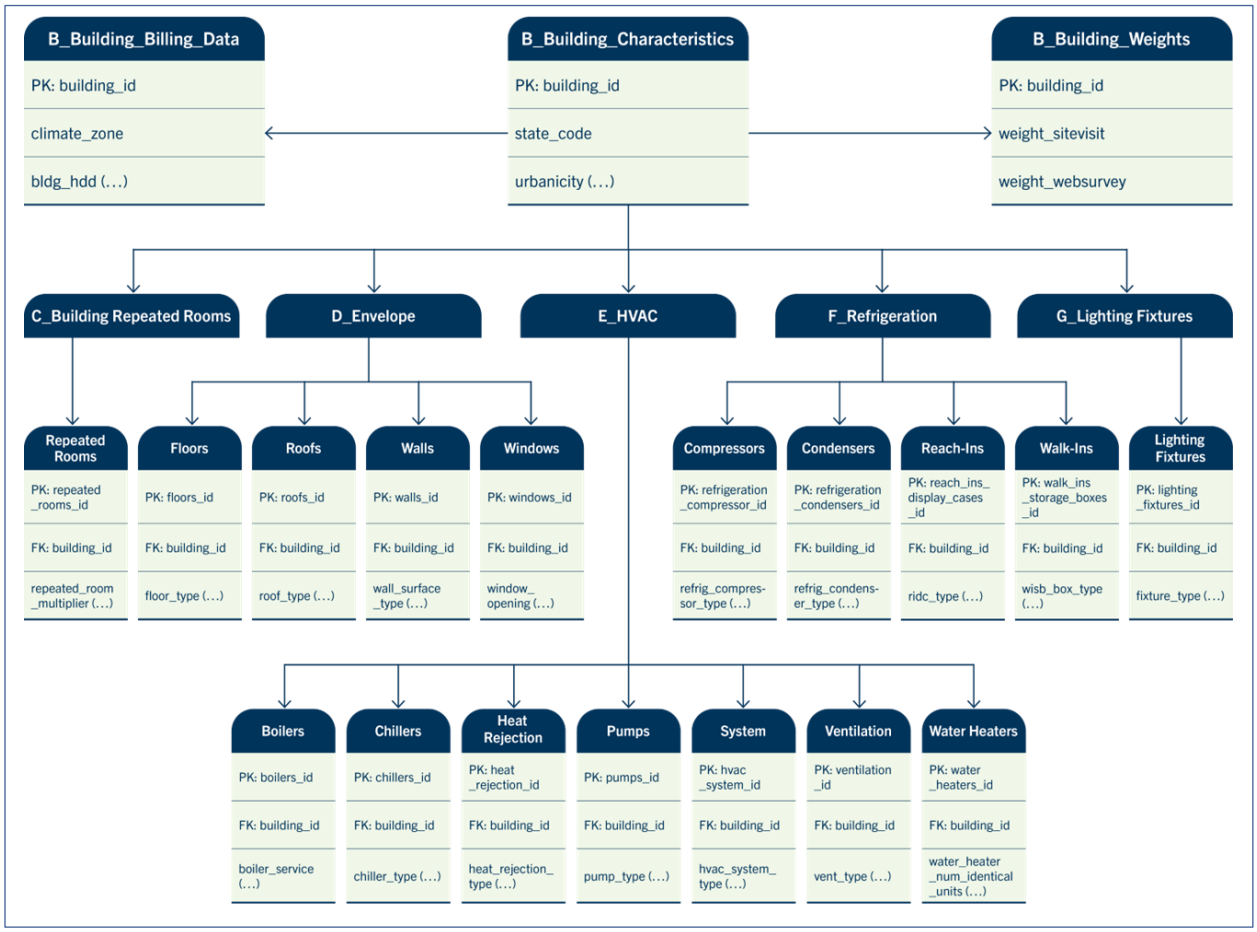
The A_Summary_Pumps table includes one row for each pump identified in a participating building, meaning a building may have one row, multiple rows, or no rows, depending on the system(s) present. This table consolidates data from the E_HVAC_Pumps and E_HVAC_Water Heaters tables to allow for summarizing pump data across system types, including pump type, motor power and efficiency, controls, and flow metrics. It also provides a summary assessment of the presence of a variable frequency drive (VFD) based on other available equipment information. The A_Summary_Pumps table can be linked to other tables using the *building_id* field. Each pump can be linked to its original source file using the *pump_consolidated_source_id* field.

CSV Data Tables

Figure 1 illustrates how the 20 CSV data tables are related to one another. Each table has a primary key, which is the unique ID value for that table. Many tables also have a foreign key, which links the table to other tables. For example, the principal table is B_Building_Characteristics, which has the primary key *building_id*, indicating the building as the unique ID value in this table. The tables B_Building_Billing_Data and B_Building_Weights also use this same primary key. The variable *building_id* then becomes the foreign key in the remaining tables, which each have their own primary key unique to the type of data presented in that table. For example, in the table C_Building_Repeated_Rooms, the primary key is

repeated_rooms_id, and the *building_id* is also included as a foreign key. The foreign key of *building_id* can be used to link tables to each other or to the B_Building_Characteristics table.

Figure 1. Table relationship diagram



Building Sampling Frame Table

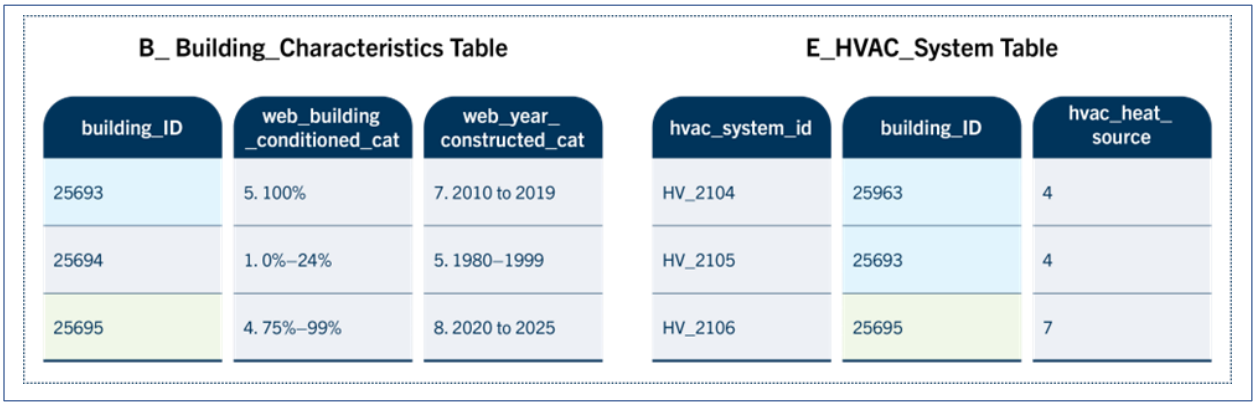
The Building Sampling Frame table is a standalone file that includes all commercial or multi-family residential buildings 1,000 square feet or larger in the 90 tracts sampled for the 2025 CBSA. Each record is one building identified through the virtual listing process. The table, including estimated building square footage, building activity, and floors above grade, can be used to understand the full range of commercial buildings identified in the sampled tracts. This file is in CSV format. The data dictionary offers more details for each column in these summary tables.

The data in the Building Sampling Frame table are based solely on the virtual listing process. Through the recruitment process, data were frequently updated to provide more accurate characterizations of the participating buildings. For example, building square footage in the analytic and all other CSV tables are based on consolidated information from virtual listing, web survey, and site visit activities. Additionally, while the Building Sampling Frame table includes all buildings with eligible building activities based on virtual listing, some buildings were determined to be ineligible based on building use, e.g., industrial and agricultural buildings. All buildings over 1,000 square feet are included in the Building Sampling Frame table, even if they were later determined to be ineligible.

Relationships Between Tables

Tables are organized with one-to-many relationships, which can also encompass one-to-none and one-to-one. With this relationship, there is a table in which a primary key appears only once, and it relates to another table in which that key can appear not at all, once, or multiple times. For example, each building is listed only once in the B_Building_Characteristics table, but it would appear on the E_HVAC_System table once for each HVAC system the facility has. This varies by the objects, equipment, and systems in each building. Figure 2 demonstrates this one-to-many relationship between the B_Building_Characteristics table and the E_HVAC_System table.

Figure 2. One-to-many relationship



B_ Building_Characteristics Table

building_ID	web_building _conditioned_cat	web_year_ constructed_cat
25693	5. 100%	7. 2010 to 2019
25694	1. 0%–24%	5. 1980–1999
25695	4. 75%–99%	8. 2020 to 2025

E_HVAC_System Table

hvac_system_id	building_ID	hvac_heat_ source
HV_2104	25963	4
HV_2105	25693	4
HV_2106	25695	7

Figure 2 demonstrates three buildings, with IDs 25693, 25694, and 25695. Building 25693 has two HVAC systems, with IDs HV_2104 and HV_2105, so it is listed once on the B_Building_Characteristics table and twice on the E_HVAC_System table. Building 25694 does not have an HVAC system, so it is listed once on the B_Building_Characteristics table but does

not appear on the E_HVAC_System table. Building 25695 has one HVAC system, so it is listed once on each table. The colors highlight these different types of relationships between the B_Building_Characteristics table and the E_HVAC_System table.

The Building Sampling Frame table is not intended to be connected with the analytic or other CSV tables.

Missing Value Codes

When data were not obtained for a particular variable, it may be because the question was not relevant to collect (e.g., subsequent HVAC questions if the building does not have HVAC), because a field technician was unable to access the space to determine the response, or because the field technician was otherwise unable to determine the correct response. Missing value codes are included in the data within the same variable as responses given.

As detailed in the data dictionary, there are four missing value codes:

- **-9: Could Not Assess/Not Ascertained**. In the site visit instrument, field technicians were asked to indicate why a question was left blank, and this code reflects their response. In the web survey instrument, this code was applied if a respondent did not give an answer to a provided question.
- **-8: Don't Know**. This value occurs when "Don't know" was an available response option to the question and was selected by the respondent or the field technician.
- **-7: Valid Skip**. This value was applied when a question was never shown to the respondent or the field technician because of a skip pattern in the instrument.
- **-6: Full Instrument Nonresponse**. Some tables include data from both the web survey and the site visit. If a building completed only one of those instruments, it was included on the table, but any variables from the other instrument had this value applied.

For some missing data related to energy use intensity (EUI) and lighting wattage, values have been imputed, with a flag added in a separate variable. For other missing data, users can consider conducting additional imputation. Possible approaches to imputation include multiple imputation by chained equations (MICE), which can be implemented by the MICE R package or SAS's PROC MI, sequential hot-deck, or mean/median imputation.

Aggregating Data to the Building Level

Users can aggregate sub-building-level data from multiple tables to the building level by using the primary and foreign keys in each table. This section provides examples for aggregating

unweighted data to the building level. Once at the building level, analysts can apply one of the two building-level weights, along with the variance estimation stratum code (*varstrat)* and the variance estimation cluster code (*varpsu*), to a building-level file using their preferred statistical software to generate weighted estimates and appropriate standard errors for desired statistics such as means, proportions, totals, and ratios. This analysis can be performed not only at the building level but also across many domains, such as state, county, building size, and others. The Estimation Approach section provides additional information on steps for applying the building-level weights to generate weighted estimates and standard errors.

Much of the data collected are already provided at the building level. But in some tables, the data are also collected at the sub-building level and need to be aggregated to the building level to appropriately apply the building-level weights. For example, to calculate the total number of light fixtures within a building, users can add the counts of all types of light fixtures within a *building_id*. Tables 2 and 3 provide an example of this calculation.

In the examples in Tables 2 and 3, each lighting fixture has a unique ID. To appropriately apply the building-level weights, all lighting fixtures associated with the same *building_id* should be summed to obtain the total number of lighting fixtures in the building, regardless of the location or lamp technology. Users should exclude missing values from the aggregation process.

Table 2. Sample light fixture data

Lighting_fixtures_id	*Building_id*	*Fixture_space_type*	*Number_identical_fixtures*	*Lamp_technology*
LF_8097	25693	Public Access – Gym	53	LED Tube
LF_8098	25694	Exterior Walkways	10	Halogen
LF_8099	25694	Exterior Walkways	5	Incandescent
LF_8100	25695	Living Quarters – Dwelling Unit	6	LED Screw
LF_8101	25695	Living Quarters – Guest Room	4	LED Screw
LF_8102	25695	Exterior Walkways	-9	Incandescent

Table 3. Sum of total light fixtures by building from sample light fixture data

Building_id	Sum of total light fixtures
25693	53=53
25694	10+5=15
25695	6+4=10

Estimation Approach

This section explains how to apply the weights and variance estimation codes developed for the 2025 CBSA to generate weighted estimates that represent the target population. It summarizes the sampling approaches that form the basis for these weights and provides step-by-step guidance on using them to estimate building and regional values from the sampled data.

Case Weight Development

Each building that completed the web survey has a weight (*weight_websurvey*), and each building that completed the site visit has a weight (*weight_sitevisit*). Each of these weights uses the inverse of the building's probability of selection into the sample as a starting point and then adjusts these values to account for nonresponse to the web survey and the site visit, respectively. These nonresponse adjustments help reduce the potential for nonresponse bias and ensure the data are representative of the target population. The Building Sampling Frame table also includes a tract-level weight (*weight_frm*) to reflect the target population. The 2025 CBSA Final Report describes the process of developing the weights in more detail.

Within-Site Sampling

All collected data are grouped under a building, so building is the highest level of organization in the database. In some instances, details about all spaces of a building could not be documented during a site visit because of practical limitations, such as access or insufficient time to visit all floors of a building. If data were not obtained for the full building—for example, for only two floors instead of all floors—the collected data have been adjusted accordingly based on within-site sampling. For the 2025 CBSA, counts reflected in the tables are already estimated at the building level, so no separate weights are necessary to create estimates at the building level following within-site sampling.[1] Buildings where within-site sampling was implemented are noted in the variable *subsampled_flag* in the table B_Building_Characteristics.

[1] The within-site sampling was not implemented correctly for building_ids 25208, 25616, and 25621. Where possible, the data collected after subsampling were integrated with the building-level data, but some records were omitted as nonrepresentative of the building.

Applying Weights and Estimating Variance

The data tables provide variables for the convenience of estimating weighted totals for lamps, wattage, and square footage. These variables' names are identified with the format ***variablename_represented***. However, these variables do not allow users to estimate variance and calculate confidence intervals for weighted totals. The steps outlined here guide users on how to appropriately calculate weighted estimates for proportions, means, ratios, and totals and estimate variance in a way that accounts for the complex sample design.

The B_Building_Weights and Building Sampling Frame tables include variables to obtain weighted estimates and implement a Taylor-series approach to estimate standard errors for weighted survey estimates. A variety of software packages, including R and Python, can apply the Taylor-series linearization method. In the B_Building_Weights table, the variables that jointly reflect the CBSA survey design include the site visit weight (*weight_sitevisit*), the web survey weight (*weight_websurvey*), the variance estimation stratum code (*varstrat*), and the variance estimation cluster code (*varpsu*) required by the variance estimation programs.[2] To obtain weighted estimates and standard errors from data in the Building Sampling Frame table that appropriately account for the complex sample design used to select the 90 Census tracts that were virtually listed, researchers should use the weight (weight_frm), variance estimation stratum code (frm_varstrat), and variance estimation cluster code (frm_varpsu) variables in that table.

When calculating weighted estimates, analysts should apply either the web survey weight or the site visit weight, not both. The weight variable to use will depend on the table and the variables of interest within the table. The variables in all but three tables (A_Summary_Building, B_Building_Billing_Data, B_Building_Characteristics) were collected during the site visit. Therefore, the site visit weight should be used to calculate weighted estimates from all variables except ones in those three tables.

For the tables A_Summary_Building, B_Building_Billing_Data, and B_Building_Characteristics, nearly all variables are constructed variables that contain data from the site visit, if completed, and otherwise contain data from the web survey. For example, the variable *building_sf* on the A_Summary_Building table uses the building square footage captured during the site visit if the building completed a site visit. Otherwise, the *building_sf* variable contains the square footage reported by the respondent during the web survey. A total of 552 buildings completed both a site

[2] The creation of the final building weights, the variance estimation stratum codes, and the variance estimation cluster codes are described in detail in the 2025 CBSA Final Report.

visit and a web survey and have weights for both. Sixteen buildings completed a site visit but not a web survey, and 124 buildings completed a web survey but not a site visit. In these situations, the buildings have a weight only for the portion they completed. Thus, for the constructed variables in these three tables, we generally recommend using the web survey weight to maximize the number of buildings included in the analysis. Table 4 provides specific reasons for this recommendation for each of the three tables. Note that for certain domains (i.e., subgroups of interest), using the site visit weight may produce the same or higher number of buildings with nonmissing data that can be included in the analysis compared to the web survey weight. In those instances, analysts will want to consider using the site visit weight instead.

In summary, the site visit weight will be applicable for most tables. The web survey weight will typically be most appropriate for the tables A_Summary_Building, B_Building_Billing_Data, and B_Building_Characteristics, though for certain domains, the site visit weight will produce more buildings with nonmissing data and should be used in those situations.

Table 4. Reasoning for specific uses of the web survey weight

Summary table	Reason for generally recommending the use of web survey weight
A_Summary_Building	Nearly all variables in this table were constructed using data recorded in both the site visit and the web survey. These analytical variables contain data recorded from the site visit if available and contain responses from the web survey if the building did not complete the site visit.
B_Building_Characteristics	Most variables in this table were constructed in the same manner as the analytical variables in the A_Summary_Building table, using data recorded both in the site visit and the web survey. The variables starting with the variable name “web” only contain data recorded from the web survey. The web survey weight should be used to produce weighted estimates for all variables whose variable name starts with “web.”
B_Building_Billing_Data	This table contains billing data obtained from utility companies with permission from buildings that completed either the site visit or the web survey. Billing data were obtained from a substantial number of the 124 buildings that completed a web survey but not a site visit.

R and Python Code for Joining Weighting and Variance Estimation Variables to Building-Level Files

This section contains example code with placeholders for maximum flexibility in demonstrating key applications in data analysis. The specific R code developed to produce weighted estimates for the 2025 CBSA Final Report is also available for reference and was delivered together with the final report in the compressed folder *2025_CBSA_table_syntax*. This code provides details about category mappings applied in analysis, such as for HVAC system types and primary building activities and contains additional examples for analyses of interest. See Appendix 2 of

the 2025 CBSA Final Report for discussion of these category mappings, and Appendix 1 for table output corresponding to the provided R code.

In these sections with example code, text in `Courier New` represents blocks of example code, R function names, Python object names, and R and Python package names. Text in *`italicized Courier New`* represents specific variable names, and text in ***`bold and italicized Courier New`*** represents placeholders for specific data table, variable, or output names to be added by the user. For example, all instances of ***`cat_var`*** would be updated with the variable name of a specific categorical variable being used in the analysis, and ***`con_var`*** would be updated with the variable name of a specific continuous variable being used in the analysis.

To generate weighted estimates and implement a Taylor-series approach to estimate standard errors, users need one of the building-level weights (*weight_sitevisit* or *weight_websurvey*), the variance estimation stratum code (*varstrat*), and the variance estimation cluster code (*varpsu*). These variables, along with *building_id*, are on the B_Building_Weights table. For example, if users want to generate a weighted estimate based on a variable in the B_Building_Characteristics data table, they need to join the building level weights, the *varpsu* variable, and the *varstrat* variable to the B_Building_Characteristics data table as follows:

R code:

```
Install.packages("tidyverse")
Library(tidyverse)
B_Building_Characteristics |>
     left_join(B_Building_Weights, by = "building_ID")
```

Python code:

```
import pandas as pd
pd.merge(B_Building_Characteristics, B_Building_Weights,
on="building_ID", how="left")
```

Due to the standalone nature of the Building Sampling Frame table, no joins are required before calculating weighted estimates.

R and Python Code for Calculating Weighted Estimates and Standard Errors

This section contains example R and Python program code for use with the 2025 CBSA data. It provides example code for generating popular statistics and standard errors, as well as example code for calculating and specifying the number of degrees of freedom for testing the significance of estimates and constructing confidence intervals. These examples are not meant to be exhaustive or provide instruction for users unfamiliar with a particular software package.

The following examples use the site visit weight (*weight_sitevisit)*. However, as noted in the previous section, for the majority of variables in the A_Summary_Building, B_Building_Billing_Data, and B_Building_Characteristics data tables, we generally recommend using the web survey weight (*weight_websurvey*) instead to maximize the number of cases in the analysis. Similarly, any analyses using the Building Sampling Frame table should use the frame weight (*weight_frm*) and the appropriate variance estimation stratum and cluster codes (*frm_varstrat* and *frm_varpsu*).

R code:

First, we specify a survey design object using the `svydesign()` function in the `survey` package. To do so, we need to supply the weight, variance estimation cluster ID variable, and variance estimation stratum ID variable.[3]

```
des <- svydesign(
 data = analysis_data,
 id = ~varpsu,
 strata = ~varstrat,
 weights = ~weight_sitevisit,
 nest = TRUE
 )
```

The following code creates estimates of the population proportions for the categorical variable ***cat_var,*** along with standard errors of the estimates.

[3] The 2019 CBSA public use database does not provide variance estimation cluster ID or stratum ID variables. When generating weighted estimates and standard errors in R with the 2019 data, users should modify the R survey design object code to set id = ~1 and remove the "strata =" specification.

```
svymean(~factor(cat_var), design = des, na.rm = T)
```

The following code creates estimates of the population mean for the continuous variable ***con_var,*** along with standard errors of the estimates.

```
svymean(~con_var, design = des, na.rm = T)
```

The following code creates estimates of the population total for the continuous variable ***con_var,*** along with standard errors of the estimates.

```
svytotal(~con_var, design = des, na.rm = T)
```

The following code creates estimates for a ratio with a defined continuous numerator ***num_var*** and continuous denominator ***denom_var,*** along with standard errors of the estimates.

```
svyratio(~num_var, ~denom_var, design = des, na.rm = T)
```

For confidence intervals on proportions, we recommend using Wilson intervals. These can be created by the `svyciprop()` function in version 4.5 or later of the `survey` package.[4] The `svyciprop()` function calculates confidence intervals for one level of a categorical variable at a time. If ***cat_var*** has three levels (values = 1, 2, and 3), the following statements output the confidence intervals for each level:

```
svyciprop(~I(cat_var ==1), des, method = "wilson", level = 0.9)
svyciprop(~I(cat_var ==2), des, method = "wilson", level = 0.9)
svyciprop(~I(cat_var ==3), des, method = "wilson", level = 0.9)
```

For confidence intervals on means and totals of continuous variables, we recommend using Wald intervals. These can be created by the `confint()` function in base R.

```
confint(svymean(~con_var, design = des, na.rm=T), level = 0.9)
```

[4] Note that version 4.5 of the survey package is not yet available on CRAN as of December 2025. The authors of the package feel it is stable and will be available on CRAN in the near future. In the meantime, version 4.5 can be installed from R-Forge using the following code: install.packages("survey", repos="http://R-Forge.R-project.org"). For analysts who prefer to wait until version 4.5 is available on CRAN before using, replace "wilson" with "xl", which calculates logit transformed confidence intervals.

```
confint(svytotal(~con_var, design = des, na.rm=T), level = 0.9)
```

To create these estimates for each of the levels of a domain variable ***domain_var***, the following code may be used:

```
svyby(~factor(cat_var), ~domain_var, svymean, design = des, na.rm = T)
svyby(~con_var, ~domain_var, svymean, design = des, na.rm = T)
svyby(~con_var, ~domain_var, svytotal, design = des, na.rm = T)
svyby(formula = ~num_var, denominator = ~denom_var, by = ~domain_var,
design = des, FUN = svyratio, na.rm = T)
```

The following code creates confidence intervals on population proportions for each level of the categorical variable ***cat_var*** for the first level of a domain variable, ***domain_var*** = 1. Note that the `subset()` function, when used on a survey design object, restricts the survey design to a subpopulation of interest while retaining all information about the original design. That information is then used to calculate appropriate domain standard errors.[5]

```
des1 <- subset(des, domain_var == 1)
svyciprop(~I(cat_var ==1), des1, method = "wilson", level = 0.9)
svyciprop(~I(cat_var ==2), des1, method = "wilson", level = 0.9)
svyciprop(~I(cat_var ==3), des1, method = "wilson", level = 0.9)
```

The following code creates a confidence interval on the estimate of the mean and the total of ***con_var*** for the first level of the domain variable, ***domain_var*** = 1. The degrees of freedom within a domain can be examined by supplying the subsetted survey design object to the `degf()` function. Analysts should proceed with caution if a domain results in fewer than 8 degrees of freedom. Please review the Considerations in Conducting Domain Analyses section for further information.

```
confint(svymean(~con_var, design = des1, na.rm=T), level = 0.9, df =
degf(des1))
```

[5] For more information, please see https://www.rdocumentation.org/packages/survey/versions/4.4-8/topics/subset.survey.design.

```
confint(svytotal(~con_var, design = des1, na.rm=T), level = 0.9, df =
degf(des1))
```

Python code:

This Python code relies on the published `samplics` package in Python (Diallo, 2021). First, we import objects from the `samplics` package.

```
from samplics.estimation.expansion import TaylorEstimator
from samplics.utils.types import PopParam, SinglePSUEst
from samplics.categorical import Tabulation, CrossTabulation
```

The user interface of the `samplics` package differs substantially from other software, such as the `survey` package in R. For each estimate to be produced, the user must create an estimator object and specify the type of parameter to be estimated (means, totals, proportions, etc.). The survey design information must be supplied to the estimator when calling the estimator's `estimate()` method. The following example code creates estimates of a population mean for a continuous variable named ***con_var***:

```
con_var_mean = TaylorEstimator(PopParam.mean)
con_var_mean.estimate(
 y = analysis_data['con_var'],
 samp_weight = analysis_data['weight_sitevisit'],
 psu = analysis_data['varpsu'],
 stratum = analysis_data['varstrat'],
 single_psu = SinglePSUEst.skip,
 remove_nan = True
)
print(con_var_mean)
```

The estimates are automatically accompanied by a standard error estimate and a confidence interval. To change the confidence level from its default value of 95 percent, the user must specify a value of `alpha` when creating the estimator object. The following example code

estimates a population total, its standard error, and a 90 percent confidence interval for the continuous variable ***con_var***:

```
con_var_total = TaylorEstimator(PopParam.total, alpha = 0.1)
con_var_total.estimate(
      y = analysis_data['con_var'],
      samp_weight = analysis_data['weight_sitevisit'],
      psu = analysis_data['varpsu'],
      stratum = analysis_data['varstrat'],
      single_psu = SinglePSUEst.skip,
      remove_nan = True
)
print(con_var_total)
```

Similarly, the following example code estimates a population ratio, its standard error, and a 90 percent confidence interval:

```
ratio = TaylorEstimator(PopParam.ratio, alpha = 0.1)
ratio.estimate(
      y=analysis_data['num_var'],
      x=analysis_data['denom_var'],
      samp_weight=analysis_data['weight_sitevisit'],
      psu = analysis_data['varpsu'],
      stratum=analysis_data['varstrat'],
      single_psu = SinglePSUEst.skip,
      remove_nan = True
)
print(ratio)
```

To estimate population proportions for a categorical variable, the `Tabulation()` object can be used. The following code estimates proportions for a categorical variable named ***cat_var***:

```
cat_var_prop = Tabulation(PopParam.prop)
cat_var_prop.tabulate(
      vars = analysis_data['cat_var'],
      samp_weight = analysis_data['weight_sitevisit'],
      psu = analysis_data['varpsu'],
      stratum = analysis_data['varstrat'],
      single_psu = SinglePSUEst.skip,
      remove_nan = True
)
print(cat_var_prop)
```

For more complex tables, the user can replace the `Tabulation()` object with `CrossTabulation()`, as in the following example code, which estimates proportions for combinations of two categorical variables, ***cat_var_1*** and ***cat_var_2***:

```
two_way_proportions = CrossTabulation(PopParam.prop)
two_way_proportions.tabulate(
      vars = analysis_data[['cat_var_1', 'cat_var_2']],
      samp_weight = analysis_data['weight_sitevisit'],
      psu = analysis_data['varpsu'],
      stratum = analysis_data['varstrat'],
      single_psu = SinglePSUEst.skip,
      remove_nan = True
)
print(two_way_proportions)
```

For confidence intervals on proportions, we recommend using Wilson intervals. Unfortunately, this confidence interval method is not available in the Python `samplics` package. Instead, the package by default uses a method referred to as the `logit` method when estimating proportions. This method is referred to in the R `survey` package as the `xlogit` method. For

confidence intervals on means and totals of continuous variables, we recommend using Wald intervals, which the `samplics` package automatically provides when estimating means and totals.

To estimate means or totals for each level of a domain variable ***domain_var***, the `domain` argument can be used. The following example code estimates means for each category of the variable ***domain_var***:

```
con_var_mean_by_group = TaylorEstimator(PopParam.mean)
con_var_mean_by_group.estimate(
      y = analysis_data['con_var'],
      domain = analysis_data['domain_var'],
      samp_weight = analysis_data['weight_sitevisit'],
      psu = analysis_data['varpsu'],
      stratum = analysis_data['varstrat'],
      single_psu = SinglePSUEst.skip,
      remove_nan = True
)
print(con_var_mean_by_group)
```

The `domain` argument should also be used when producing estimates for a subset of the data. Users should not manually subset the data table before calling a function from `samplics` because this can produce invalid standard error estimates. Instead, the user should create a variable that identifies the subset of interest and supply that variable to the `domain` argument. The following example code produces an estimate for a subset of interest, relying on the variable named ***is_in_subset_of_interest***. Analysts should proceed with caution if a domain results in fewer than 8 degrees of freedom. Please review the Considerations in Conducting Domain Analyses section for further information.

```
con_var_mean_in_subset = TaylorEstimator(PopParam.mean)
con_var_mean_in_subset.estimate(
      y = analysis_data['con_var'],
      domain = analysis_data['is_in_subset_of_interest'],
```

```
        samp_weight = analysis_data['weight_sitevisit'],
        psu = analysis_data['varpsu'],
        stratum = analysis_data['varstrat'],
        single_psu = SinglePSUEst.skip,
        remove_nan = True
)
print(con_var_mean_in_subset)
```

Considerations in Conducting Domain Analyses

When conducting domain (i.e., subgroup) analyses, special attention should be given to two items, both of which concern the variance estimates for the domain, which, in turn, affect the width of the estimated confidence interval.

1. Analysts should not manually subset the data to keep only cases in the domain(s) of interest before generating estimates because this can produce invalid variance estimates. Instead, users should create a variable that identifies the subset of interest and supply that variable to the appropriate domain function or argument in their preferred statistical package. Using the subset() function on a survey design object in R, as shown in the Applying Weights and Estimating Variance section, is a valid way to perform domain analysis because subset.survey.design maintains information about the original design, which is then used in variance estimation.
2. Analysts should examine whether the data support reliable estimates for domains of interest. Variance estimates for small domains may be unstable, and the instability is reflected in the degrees of freedom. For a domain analysis, the degrees of freedom within a domain is equal to the number of unique variance PSUs in that domain minus the number of unique variance strata in that domain (Korn & Graubard, 2011). Statistical software packages such as R or Python will compute and provide the appropriate degrees of freedom for each domain analysis. Small degrees of freedom generally result in a wide confidence interval, and estimates based on fewer than 8 degrees of freedom have a relative standard error of the variance approximately 50 percent or higher.[6] For this reason, researchers should investigate domain-level degrees of freedom for all

[6] This is calculated based on the approximation given in Valliant and Rust (2010) $RSE\left(var\left(\hat{\theta}\right)\right) = 100\sqrt{2/df}$.

domain analyses and proceed with caution if any of the domains result in fewer than 8 degrees of freedom.

References

Diallo, M. S. (2021). samplics: A Python Package for selecting, weighting and analyzing data from complex sampling designs. *Journal of Open Source Software, 6*(68), 3376. https://doi.org/10.21105/joss.03376

Korn, E. L., & Graubard, B. I. (2011). *Analysis of health surveys.* Wiley.

Valliant, R., & Rust, K. F. (2010). Degrees of freedom approximations and rules-of-thumb. *Journal of Official Statistics, 26*(4), 585–602.