Data Transformation Overview. Data transformation is the **process of converting data from one format to another**. The most common data transformations are converting raw data into a clean and usable form, converting data types, removing duplicate data, and enriching the data to benefit an organization.

**What is data transformation?**

Data transformation is the process of changing the format, structure, or values of data. For data analytics projects, data may be transformed at two stages of the data pipeline. Organizations that use on-premises data warehouses generally use an ETL ([**extract, transform, load**](https://www.stitchdata.com/resources/glossary/etl/)) process, in which [**data transformation is the middle step**](https://www.stitchdata.com/etldatabase/etl-transform/). Today, most organizations use cloud-based data warehouses, which can scale compute and storage resources with latency measured in seconds or minutes. The scalability of the cloud platform lets organizations skip preload transformations and load raw data into the data warehouse, then transform it at query time — a model called ELT ( [**extract, load, transform)**](https://www.stitchdata.com/resources/what-is-elt/).

Processes such as [**data integration**](https://www.stitchdata.com/resources/glossary/data-integration/), [**data migration**](https://www.stitchdata.com/resources/glossary/data-migration/), [**data warehousing**](https://www.stitchdata.com/resources/glossary/data-warehouse/), and [**data wrangling**](https://www.stitchdata.com/resources/glossary/data-wrangling/) all may involve data transformation.

Data transformation may be constructive (adding, copying, and replicating data), destructive (deleting fields and records), aesthetic (standardizing salutations or street names), or structural (renaming, moving, and combining columns in a database).

An enterprise can choose among a variety of [**ETL tools**](https://www.stitchdata.com/etldatabase/etl-tools/) that automate the process of data transformation. Data analysts, data engineers, and data scientists also transform data using [**scripting languages such as Python**](https://towardsdatascience.com/python-data-transformation-tools-for-etl-2cb20d76fcd0) or [**domain-specific languages like SQL**](https://towardsdatascience.com/python-vs-sql-comparison-for-data-pipelines-8ca727b34032).

## Benefits and challenges of data transformation

Transforming data yields several benefits:

* Data is transformed to make it better-organized. Transformed data may be easier for both humans and computers to use.
* Properly formatted and validated data improves data quality and protects applications from potential landmines such as null values, unexpected duplicates, incorrect indexing, and incompatible formats.
* Data transformation facilitates compatibility between applications, systems, and types of data. Data used for multiple purposes may need to be transformed in different ways..

However, there are challenges to transforming data effectively:

* Data transformation can be expensive. The cost is dependent on the specific infrastructure, software, and tools used to process data. Expenses may include those related to licensing, computing resources, and hiring necessary personnel.
* Data transformation processes can be resource-intensive. Performing transformations in an on-premises data warehouse after loading, or transforming data before feeding it into applications, can create a computational burden that slows down other operations. If you use a cloud-based data warehouse, you can do the transformations after loading because the platform can scale up to meet demand.
* Lack of expertise and carelessness can introduce problems during transformation. Data analysts without appropriate subject matter expertise are less likely to notice typos or incorrect data because they are less familiar with the range of accurate and permissible values. For example, someone working on medical data who is unfamiliar with relevant terms might fail to flag disease names that should be mapped to a singular value or notice misspellings.
* Enterprises can perform transformations that don’t suit their needs. A business might change information to a specific format for one application only to then revert the information back to its prior format for a different application.

## How to transform data

Data transformation can increase the efficiency of analytic and business processes and enable better data-driven decision-making. The first phase of data transformations should include things like data type conversion and flattening of hierarchical data. These operations shape data to increase compatibility with analytics systems. Data analysts and data scientists can implement further transformations additively as necessary as [**individual layers of processing**](https://www.stitchdata.com/blog/transformation-layer-data-modeling/). Each layer of processing should be designed to perform a specific set of tasks that meet a known business or technical requirement.

Data transformation serves many functions within the data analytics stack.

### Extraction and parsing

In the modern ELT process, data ingestion begins with extracting information from a data source, followed by copying the data to its destination. Initial transformations are focused on shaping the format and structure of data to ensure its compatibility with both the destination system and the data already there. Parsing fields out of comma-delimited log data for loading to a relational database is an example of this type of data transformation.

### Translation and mapping

Some of the most basic data transformations involve the mapping and translation of data. For example, a column containing integers representing error codes can be mapped to the relevant error descriptions, making that column easier to understand and more useful for display in a customer-facing application.

Translation converts data from formats used in one system to formats appropriate for a different system. Even after parsing, web data might arrive in the form of hierarchical JSON or XML files, but need to be translated into row and column data for inclusion in a relational database.

### Filtering, aggregation, and summarization

Data transformation is often concerned with whittling data down and making it more manageable. Data may be consolidated by filtering out unnecessary fields, columns, and records. Omitted data might include numerical indexes in data intended for graphs and dashboards or records from business regions that aren’t of interest in a particular study.

Data might also be aggregated or summarized. by, for instance, transforming a time series of customer transactions to hourly or daily sales counts.

BI tools can do this filtering and aggregation, but it can be more efficient to do the transformations before a reporting tool accesses the data.

### Enrichment and imputation

Data from different sources can be merged to create denormalized, enriched information. A customer’s transactions can be rolled up into a grand total and added into a customer information table for quicker reference or for use by customer analytics systems. Long or freeform fields may be split into multiple columns, and missing values can be imputed or corrupted data replaced as a result of these kinds of transformations.

### Indexing and ordering

Data can be transformed so that it’s ordered logically or to suit a data storage scheme. In relational database management systems, for example, creating indexes can improve performance or improve the management of relationships between different tables.

### Anonymization and encryption

Data containing personally identifiable information, or other information that could compromise privacy or security, should be anonymized before propagation. Encryption of private data is a requirement in many industries, and systems can perform encryption at multiple levels, from individual database cells to entire records or fields.

### Modeling, typecasting, formatting, and renaming

Finally, a whole set of transformations can reshape data without changing content. This includes casting and converting data types for compatibility, adjusting dates and times with offsets and format localization, and renaming schemas, tables, and columns for clarity.

## Refining the data transformation process

Before your enterprise can run analytics, and even before you transform the data, you must replicate it to a data warehouse architected for analytics. Most organizations today choose a cloud data warehouse, allowing them to take full advantage of ELT. Stitch can load all of your data to your [**preferred data warehouse**](https://www.stitchdata.com/integrations/destinations/) in a raw state, ready for transformation

# How to Calculate Z-Scores in Python

In statistics, a **z-score**tells us how many standard deviations away a value is from [the mean](https://www.statology.org/measures-central-tendency/). We use the following formula to calculate a z-score:

**z** = (X – μ) / σ

where:

* X is a single raw data value
* μ is the population mean
* σ is the population standard deviation

This tutorial explains how to calculate z-scores for raw data values in Python.

### ****How to Calculate Z-Scores in Python****

We can calculate z-scores in Python using **scipy.stats.zscore**, which uses the following syntax:

**scipy.stats.zscore(a, axis=0, ddof=0, nan\_policy=’propagate’)**

where:

* **a**: an array like object containing data
* **axis**: the axis along which to calculate the z-scores. Default is 0.
* **ddof**: degrees of freedom correction in the calculation of the standard deviation. Default is 0.
* **nan\_policy**: how to handle when input contains nan. Default is propagate, which returns nan. ‘raise’ throws an error and ‘omit’ performs calculations ignoring nan values.

The following examples illustrate how to use this function to calculate z-scores for one-dimensional numpy arrays, multi-dimensional numpy arrays, and Pandas DataFrames.

### ****Numpy One-Dimensional Arrays****

**Step 1: Import modules.**

**import pandas as pd**

**import numpy as np**

**import scipy.stats as stats**

**Step 2: Create an array of values.**

**data = np.array([6, 7, 7, 12, 13, 13, 15, 16, 19, 22])**

**Step 3: Calculate the z-scores for each value in the array.**

**stats.zscore(data)**

**[-1.394, -1.195, -1.195, -0.199, 0, 0, 0.398, 0.598, 1.195, 1.793]**

Each z-score tells us how many standard deviations away an individual value is from the mean. For example:

* The first value of “6” in the array is **1.394**standard deviations below the mean.
* The fifth value of “13” in the array is **0**standard deviations away from the mean, i.e. it is equal to the mean.
* The last value of “22” in the array is **1.793**standard deviations above the mean.

### ****Numpy Multi-Dimensional Arrays****

If we have a multi-dimensional array, we can use the **axis**parameter to specify that we want to calculate each z-score relative to its own array. For example, suppose we have the following multi-dimensional array:

**data = np.array([[5, 6, 7, 7, 8],**

**[8, 8, 8, 9, 9],**

**[2, 2, 4, 4, 5]])**

We can use the following syntax to calculate the z-scores for each array:

**stats.zscore(data, axis=1)**

**[[-1.569 -0.588 0.392 0.392 1.373]**

**[-0.816 -0.816 -0.816 1.225 1.225]**

**[-1.167 -1.167 0.5 0.5 1.333]]**

The z-scores for each individual value are shown relative to the array they’re in. For example:

* The first value of “5” in the first array is **1.159**standard The first value of “8” in the second array is **.816**standard deviations below the mean of its array.
* The first value of “2” in the third array is **1.167**standard deviations below the mean of its array.

### ****Pandas DataFrames****

Suppose we instead have a Pandas DataFrame:

**data = pd.DataFrame(np.random.randint(0, 10, size=(5, 3)), columns=['A', 'B', 'C'])**

**data**

**A B C**

**0 8 0 9**

**1 4 0 7**

**2 9 6 8**

**3 1 8 1**

**4 8 0 8**

We can use the **apply**function to calculate the z-score of individual values by column:

**data.apply(stats.zscore)**

**A B C**

**0 0.659380 -0.802955 0.836080**

**1 -0.659380 -0.802955 0.139347**

**2 0.989071 0.917663 0.487713**

**3 -1.648451 1.491202 -1.950852**

**4 0.659380 -0.802955 0.487713**

The z-scores for each individual value are shown relative to the column they’re in. For example:

* The first value of “8” in the first column is **0.659** standard deviations above the mean value of its column.
* The first value of “0” in the second column is **.803** standard deviations below the mean value of its column.
* The first value of “9” in the third column is **.836** standard deviations above the mean value of its column.

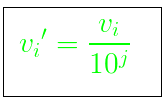
**Additional Resources:**

[How to Calculate Z-Scores in Excel](https://www.statology.org/z-score-excel/)

**Methods of Data Normalization –**

* Decimal Scaling
* Min-Max Normalization
* z-Score Normalization(zero-mean Normalization)

### Decimal Scaling Method For Normalization –

It normalizes by moving the decimal point of values of the data. To normalize the data by this technique, we divide each value of the data by the maximum absolute value of data. The data value, vi, of data is normalized to vi‘ by using the formula below –  


where j is the smallest integer such that max(|vi‘|)<1.

**Example –**

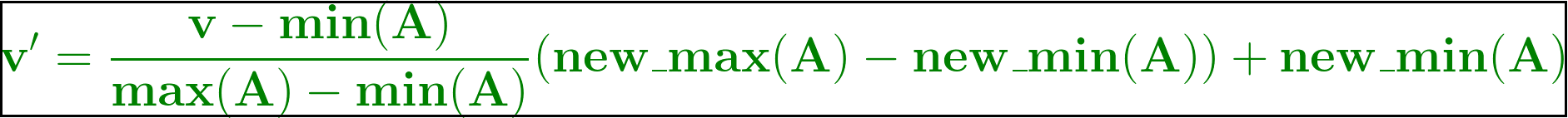
*Let the input data is: -10, 201, 301, -401, 501, 601, 701*

*To normalize the above data,****Step 1:****Maximum absolute value in given data(m): 701****Step 2:****Divide the given data by 1000 (i.e j=3)*

***Result:****The normalized data is: -0.01, 0.201, 0.301, -0.401, 0.501, 0.601, 0.701*

### Min-Max Normalization –

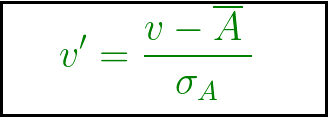
In this technique of data normalization, linear transformation is performed on the original data. Minimum and maximum value from data is fetched and each value is replaced according to the following formula.



Where A is the attribute data,  
Min(A), Max(A) are the minimum and maximum absolute value of A respectively.  
v’ is the new value of each entry in data.  
v is the old value of each entry in data.  
new\_max(A), new\_min(A) is the max and min value of the range(i.e boundary value of range required) respectively.

### Z-score normalization –

In this technique, values are normalized based on mean and standard deviation of the data A. The formula used is:



v’, v is the new and old of each entry in data respectively. σA, A is the standard deviation and mean of A respectively.