workato Automation Institute

COMMON FORMULAS

Data ingestion formulas

These formulas handle data from various sources and file formats (txt, csv, json, xml, html, etc.). Data ingestion includes connecting databases, executing queries, and retrieving data by reading, parsing, or extracting data in structured formats (JSON, XML). It also encompasses processing valuable information from API responses in the interface with various websites that provide APIs.

present?

This formula will check the input and if there is a value present, it will return true. If the input is nil, boolean false, an empty string, or an empty list, the formula will return false.

Syntax

`Input.present?`

- `Input` - An input datapill. It can be a string, number, date, or list datatype.

Sample usage

Formula	Output
`"Any Value".present?`	true
`123.present?`	true
`0.present?`	true
`"2017-04-02T12:30:00.000000-07:00".present?`	true
`nil.present?`	false
`"".present?`	false
`[].present?`	false

How it works

If the input is null, an empty string or an empty list, the formula will return false. For any other data, it returns true.

presence

Returns the data if it exists, returns nil if it does not.

Syntax

`Input.presence`

- `Input` - An input datapill. It can be a string, number, date, or datetime datatype.

Sample usage

Formula	Output
`nil.presence`	nil
`"".presence`	nil
`"Any Value".presence`	"Any Value"
`45.0.presence`	45.0
`0.presence`	0

How it works

If the input is null or an empty string, the formula will return nil. For any other data, it returns the original input data.

where

Retrieves only the rows (hashes) which satisfy the specified WHERE condition. This formula accepts a single argument in the form of a hash with one or more key-value pairs.

This formula supports the following operands. Operands should be added to the end of key separated by a space.

Supported Operands

Name	Notation	Example
Equal to (default)	==	leads.where('state': 'CA')
More than	>	leads.where('company_revenue >': 10000)
More than or equal to	>=	leads.where('company_revenue >=': 10000)
Not equal to	!=	leads.where('state !=': 'CA')

Sample Usage

Formula	Result
contacts.where('state': 'CA', 'company_revenue >=': 10000)	[{ 'name' => 'Jack', 'email' => 'jack@hbo.com', 'state' => 'CA', 'company' => 'HBO', 'company_rev' => 30000 }]

Pro tip: Use datapills as the conditional argument!

Instead of using a static value (for example, 'CA'), you can use a datapill as the conditional argument. The value of the datapill will be processed at run-time.

contacts.where(state: datapill)

pluck

Retrieves only the columns which have been specified.

Sample Usage

Formula	Result
contacts.pluck("email")	["joe@abc.com", "jill@nbc.com"]
contacts.where("state ==": "CA").pluck("email", "company")	[["joe@abc.com", "ABC"], ["jill@nbc.com", "NBC"],]
contacts.pluck("email", ["description", "summary"])	[["joe@abc.com", "First time buyer"], ["jill@nbc.com", "Referral"], ["joan@nbc.com", "Recurring customer"], ["jack@hbo.com", "Recurring customer"]]

Data storage formulas

Data storage formulas allow storage of raw and/or processed data for performance and stability when designing warehouses and lakes for schema integration. It handles multidimensional data representations and manipulation.

flatten

Flattens a multi-dimensional array (i.e. array of arrays) to a single dimension array.

Syntax

`List`.flatten

- `List` - An input of list datatype.

Sample usage

Formula	Result
`[[1, 2, 3], [4, 5, 6]]`.flatten	[1, 2, 3, 4, 5, 6]
`[[1, [2, 3], 9], [9, 8, 7]]`.flatten	[1, 2, 3, 9, 9, 8, 7]

compact

Removes nil values from array and hash.

Sample usage

Formula	Result
["foo", nil, "bar"].compact	["foo", "bar"]
{ foo: 1, bar: nil, baz: 2 }.compact	{ foo: 1, baz: 2 }

concat

Concatenates 2 lists into a single list. Nested lists will NOT be flattened.

Syntax

`List`.concat(`list_to_be_joined`)

- `List` - An input of list datatype
- `list_to_be_joined` - The other list to be concatenated with the original list input.

Sample usage

Formula	Result
`["Hello", "World", nil, ["Sub-array", "Here"]]`.concat(`["Workato", "Rocks"]`)	["Hello", "World", nil, ["Sub-array", "Here"], "Workato", "Rocks"]

reverse

Reverses the order of a list.

Syntax

`List`.reverse

- `List` - An input of list datatype.

Sample usage

Formula	Result
`[[1, 2, 3], [4, 5, 6]]`.flatten	[1, 2, 3, 4, 5, 6]
`[[1, [2, 3], 9], [9, 8, 7]]`.flatten	[1, 2, 3, 9, 9, 8, 7]

Data transformations formulas

These are techniques for integrating data from various sources. It combines and merges datasets, handles data inconsistencies, and creates unified datasets for further analysis and decision-making. Many crucial data orchestration concepts such as data alignment, cleaning, validation, standardization, and integrity happen using these formulas.

to_date

This formula converts the input data into a date. Returns the date formatted as YYYY-MM-DD.

Syntax

String.to_date(format: format)

- String - An input datetime or a string that describes a date or datetime.
- format - (optional) The date format of the input written as a string. If not specified, Workato will parse the input string automatically.

Sample usage

Formula	Output
"23-01-2020 10:30 pm".to_date(format: "DD-MM-YYYY")	"2020-01-23"
"01-23-2020 10:30 pm".to_date(format: "MM-DD-YYYY")	"2020-01-23"
"2020/01/23".to_date(format: "YYYY/MM/DD")	"2020-01-23"

How it works

Converts the input data into a date datatype.

strftime

Returns a datetime input as a user-defined string.

Syntax

Date.strftime(format)

- Date - An input date or datetime.
- format - The format of the user-defined datetime written as a string.

Sample Usage

Formula	Result
"2020-06-05T17:13:27.000000-07:00".strftime("%Y/%m/%d")	"2020/06/05"
"2020-06-05T17:13:27.000000-07:00".strftime("%Y-%m-%dT%H:%M:%S%z")	"2020-06-05T17:13:27-0700"
"2020-06-05T17:13:27.000000-07:00".strftime("%B %e, %l:%M%p")	"June 5, 5:13 pm"
"2020-06-05T17:13:27.000000-07:00".strftime("%A, %d %B %Y %k:%M")	"Friday, 05 June 2020 0:00"

How it works

Allows the user to define a datetime format. Returns the datetime input in the specified format.

to_s

Converts non-string data types such as numbers or dates to a string (text) datatype.

Syntax

Input.to_s

- Input - Any input data. You can use number, array, object, or datetime datatypes.

Sample usage

Formula	Result
-45.67.to_s	"-45.67"
[1,2,3].to_s	"[1,2,3]"
{key: "Workato"}.to_s	"{:key=>"Workato"}"
""2020-06-05T17:13:27.000000-07:00"".to_s	"2020-06-05T17:13:27.000000-07:00"
"2020-06-05T17:13:27.000000-07:00".to_s(:short)	"05 Jun 17:13"
"2020-06-05T17:13:27.000000-07:00".to_s(:long)	"June 05, 2020 17:13"

How it works

This formula returns a string representation of the input data.

to_i

Converts data to an integer (whole number) datatype.

Syntax

Input.to_i

- Input - Numerical input data. You can use a string datatype or a float datatype.

Sample usage

Formula	Result
today.to_time.to_i	1645660800
now.to_i	1645714000

How it works

This formula checks if the input contains any numbers. If no numbers are found, it returns 0. If the number has a decimal point, everything after the decimal is omitted.

first

This formula returns the first item in a list.

It can also be used to return the first n items in a list. In this case, the output will be formatted as a list.

Syntax

`List`.first(`number`)

- `List` - An input list.
- `number` - (optional) The number of items to retrieve from the list. If not specified, the formula will return only one item.

Sample usage

Formula	Output
`["One","Two","Three","Four","Five"]`.first()	“One”
`["One","Two","Three","Four","Five"]`.first(2)	["One", "Two”]
`[1,2,3,4,5]`.first()	1
`[1,2,3,4,5]`.first(3)	[1,2,3]

How it works

This formula returns the first n items from a list. If n is greater than one, the output is formatted as a list.

last

This formula returns the last item in a list.

It can also be used to return the last n items in a list. In this case, the output will be formatted as a list.

Syntax

`List`.last(`number`)

- `List` - An input list.
- `number` - (optional) The number of items to retrieve from the list. If not specified, the formula will return only one item.

Sample usage

Formula	Output
`["One","Two","Three","Four","Five"]`.last()	“Five”
`["One","Two","Three","Four","Five"]`.last(2)	["Four", "Five”]
`[1,2,3,4,5]`.last()	5
`[1,2,3,4,5]`.last(3)	[3,4,5]

How it works

This formula returns the last n items from a list. If n is greater than one, the output is formatted as a list.

gsub

Replace parts of a text string. Returns a new string with the replaced characters.

Syntax

String.gsub(find,replace)

- String - An input string. You can use a datapill or a static string value.
- find - The string or regular expression (regex) to look for. Use a /pattern/ syntax for regex.
- replace - The replacement string. You can define the replacement using a string or hash.

Sample usage

Formula	Result
"I have a blue house and a blue car".gsub("blue", "red")	"I have a red house and a red car"
"Jean Marie".gsub("J", "M")	"Mean Marie"
"Jean Marie".gsub(/[Jr]/, 'M')	"Mean MaMie" "Jean
"Jean Marie".downcase.gsub("j", "M")	"Mean marie"

How it works

This formula is similar to find and replace. It takes two input parameters:

- **First input:** The string that you plan to replace. The input is case-sensitive, so make sure to input uppercase or lowercase correctly to find all occurrences that are an exact match.
- **Second input:** The new string that replaces all occurrences of the first input.

split

This formula divides a string around a specified character and returns an array of strings.

Syntax

String.split(char)

- String - An input string value. You can use a datapill or a static value.
- char - (optional) The character to split the text at. This is case sensitive. If no character is defined, then by default, strings are split by white spaces.

Sample Usage

Formula	Result
"Ms-Jean-Marie".split("-")	["Ms", "Jean", "Marie"]
"Ms Jean Marie".split	["Ms", "Jean", "Marie"]
"Split string".split()	["Split", "string"]
"Split string".split("t")	["Split", " s", "ring"]
"01/23/2014".split("/")	["01", "23", "2014"]
"01/23/2014".split("/").join("-")	"01-23-2014"

How it works

This formula looks for the specific character in the input string. Every time it is found, the input will be split into a new string.

Data Governance Formulas

Data governance entails tackling lineage and dictionaries, accessing policies, and change management.

encode

Returns the string encoded.

Syntax

String.encode(encoding)

- String - An input string.
- encoding - Name of the encoding (for example, Windows-1252). Learn more about ruby encodings here.

Sample usage

Formula
"Jean Marie".encode("Windows-1252")

encode /decode

The following encode and decode operations are also available:

- encode_hex / decode_hex
- encode_url / decode_url
- encode_base64 / decode_base64
- encodeurlsafebase64 / decodeurlsafebase64

encrypt

Encrypts the input string with a secret key using AES-256-CBC algorithm. Encrypted **output** string is packed in RNCryptor V3 format and base64 encoded.

Example

encrypt([ssn], [encryption_key])

Note: For both encrypt and decrypt, the encryption key should not be hard coded in the recipe. Use environment properties (with key or password in the name) to store the encryption keys.

decrypt

Decrypts the encrypted input string with a secret key using AES-256-CBC algorithm. Encrypted **input** string should be packed in RNCryptor V3 format and base64 encoded.

Example

decrypt([encrypted_ssn], [encryption_key])

Note: For both encrypt and decrypt, the encryption key should not be hard coded in the recipe. Use environment properties (with key or password in the name) to store the encryption keys.

Data Operations (Activation) Formulas

Data operations lead to deciding architectures (ETL/ELT), features, and details of data integration tools. Some roles of data ops are scheduling data workflows, managing dependencies, error handling and recovery, monitoring, and logging.

now

Returns the time and date at runtime in US Pacific Time Zone (PST).

Sample usage

Formula	Output
now	"2022-02-01T07:00:00.000000-08:00"
now + 8.hours	"2022-02-01T15:00:00.000000-08:00"
now + 2.days	"2022-02-03T07:00:00.000000-08:00"

How it works

The formula calculates the timestamp when the job is processed. Each step using this formula returns the timestamp at which the step runs.

beginning_of_day

Returns datetime for midnight on date of a given date/datetime.

Syntax

`Date`.beginning_of_day

- `Date` - An input date or datetime.

Sample usage

Formula	Result
`today`.beginning_of_day	"2020-12-02T00:00:00.000000-07:00"
`"2020-06-01"`.to_date.beginning_of_day	"2020-06-01T00:00:00.000000+00:00"
`"2020-06-01T01:30:45.000000+00:00"`.beginning_of_day	"2020-06-01T00:00:00.000000+00:00"

Pro tip! Other variations of this formula include:

- beginning_of_hour
- beginning_of_week
- beginning_of_month
- beginning_of_year

Learn more!

Explore the full list of Workato supported formulas in the Formulas landing page at **https://docs.workato.com/formulas.html**