ReKisstory Tutorial for Data Mix

2025-02-09
Go Sugimoto

Is it hard to use Data Mix?

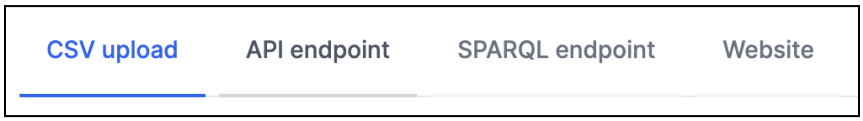
It is understandable, because it is new to many people. Why is it complicated? It is because our data in our world is so diverse. It is not so simple to organize diversity in a uniform manner. But do not worry, here we can learn how to make the best out of it.

In this document, we will first explain what you can do with Data Mix. Then, we present examples of importing different types of data in Data Mix. So, you can immediately copy and paste them for testing. You will also find some tips. Once you understand the basic use of Data Mix, you can start thinking about your data.

What is Data Mix?

Data Mix is a function in ReKisstory to ***import your data to "mix" it with the search results of ReKisstory***. ReKisstory only holds over 100 million fact items from Wikidata, which contains generic data often found in an encyropedia (e.g. Wikipedia). What if you can combine the data with more specified (local) data used for marketing, health care, sports, or climate change? Data Mix enables you to analyze a wide range of data in combination!

Four data import methods



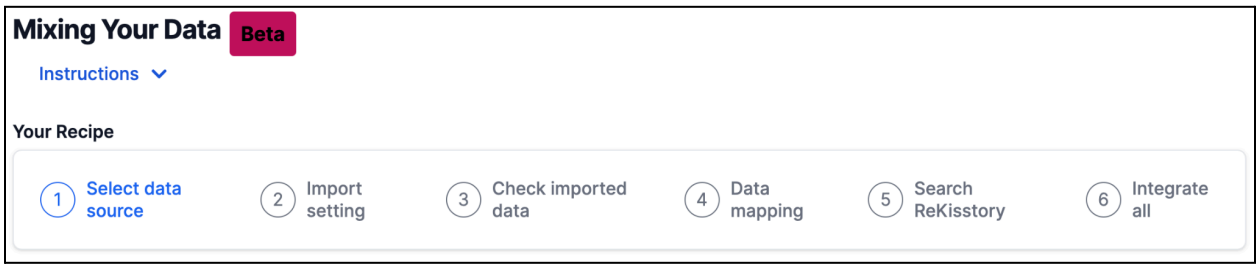
You do not always have to have your data in your hand. You can also use data accessible on the web. There are four methods to import you data:

1. CSV upload
2. API endpoint
3. SPARQL endpoint
4. Website (AI)

If you have data in CSV (you can export this file format from a spreadsheet, e.g. Microsoft Excel and Google Sheets). If you know how to obtain data via API or SPARQL endpoint, you can fetch data from the endpoint on the web. If you have no data, but would like to reuse data from a website, our experimental AI service automatically extract data from it.

Data Mix Procedure

We have a clear step-by-step indicator called ***Your Recipe*** at the top of the section:



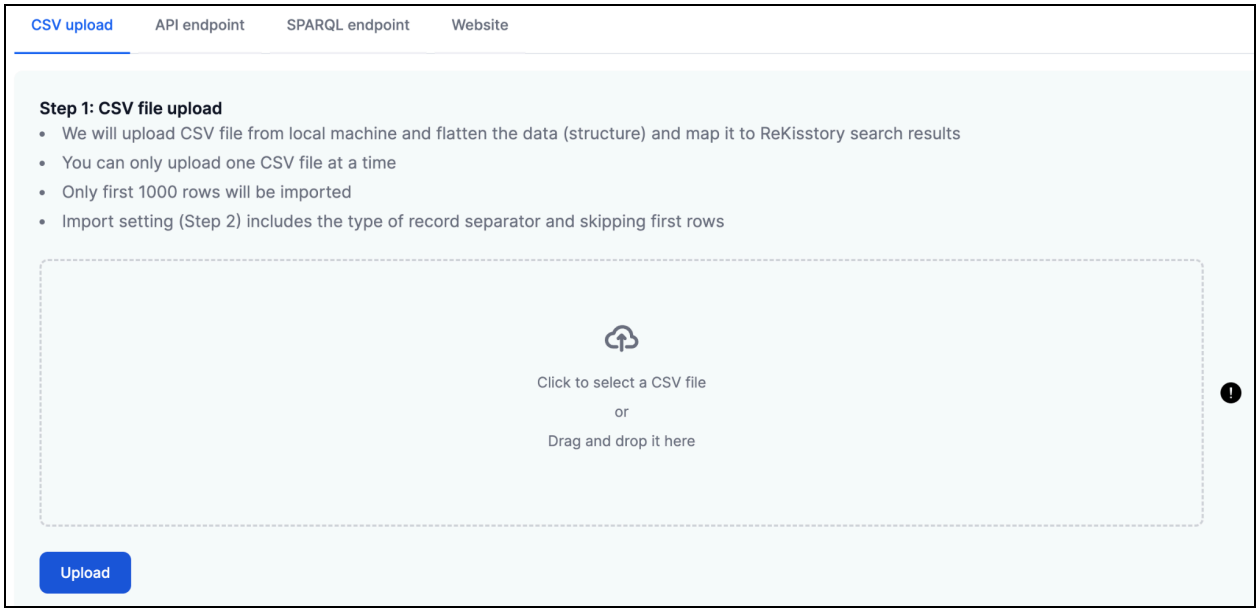
Once data is imported by one of the four methods (Step 1,2,3), the next step is to map the data to our data model (except the Website option)(Step 4). This process enables you to specify how your data should be combined with the ReKisstory search. An easy-to-understand mapping tool is provided, but you have to do it manually.

After the mapping is defined, you can simply search ReKisstory (***Compare***) and see the combined results (Step 5,6).

Step 1, 2, 3 example by method

1. CSV upload

Step 1: Drag and drop a CSV file in the input box and hit Upload

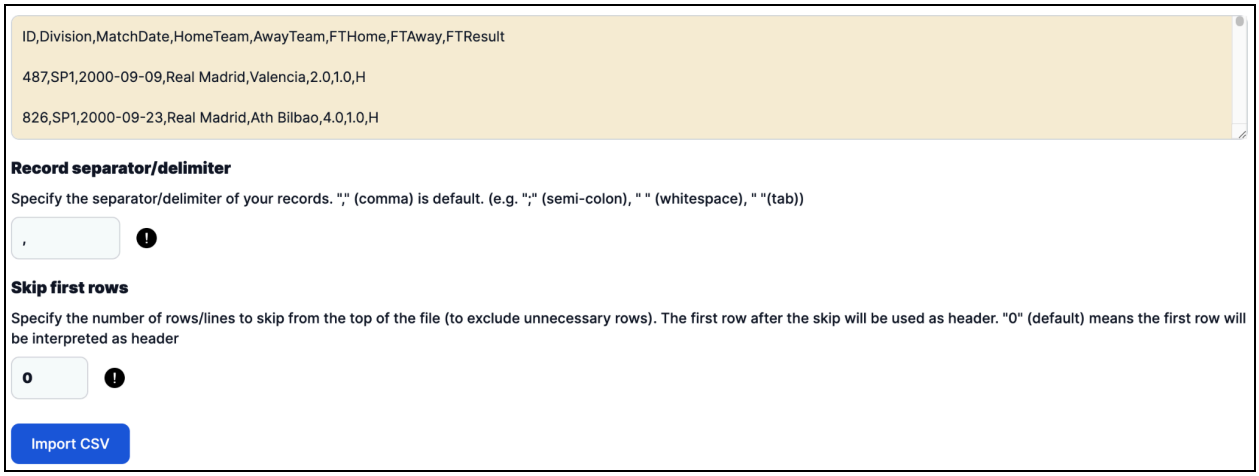


Step 2: Test import result and import setting

If the CSV upload is successful, you will see the test import screen. If not, please start from Step 1. In this step, sample data appears to check the imported data. Before starting the mapping, you can change the two settings for the real data import (optional). In particular, if the preview table looks strange, this is the chance to clean the data for the next step:

1) You can modify the data separator (if the comma is not the separator) including semicolon (;), and white spade or tab, because wrongly separated CSV data will produce unexpected mapping.

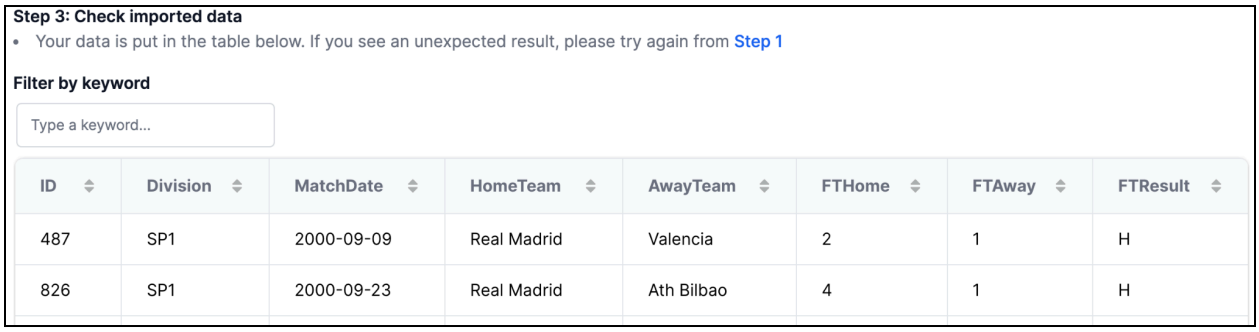
2) You can specify which row is the header row (i.e. which row starts the actual data). Type the number of rows to be skipped.



Comma and 0 (zero) are pre-filled as default. So, normally you do not need to do anything in this step. If your options are correct, hit Import CSV

Step 3: Check imported data

Double-check if the data is correctly displayed as a table.

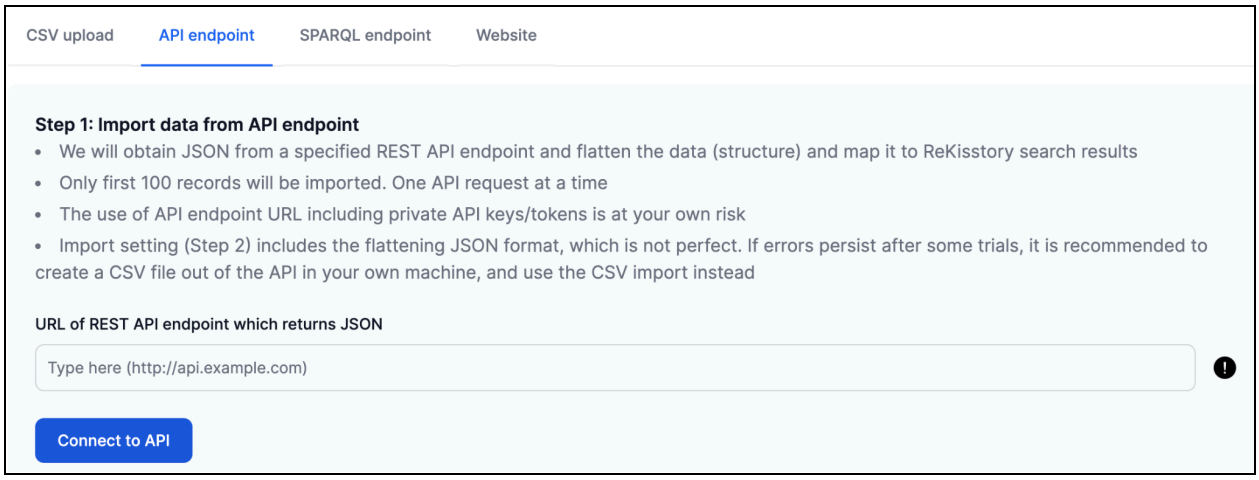


Step 4: Data Mapping

This step is the same for all three import methods. Please go further below to continue.

2. API endpoint

Step 1: Type the URL of a REST API endpoint in the input field and hit Connect to API



Example public API endpoints for testing:

- https://wikidata.reconci.link/en/api?query=obama (Used for this tutorial. Search results for “Obama”)
- https://datausa.io/api/data?drilldowns=Nation&measures=Population (US population by year)
- https://ergast.com/api/f1/drivers.json (F1 drivers)
- https://intavia-backend.acdh-dev.oeaw.ac.at/v2/api/events/search?datasets=http%3A%2F%2Fapis.acdh.oeaw.ac.at%2Fdata%2Fv5&datasets=http%3A%2F%2Fldf.fi%2Fnbf%2Fdata&datasets=http%3A%2F%2Fwww.intavia.eu%2Fsbi&datasets=http%3A%2F%2Fdata.acdh.oeaw.ac.at%2Fintavia%2Fcho%2Fv6&datasets=http%3A%2F%2Fdata.biographynet.nl&datasets=http%3A%2F%2Feuropeana_2023-10-27.intavia.eu&page=1&limit=50&q=Krieg (Search results for “Krieg” in biographical data)

Step 2: Test import result and import setting

If a test connection to API is successful, you will see the test import screen. If not, please start from Step 1. In this step, a sample data appears in a yellow box to check the imported data. Before starting the mapping, you can change the setting for the real data import (optional). In particular, you can flatten the data, because nested data will produce unexpected mapping.

Step 2: Import setting (API)

- Check the structure of JSON from the API endpint below
- In order to import JSON, a flat data structure is required
- If JSON is already flat, please indicate the key to the list of records
- If JSON is not flat (i.e nested), please indicate how to flatten the nestings


```
{
   "result": [
      {
         "description": "president of the United States from 2009 to 2017",
         "features": [
            {
               "id": "all_labels",
               "value": 100
            }
         ],
         "id": "Q76",
         "match": false,
         "name": "Barack Obama",
         "score": 100.0,
         "type": [
```


In the example, you can see the data is nested twice. The first nesting is “results” under which all data is found. ***To identify a nesting, find a list/array (square bracket) in the imported data ([ ])***. This must be specified in the first input field:

Specify the key in JSON which corresponds to the list of objects/records (i.e. list of table rows). Do not fill this input, if the top-level already contains the list

result

However, if you import data with only “results” specified as nesting, you will get concatenated data in a table column:

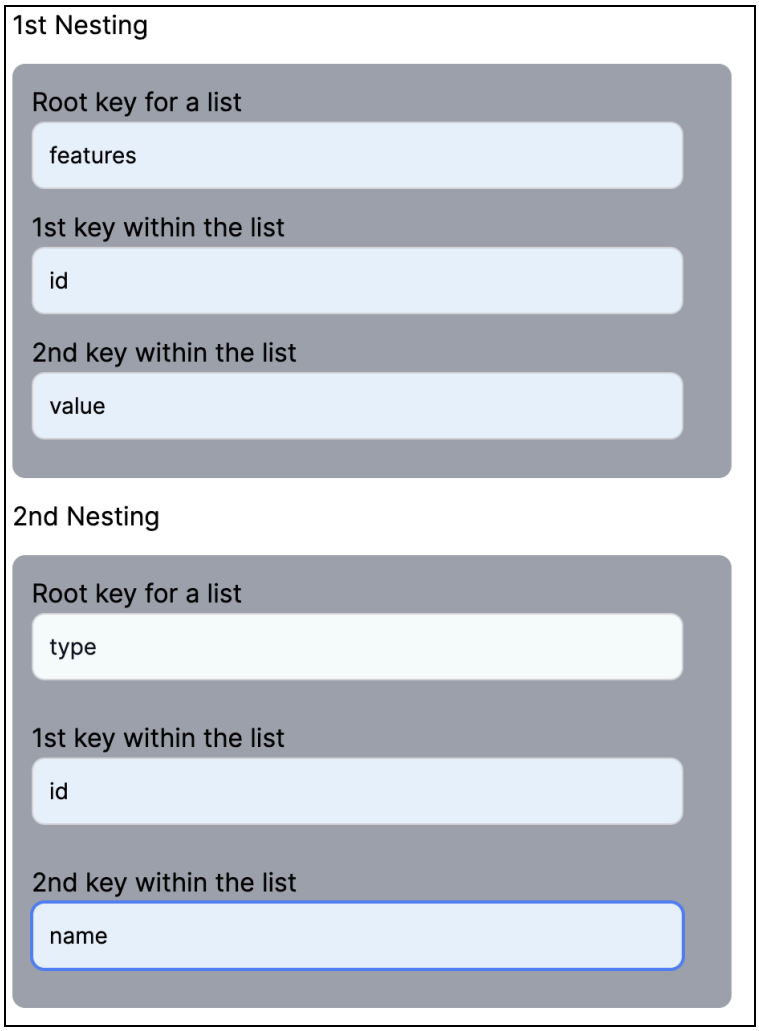
Example of nested data (if you do not change the import setting to avoid nesting)

description	features	id	m...	name	s...	type
president of the United States from 2009 to 2017	[{'id': 'all_labels', 'value': 100}]	Q76	False	Barack Obama	100.0	[{'id': 'Q5', 'name': 'human'}]

This is because there are two lists inside the data in the “results”. They are highlighted in red and blue). In this case, you find a list under “features” and “type”.


```
{
   "result": [
      {
         "description": "President of the United States from 2009 to 2017",
         "features": [
            {
               "id": "all_labels",
               "value": 100
            }
         ],
         "id": "Q76",
         "match": false,
         "name": "Barack Obama",
         "score": 100.0,
         "type": [
            {
               "id": "Q5",
               "name": "human"
            }
         ]
      },
```


Although the concatenated data is OK to mix with ReKisstory, it is not the best. To avoid this, the ReKisstory can flatten these nesting: you can use the import setting:



The two nested data is flattened as follows:

1) What is the key (i.e. name of the nesting list)? >> ***Root key for a list***
2) What are the keys in the list (i.e. name of the record in each list)? >> ***1st ket within the list***, ***2nd key within the list***

Step 3: Check imported data

features_id	features_value	description	name	id	score	match	type_id	type_name
all_labels	100	president of the United States from 2009 to 2017	Barack Obama	Q76	100.0	False	Q5	human

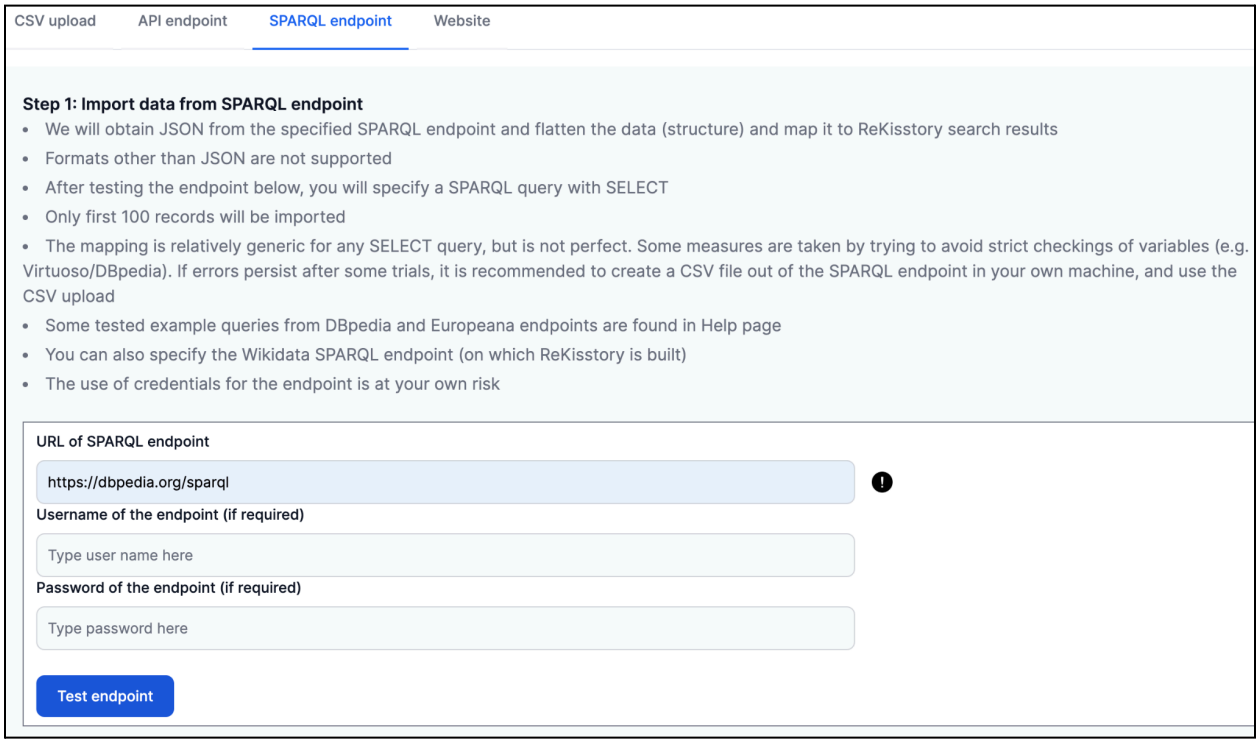
When the real data import is successful, you will see the preview of the imported data. In case of nesting, the tool automatically create columns with new names. In the example, new columns have names of concatenated key names:

- features_id
- features_value
- type_id
- type_name

Step 4: Data Mapping
This step is the same for all three import methods. Please go further below to continue.

3. SPARQL endpoint

Step 1: Type the URL of a SPARQL API endpoint in the input field and hit Test endpoint



Example public API endpoints for testing:

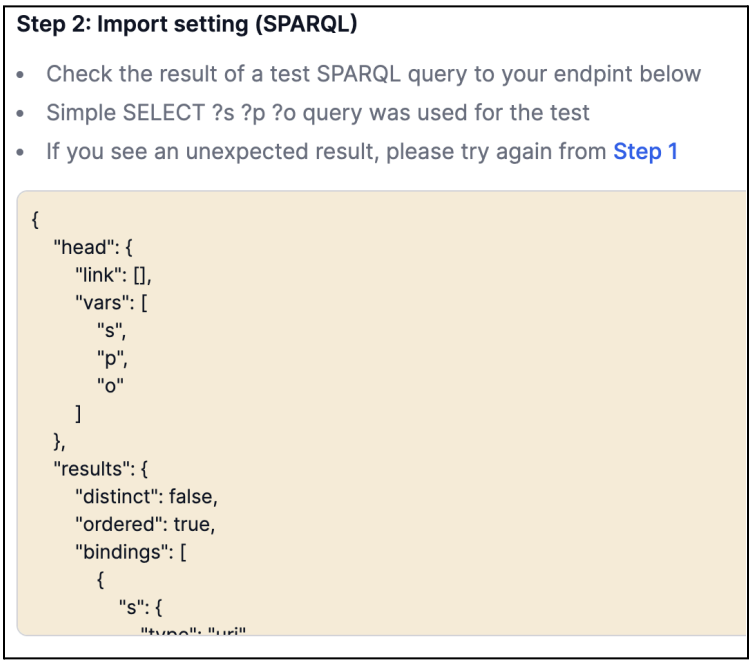
- https://dbpedia.org/sparql (Used for this tutorial. DBpedia generic encycropedia)
- https://sparql.europeana.eu/ (Europeana cultural heritage collections)
- https://query.wikidata.org/sparql (Wikidata. ReKisstory's data source can be also searched on your own way and added to the results)

. **Some considerations**

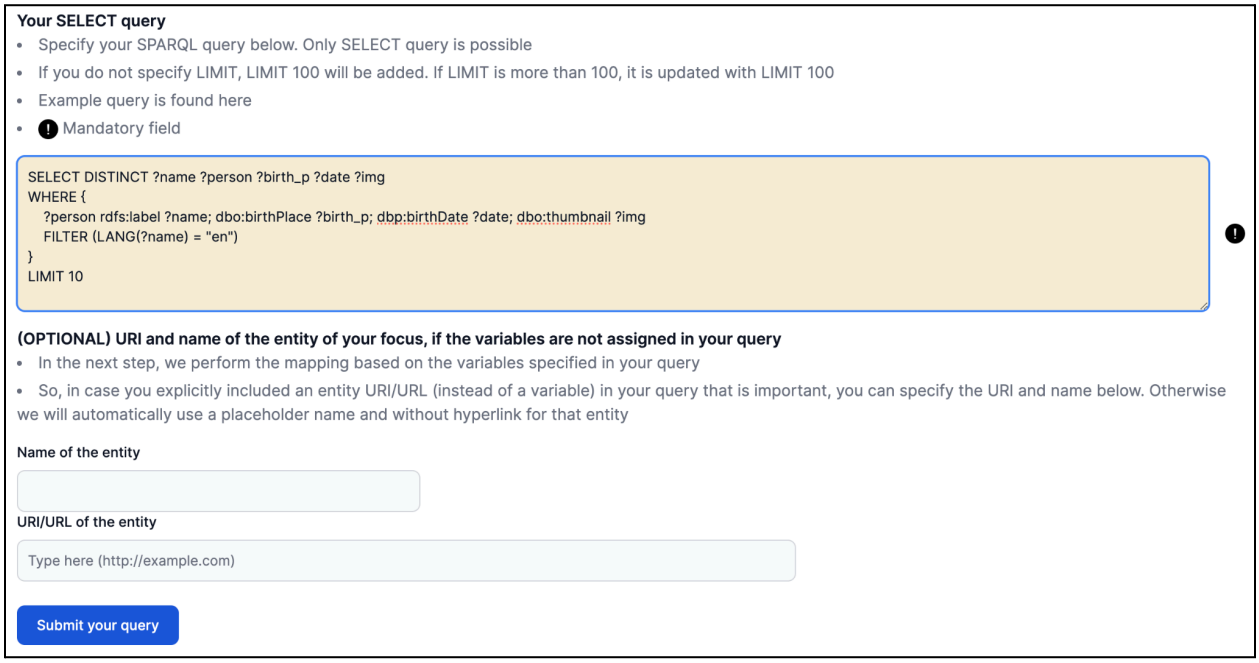
- Only SELECT query will be possible in the next step
- Only JSON format
- If not working, try to obtain CSV from the endpoint by yourself and use CSV upload method for Data Mix
- In case a username and password are needed to access the SPARQL endpoint, you can specify them. However, it is at your own risk

Step 2: Test import result and import setting

If a test connection to the endpoint is successful, you will see the test import screen. If not, please start from Step 1. In this step, sample data appears in a yellow box to check the imported data.



If everything is correct, you can type SELECT query in the input box:



Use the following example queries for testing:

https://dbpedia.org/sparql

```
SELECT DISTINCT ?name ?person ?birth_p ?date ?img
WHERE {
	?person rdfs:label ?name; dbo:birthPlace ?birth_p; dbp:birthDate ?date; dbo:thumbnail
?img
	FILTER (LANG(?name) = "en")
}
LIMIT 10
```

https://query.wikidata.org/sparql

```
SELECT DISTINCT ?person ?year ?birth_p ?name ?img
WHERE {
	?person wdt:P31 wd:Q5; wdt:P569 ?year; wdt:P20 ?birth_p; rdfs:label ?name;
wdt:P18 ?img .
	FILTER (LANG(?name) = "en")
}
LIMIT 10
```

https://sparql.europeana.eu/

```
SELECT DISTINCT ?cho ?uri ?wduri ?dc_creator ?dc_date ?dcterms_created ?dc_identifier
?dc_title ?dc_type ?dcterms_spatial ?edm_currentLocation ?edm_isShownBy
WHERE { ?cho dc:creator ?uri . ?uri owl:sameAs ?wduri . FILTER(contains(STR(?wduri),
"wikidata.org")) BIND(IRI(REPLACE(str(?cho), "http://data.europeana.eu/proxy/europeana/",
"http://data.europeana.eu/proxy/provider/")) AS ?provider_proxy)
BIND(IRI(REPLACE(str(?cho), "http://data.europeana.eu/proxy/europeana/",
"http://data.europeana.eu/aggregation/provider/")) AS ?provider_agg)
BIND(IRI(REPLACE(str(?cho), "http://data.europeana.eu/proxy/europeana/",
"http://data.europeana.eu/item/")) AS ?item) ?provider_proxy edm:type ?edm_type .
?provider_proxy dc:creator ?dc_creator . ?provider_proxy dc:date ?dc_date . OPTIONAL
{?provider_proxy dcterms:created ?dcterms_created} OPTIONAL {?provider_proxy
dc:identifier ?dc_identifier} OPTIONAL {?provider_proxy dc:title ?dc_title} OPTIONAL
{?provider_proxy dc:type ?dc_type} OPTIONAL {?provider_proxy dcterms:spatial
?dcterms_spatial} OPTIONAL {?provider_proxy edm:currentLocation ?edm_currentLocation}
?provider_agg edm:rights ?edm_rights . OPTIONAL {?provider_agg edm:hasView
?edm_webResource} OPTIONAL {?provider_agg edm:object ?edm_object} OPTIONAL
{?provider_agg edm:isShownAt ?edm_isShownAt} ?provider_agg edm:isShownBy
?edm_isShownBy . }
LIMIT 100
```

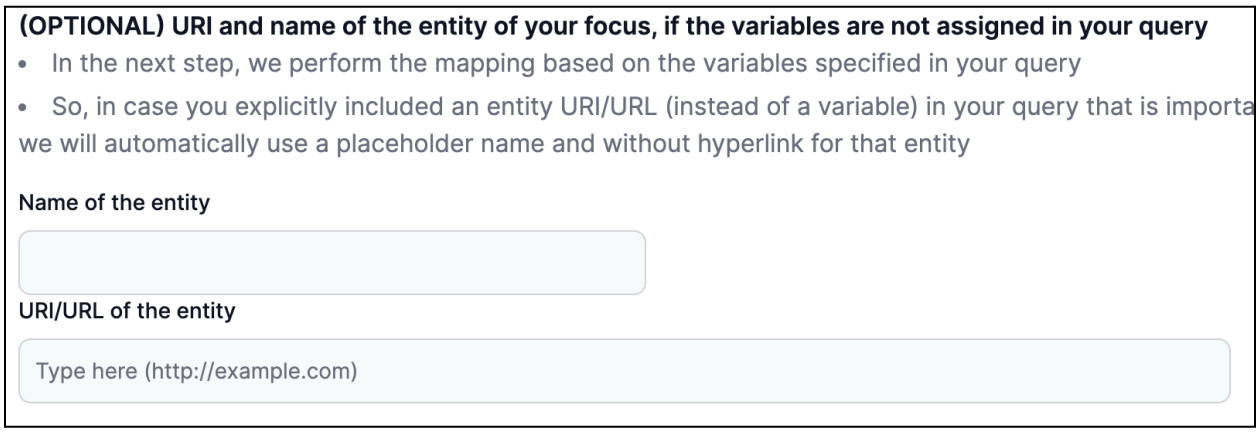
https://query.wikidata.org/sparql

```
SELECT DISTINCT ?roleTypeLabel ?starttime ?coordinate_loc
?pointtime ?coordinate_loc_ ?statement ?statement_coordinate_loc ?place_birth
?statement_place_birth ?birthdate ?statement_birth ?p_name_proxy ?p_name ?p_label
```

```
?s_name ?property ?propertyLabel  ?coordinate_place_birth (SAMPLE(?image_duplicate) as
?image)
WHERE {
BIND(?arg AS ?roleTypeLabel)
BIND(?date AS ?starttime)
BIND(?coordinate_place_birth AS ?coordinate_loc)
{
   wd:Q38234 ?p_name ?arg .
   ?p_name_proxy rdfs:label ?p_label .
   ?p_name_proxy wikibase:claim ?p_name .
   ?p_name_proxy wikibase:statementProperty ?s_name .
   OPTIONAL { wd:Q38234 wdt:P625 ?coordinate_loc_ . }
   OPTIONAL { wd:Q38234 wdt:P19 ?place_birth . }
   OPTIONAL { wd:Q38234 wdt:P569 ?birth_y . }

   ?arg ?s_name ?property .
   ?property rdfs:label ?propertyLabel .
   OPTIONAL { ?property wdt:P625 ?coordinate_loc . }
   OPTIONAL { ?property wdt:P18 ?image_duplicate . }
   { ?arg pq:P585 ?date . }
   UNION
   { ?arg pq:P577 ?pubdate . }
   FILTER (lang(?p_label) = 'ja') .
   FILTER (lang(?propertyLabel) = 'ja') .
   }
   UNION
   { wd:Q38234 wdt:P625 ?coordinate_loc_ . wd:Q38234 p:P625 ?statement_coordinate_loc .
wd:Q38234 ?p_name ?statement_coordinate_loc . ?p_name_proxy rdfs:label ?p_label .
?p_name_proxy wikibase:claim ?p_name . FILTER (lang(?p_label) = 'ja') . }
   UNION
   { wd:Q38234 wdt:P19 ?place_birth . wd:Q38234 p:P19 ?statement_place_birth .
wd:Q38234 ?p_name ?statement_place_birth . ?p_name_proxy rdfs:label ?p_label .
?p_name_proxy wikibase:claim ?p_name . ?place_birth wdt:P625 ?coordinate_place_birth .
FILTER (lang(?p_label) = 'ja') . }
   UNION
   { wd:Q38234 wdt:P569 ?birth_y . wd:Q38234 p:P569 ?statement_birth . wd:Q38234
?p_name ?statement_birth . ?p_name_proxy rdfs:label ?p_label . ?p_name_proxy
wikibase:claim ?p_name . FILTER (lang(?p_label) = 'ja') . }

   SERVICE wikibase:label { bd:serviceParam wikibase:language "ja, en" . }

}
GROUP BY ?roleTypeLabel ?starttime ?coordinate_loc
?pointtime ?coordinate_loc_ ?statement ?statement_coordinate_loc ?place_birth
?statement_place_birth ?birthdate ?statement_birth ?p_name_proxy ?p_name ?p_label
?s_name ?property ?propertyLabel ?coordinate_place_birth
```

Optionally, you can assign URI and name of the main entity:

As the data mapping in generic, you do not need to include URIs in your SPARQL query. We use variables for mapping. So, if URIs are used in your query, ReKisstory cannot know them: we will use a placeholder name and the result will have no hyperlinks. If you would like to have hyperlinks in the result, you can specify in the input bow below. This would be useful, especially for the main entity you will see in the data.



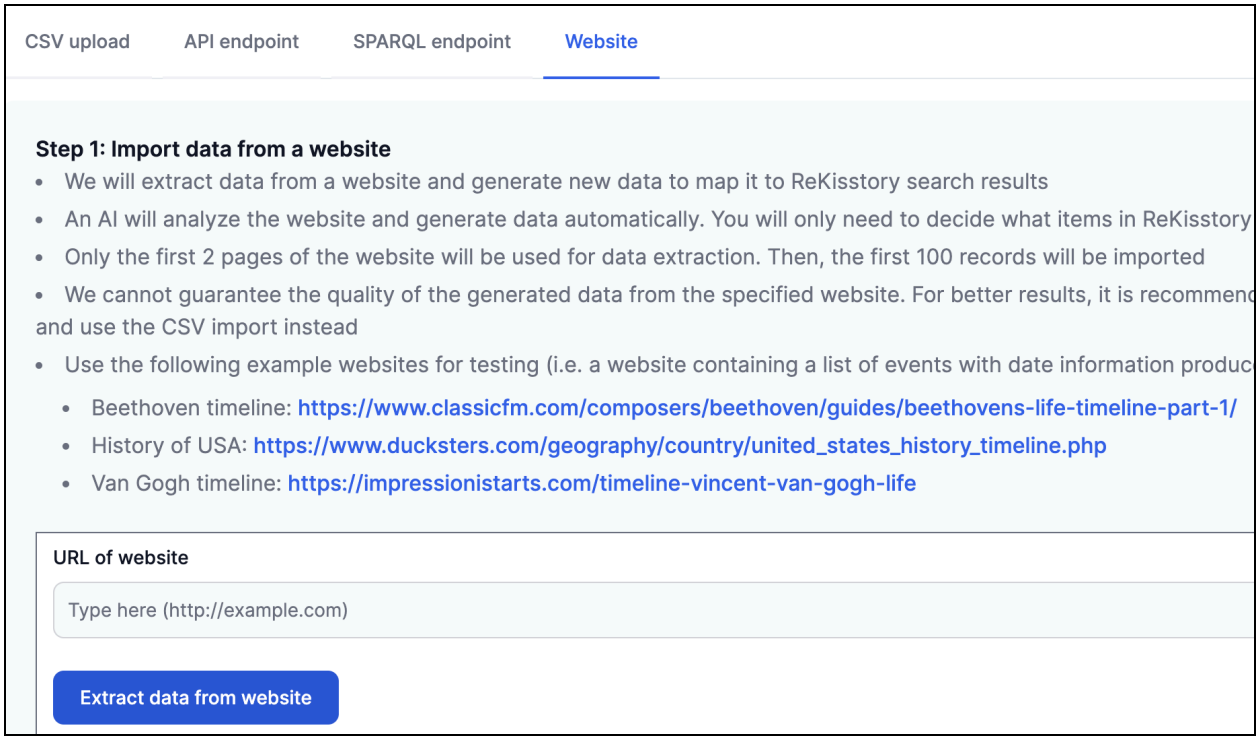
Step 3: Check imported data
When the real data import is successful, you will see the preview of the imported data.

name	person	birth_p	date	img
Cab Calloway	http://dbpedia.org/resource/Cab_Calloway	http://dbpedia.org/resource/Rochester,_New_York	1907-12-25	http://commons.wikimedia.org/wiki/Special:FilePath
Cabell Breckinridge	http://dbpedia.org/resource/Cabell_Breckinridge	http://dbpedia.org/resource/Albemarle_County,_Virginia	1788-07-14	http://commons.wikimedia.org/wiki/Special:FilePath width=300

Step 4: Data Mapping
This step is the same for all three import methods. Please go further below to continue.

4. Website (AI)

Step 1: Type the URL of a website in the input field and hit Extract data from website



You go to Step 5, if the information extraction is successful by AI.

Step 4: Data Mapping

Data Mapping is slightly complicated process, because there is some manual work. However, our interface makes it as easy as possible.

You need to fill the left input boxes (“Your Column Header”) to find the match with the ReKisstory data model (“Target Column Header”). Other columns shows what a target column header (per row) should contain. Your Column Headers are the labels in the header of your data (you saw the preview in the table in Step 3)[1].

For instance, The first row is about Item (link). This column should contain unique identifiers for your data. It could be a numeric number (ID1, 2, 3…) or URI (http://www.example.com/id1, http://www.example.com/id2, http://www.example.com/id3..). Data type is xsd:anyURI as most desirable. The description of header is given. To remember your column header.

[1] The “headers” are called differently, depending on your import method. For CSV upload, they are header columns (labels of the columns). For API, they are actually keys in the JSON data (because we “mapped” keys to the table headers in Step 2). For SPARQL, they are the variables used in the SELECT query.

Your Column Header	Target Column Header	Data Type	Description	Example
ID	Item (link)	xsd:anyURI (most desirable)	Unique Identifier. URI of item is the most desirable	123 (integer), "123" (string), http://dbpedia.org/resource/Shohei_Ohtani (URI)
HomeTeam	Item	RDF Literal	Name of item	Shohei Ohtani
column header 3	Object (link)	xsd:anyURI	URI of Object (that Item has a Relation)	http://www.wikidata.org/entity/statement/Q34661-100ba6c8-43a9-5d4f-62d9-f3b21f08311a
AwayTeam	Object	RDF Literal	Name of Object (that Item has a Relation)	Judith and the Head of Holofernes

In case of MatchesRealMadrid.csv example from CSV upload, we can start mapping:

- ID can be the **“Item (link)”** header, because it contains unique identifier
- HomeTeam can be in the **“Item”** header, because it contains the name of home teams
- AwayTeam can be in the **“Object”** header, because it contains the name of away teams

Two Timeline Visualization possibilities

If your data contains numeric data together with timestamps, you have two choices: 1) use them as statistics over time, 2) use the as a single data point over time. The choic will determine what type of Timeline visualization will be provided. See the Data Mix results section below for more details.

If you opt for 1), provide a mapping for *Statistics* and *Statistics 2* in the mapping table. If you opt for 2), do not specify them. Instead, you can use other fields for your numeric data for mapping.

Complete mapping example
(“FTHome” and “FTAway” are numeric data and are used as statistics over time (option 1 above))

Your Column Header	Target Column Header
ID	Item (link)
HomeTeam	Item
AwayTeam	Object
MatchDate	Starttime

FTHome[2]	Statistics
FTAway[3]	Statistics2

The same data mapping, except FTHome and FTAway will not be used as statistics ove time, but single points.

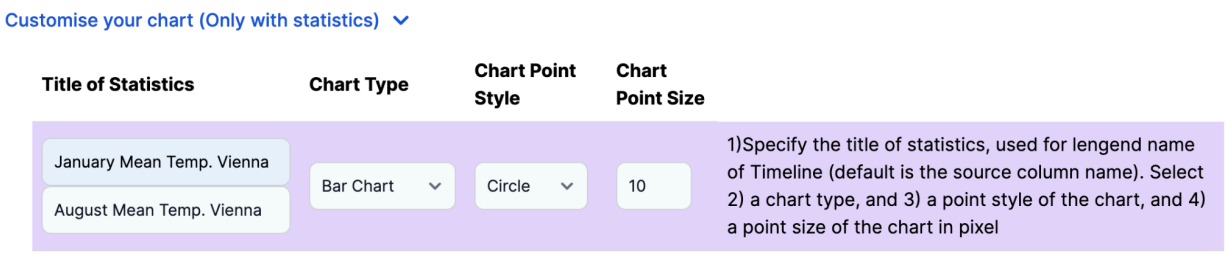
Your Column Header	Target Column Header
ID	Item (link)
HomeTeam	Item
MatchDate	Starttime
FTHome	Object
FTAway	Relation

Some considerations:

- At least one header should be filled, but no mandatory input (the more you define the mapping, the more data will appear in the results)
- Colored rows are recommended to fill to make sense of the results
    - Red: identifiers and names
    - Purplse: timestamp (for timeline view), coordinates (for map view), statistics (for timeline view)

Customize the chart design

If you specify a mapping at least one Statistics, you can expand the hidden below the mapping table. Here you can specify the labels of your statistical data (in legend), chart type (line, bar, scatter), chart type point stype (circle, square), and chart point size in pixels.



[2] Goal scores of the home team
[3] Goal scores of the away team

Step 5: Search ReKisstory (Compare)

Compare search here is almost the same as the Compare section. You just need to type text in the inputbox and select one item from the suggested items. You see a few more options for each item:

Merge with statistics:
If you check this box, the statistics in the mapping will be mapped to this entity. You can only tick one checkbox from one of the four items.

Values on Y-axis (for stats mixing):
You specify an integer in the box (negative or positive value) to plot this item as continuous data over time in the Timeline view. This option is used to increase the visibility of the merged data in the Timeline view. It is best to avoid the integer that are included in your external data. Otherwise, this item may overlap with statistical data, and it is hard to see the data separately.

Step 6: Double-check all your input

Hit the Search & Integrate Data button and see the Data Mix results. The search process may take longer time (1-2 minutes), because you search ReKisstory, as well as processing the mapping, integrating all data, and rendering the results in many views. Please be patient.

Data Mix Results

The result view is similar to the result view of Compare. It includes tabs for different views (Table, Timeline, Map, Gallery, and Wikipedia articles).

The timeline view may look different if your data contains numeric data (statistics) with timestamps. During the mapping, you can decide if you would like to take data as a) continuous data over time, or b) a single data point.

a) As continuous data over time

The result Timeline includes a) data from CSV about the results of La Liga Spanish football league and b) Real Madrid football club from Compare search. The three lines illustrate: Home scores (Real Madrid) in red and Away Scores (away team) in green, as well as the history of Real Madrid in blue (e.g. Zidadine Zidane started as head coach in January 2016).

b) As a single data point

The real examples of the whole CSV upload process can be found in Tutorials

- Data Mix with spatial data
- Data Mix with statistical data