2023Z Knowledge Graphs
RML transformation of a static dataset into a knowledge graph

Illia Tesliuk, 296076

1 Problem statement

This project aims to achieve the transformation of structured tabular data, where rows represent observations and columns represent attributes for these observations, into a knowledge graph represented by RDF subject-predicate-object triples. The goal of the transformation is to preserve relationships and semantics among different entities.

2 Project motivation

There are several motivations to convert static datasets into knowledge graphs. Firstly, contrary to tabular data, knowledge graphs allow for the representation of complex relationships between entities, thus enabling the discovery of connections that may not be apparent in tabular formats [1]. Secondly, with structured connections between entities, knowledge graphs make it easier to perform sophisticated search queries. Finally, knowledge graphs facilitate the integration of data from various sources and formats, offering a unified framework for the representation and organization of information. For example, each year a knowledge graph can be updated with the most streamed tracks within the last 12 months [1].

3 Tools

One of the common ways to transform static data into RDF graph is to use RML (RDF Mapping Language). RML is a language specifically designed for expressing mappings between different data formats, such as CSV, XML, and RDF and RDF graphs. It provides a way to define how the data in these non-RDF formats should be mapped to RDF triples of subjects, predicates and objects.

However, RML mappings appear to have been designed with a focus on machine consumption rather than human comprehension. YARRRML [2] instead is a more common choice to create transformation mappings. YARRRML is a human-readable text-based representation for declarative Linked Data generation rules. A user defines mapping rules in a YAML file and a special parser converts them into an RML mapping which can be later used to create RDF triples.

I use *yarrrml-parser*[2] to convert YARRRML rules into RML format, and *rmlmapper-java*[3] to convert tabular data into knowledge graphs.

4 Dataset

For this project, I've selected an open Spotify Songs database from Kaggle [4]. It consists of a CSV file with information about almost 30,000 tracks collected from the Spotify API. Each song is represented with 23 attributes, including basic information:

- *Track name*
- *Album name*

[1] https://www.kaggle.com/datasets/nelgiriyewithana/top-spotify-songs-2023
[2] https://github.com/RMLio/yarrrml-parser
[3] https://github.com/RMLio/rmlmapper-java
[4] https://www.kaggle.com/datasets/joebeachcapital/30000-spotify-songs

- *Artist name*
- *Playlist name*
- *Playlist genre*
- *Popularity*
- *Duration*

Songs are additionally described by a variety of metrics that shape musical characteristics [3]. Spotify uses raw audio analysis to translate the track sound into a set of measurable metrics, that can be utilized in recommendation algorithms:

- *Key* - overall key of the track
- *Mode* - "minor" / "major"
- *Tempo* - the overall "speed" of the track in beats per minute (BPM)
- *Loudness* - overall track loudness in decibels (dB)
- *Energy* - measures activity and intensity of the track
- *Danceability* - describes how suitable a track is for dancing
- *Speechiness* - detects the presence of spoken words in a track
- *Acousticness* - describes the degree to which a song relies on acoustic instruments
- *Liveness* - detects the presence of an audience in the recording (live / studio)
- *Valence* - describes the musical positiveness conveyed by a track

However, this dataset requires some preprocessing before being mapped into RDF triples. Namely, some tracks involve multiple artists, which is usually indicated via 'feat' keyword in either track name or artist name. It would be difficult to create a custom parse function in YARRRML, therefore this preprocessing must be done in Python.

I've detected 5 types of entities to be extracted from the Spotify dataset, namely tracks, albums, artists, playlists and genres. For convenience of the YARRRML's mapping process, I've split the original CSV file into 5 separate tables to be sources of single entities.

5 Ontologies

I didn't manage to find a single ontology that would cover the majority of the track attributes presented in the dataset. I'm using Music Ontology Specification [5] (*mo* prefix) to describe such terms as *Track*, *MusicArtist*, *Record*, *Genre* and a *duration* feature. Additionally, I've selected the Playlist Ontology [6] (*plo* prefix) to describe the *Playlist* class. Since the floating-point metrics described above are unique to the Spotify dataset and are not present in any music-related ontologies, I've decided to create a custom Spotify ontology to cover these terms.

6 Methodology

To represent properties unavailable in *mo* and *plo* ontologies, I've defined a custom *spo* ontology in a turtle (.ttl) file with the `https://mini.pw.edu.pl/kg/spo/#` prefix. For each attribute present in the dataset, I've identified their domains and ranges and populated the *spo* ontology with the corresponding instances of *rdf:Property* (See Table 1). Numerical song attributes are defined to have a *rdfs:Literal* range, while those linking to other entity types are linked to the corresponding class.

[5] http://musicontology.com/specification/
[6] https://lov.linkeddata.es/dataset/lov/vocabs/plo

	Property name	Domain	Range
1	key, mode, energy, tempo, loudness, danceability, speechiness, liveness, acousticness, instrumentalness, valence, duration, popularity	mo:Track	rdfs:Literal
2	artist	mo:Track	mo:MusicArtist
3	album	mo:Track	mo:Record
4	playlist	mo:Track	plo:Playlist
5	releaseDate	mo:Record	rdfs:Literal
5	genre	plo:Playlist	mo:Genre
6	subgenre	plo:Playlist	mo:Genre

Table 1: List of properties (*rdf:Property*) defined in a custom *spo* ontology.

I've used a single YAML file to declare YARRRML rules for every data source. The rules are grouped per entity and are assigned a unique key, namely *songs*, *artists*, *albums*, *playlists* and *genres*. Each entity section declares the name and format of its data source. YARRRML iterates over each row of the source file and applies rules defined in the corresponding section. Values of the columns are accessed using the following syntax: `$(column name)`.

IRI is used as the subject of the RDF triples for the entity. For example, the *song* prefix is defined as `https://mini.pw.edu.pl/kg/Song/`.Individual subjects are created by expanding the corresponding entity prefix with the ID read from the source file. For example, the subject rule for artists is defined as `artist:$(artistId)`, where *artist* is the entity prefix and *artistId* is the column name in the corresponding CSV file.

The goal of a predicate is to establish a relation between subject resources and object resources. Therefore, they are declared by providing the full property name that consists of the ontology prefix and the property's name within the ontology (e.g., *foaf:name*). Objects are declared with the help of data types (e.g., *xsd:integer*) and values that are read from the source file using the column access notation.

Alternatively, the value of the property can be read from a different source file. To achieve this, a mapping between the two entity types must be established. This is done using the `mapping` keyword followed by the name of the other entity. A new line starts with a `condition` keyword, which is followed by a declaration of a `function` in the next line.

I used the *equal* function, which takes two input parameters, each coming from a different source file. Each parameter is defined by a triple, including its name, the value read from the column of the source file, and an `s` or `o` letter, indicating whether the value of the column comes from the subject or the object of the triples. When the two column values are equal, a link between two entities is established. In total, I created rules for four mappings, namely (*Track*,*MusicArtist*), (*Track*,*Record*), (*Track*,*Playlist*) and (*Playlist*,*Genre*).

7 Results

YARRRML rules described in the previous section were converted into RML rules and applied to the input dataset. The conversion resulted in a list of RDF triples written in a Turtle file. The resulting knowledge graph comprises 605,253 RDF triples. The output file appropriately defines entities and their properties, including mappings between different entities. To confirm the correctness of the graph, I ran 10 SPARQL queries on the knowledge graph. The queries, their descriptions, and outputs are presented in the Appendix.

8 Conclusion

In this project, I transformed a static '30,000 Spotify Songs' dataset into an RDF knowledge graph. I defined YARRRML conversion rules in a single YAML file that establishes rules for every type of entity. These rules define how particular entities and their properties should be represented in the form of RDF subject-predicate-object triples. I created a custom ontology that defines properties

used in the Spotify recommender systems for representing the song's sonic characteristics. I tested the resulting knowledge graph with 10 different queries and ensured that mappings between different entities were established correctly. In conclusion, YARRRML is a convenient tool that allows the creation of user-friendly conversion tools.

References

[1] Katharina Brunner. Rdf, rml, yarrrml: A basic tutorial to create linked data from a relational database table, Mar 2024.

[2] Pieter Heyvaert, Ben De Meester, and Anastasia Dimou. Generating linked data with yarrrml, 2018.

[3] Nima Torabi. The inner workings of spotify's ai-powered music recommendations: How spotify shapes your playlist, Aug 2023.

A SPARQL Queries

```
SELECT (SAMPLE(?album_id) as ?album_id1) (SAMPLE(?album_name) as ?album_name1) (SAMPLE(?artist_name) as ?artist_name1) (SAMPLE(?releaseDate) as ?releaseDate1) (AVG(?popularity) as ?avg_popularity) WHERE {
    ?album_id rdf:type mo:Record ;
            foaf:name ?album_name ;
            spo:releaseDate ?releaseDate .
  FILTER(?releaseDate > "2000-01-01"^^xsd:date && ?releaseDate < "2010-01-01"^^xsd:date)
  ?song_id rdf:type mo:Track ;
            spo:album ?album_id ;
            spo:artist ?artist_id ;
            spo:popularity ?popularity .
  ?artist_id foaf:name ?artist_name .
}
GROUP BY ?album_id
```

Table | Response | 2182 results in 2.328 seconds | Simple view ☐ Ellipse ☑ | Filter query results | Page size: 50

	album_id1	album_name1	artist_name1	releaseDate1	avg_popularity
1	<https://mini.pw.edu.pl/kg/...	The Eminem Show	Eminem	"2002-05-26"^^<http://ww...	"83.0"^^<http://www.w3.org/2001/XMLSc...
2	<https://mini.pw.edu.pl/kg/...	We Sing. We Dance. We Steal Things.	Jason Mraz	"2008-05-12"^^<http://ww...	"82.0"^^<http://www.w3.org/2001/XMLSc...
3	<https://mini.pw.edu.pl/kg/...	All That We Needed	Plain White T's	"2005-01-01"^^<http://ww...	"80.0"^^<http://www.w3.org/2001/XMLSc...
4	<https://mini.pw.edu.pl/kg/...	Oral Fixation, Vol. 2 (Expanded Edition)	Shakira	"2005-11-28"^^<http://ww...	"80.0"^^<http://www.w3.org/2001/XMLSc...
5	<https://mini.pw.edu.pl/kg/...	Infest	Papa Roach	"2001-04-25"^^<http://ww...	"79.0"^^<http://www.w3.org/2001/XMLSc...
6	<https://mini.pw.edu.pl/kg/...	Mail on Sunday	Flo Rida	"2008-03-17"^^<http://ww...	"79.0"^^<http://www.w3.org/2001/XMLSc...

Figure 1: Albums released between 2000 and 2010, sorted by average popularity.

```
SELECT DISTINCT (SAMPLE(?album_id) as ?album_id1) (SAMPLE(?album_name) as ?album_name1) (AVG(?popularity) as ?avg_popularity) WHERE {
  ?album_id rdf:type mo:Record ;
       foaf:name ?album_name .
  ?song_id spo:album ?album_id;
           spo:artist ?artist_id ;
           spo:popularity ?popularity .
  ?artist_id foaf:name "Ed Sheeran" .
}
GROUP BY ?album_id
ORDER BY DESC(?avg_popularity)
```

Table | Response | 27 results in 0.04 seconds | Simple view ☐ Ellipse ☑ | Filter query results | Page size: 50

	album_id1	album_name1	avg_popularity
1	<https://mini.pw.edu.pl/kg/Album/5Nux7ozBJ5K...	I Don't Care (with Justin Bieber)	"90.0"^^<http://www.w3.org/2001/XMLSchema#decimal>
2	<https://mini.pw.edu.pl/kg/Album/3E12WU80fD...	Beautiful People (feat. Khalid)	"89.0"^^<http://www.w3.org/2001/XMLSchema#decimal>
3	<https://mini.pw.edu.pl/kg/Album/3T4tUhGYeR...	÷ (Deluxe)	"84.0"^^<http://www.w3.org/2001/XMLSchema#decimal>
4	<https://mini.pw.edu.pl/kg/Album/3oIFxDIo2fwu...	No.6 Collaborations Project	"83.25"^^<http://www.w3.org/2001/XMLSchema#decimal>
5	<https://mini.pw.edu.pl/kg/Album/52kvZcbEDm...	Perfect Duet (Ed Sheeran & Beyoncé)	"77.0"^^<http://www.w3.org/2001/XMLSchema#decimal>
6	<https://mini.pw.edu.pl/kg/Album/7oJa8bPFKVb...	Shape of You	"75.0"^^<http://www.w3.org/2001/XMLSchema#decimal>
7	<https://mini.pw.edu.pl/kg/Album/6Z5DhADmyy...	Best Part of Me (feat. YEBBA)	"74.0"^^<http://www.w3.org/2001/XMLSchema#decimal>
8	<https://mini.pw.edu.pl/kg/Album/3BjxjIkTZKUp...	South of the Border (feat. Camila Cabello & Cardi B) [Ch...	"69.0"^^<http://www.w3.org/2001/XMLSchema#decimal>

Figure 2: Ed Sheeran's albums, sorted by average popularity.

```
SELECT (SAMPLE(?artist_id) as ?artist_id1) (SAMPLE(?artist_name) as ?artist_name1) (AVG(?acousticness) as ?average_acousticness) WHERE {
  ?artist_id rdf:type mo:MusicArtist ;
       foaf:name ?artist_name .
  ?song_id spo:artist ?artist_id;
           spo:acousticness ?acousticness .
}
GROUP BY ?artist_id
ORDER BY DESC(?average_acousticness)
```

Table | Response | 10692 results in 1.243 seconds | Simple view ☐ Ellipse ☑ | Filter query results | Page size: 50

	artist_id1	artist_name1	average_acousticness
1	<https://mini.pw.edu.pl/kg/Artist/5...	Simón Campusano	"0.989e0"^^<http://www.w3.org/2001/XMLSchema#double>
2	<https://mini.pw.edu.pl/kg/Artist/8...	Nat King Cole	"0.989e0"^^<http://www.w3.org/2001/XMLSchema#double>
3	<https://mini.pw.edu.pl/kg/Artist/1...	Christian Leave	"0.983e0"^^<http://www.w3.org/2001/XMLSchema#double>
4	<https://mini.pw.edu.pl/kg/Artist/5...	The Magnetic Fields	"0.978e0"^^<http://www.w3.org/2001/XMLSchema#double>
5	<https://mini.pw.edu.pl/kg/Artist/2...	Gary Jules	"0.976e0"^^<http://www.w3.org/2001/XMLSchema#double>
6	<https://mini.pw.edu.pl/kg/Artist/2...	Wenzel	"0.973e0"^^<http://www.w3.org/2001/XMLSchema#double>
7	<https://mini.pw.edu.pl/kg/Artist/2...	organic_kid	"0.972e0"^^<http://www.w3.org/2001/XMLSchema#double>
8	<https://mini.pw.edu.pl/kg/Artist/5...	La Lá	"0.972e0"^^<http://www.w3.org/2001/XMLSchema#double>
9	<https://mini.pw.edu.pl/kg/Artist/7...	Mark Isham	"0.972e0"^^<http://www.w3.org/2001/XMLSchema#double>

Figure 3: Artists, sorted by the average acousticness of their songs.

```
SELECT (SAMPLE(?artist_id) as ?artist_id1) (SAMPLE(?artist_name) as ?artist_name1) (AVG(?danceability) as ?average_danceability) WHERE {
  ?artist_id rdf:type mo:MusicArtist ;
       foaf:name ?artist_name .
  ?song_id spo:artist ?artist_id;
           spo:danceability ?danceability .
}
GROUP BY ?artist_id
ORDER BY DESC(?average_danceability)
```

Table | Response | 10692 results in 4.58 seconds | Simple view ☐ Ellipse ☑ | Filter query results | Page size: 50

	artist_id1	artist_name1	average_danceability
1	<https://mini.pw.edu.pl/kg/Artist/1...	Fusion Groove Orchestra	"0.983e0"^^<http://www.w3.org/2001/XMLSchema#double>
2	<https://mini.pw.edu.pl/kg/Artist/9...	DJ ZsuZsu	"0.981e0"^^<http://www.w3.org/2001/XMLSchema#double>
3	<https://mini.pw.edu.pl/kg/Artist/6...	DJ Goozo	"0.979e0"^^<http://www.w3.org/2001/XMLSchema#double>
4	<https://mini.pw.edu.pl/kg/Artist/9...	[dunkelbunt]	"0.974e0"^^<http://www.w3.org/2001/XMLSchema#double>
5	<https://mini.pw.edu.pl/kg/Artist/4...	WE$T DUBAI	"0.971e0"^^<http://www.w3.org/2001/XMLSchema#double>
6	<https://mini.pw.edu.pl/kg/Artist/4...	Cal Scruby	"0.971e0"^^<http://www.w3.org/2001/XMLSchema#double>
7	<https://mini.pw.edu.pl/kg/Artist/9...	Greenskeepers	"0.97e0"^^<http://www.w3.org/2001/XMLSchema#double>
8	<https://mini.pw.edu.pl/kg/Artist/7...	Westside Cartel	"0.968e0"^^<http://www.w3.org/2001/XMLSchema#double>
9	<https://mini.pw.edu.pl/kg/Artist/2...	Sydney Yungins	"0.967e0"^^<http://www.w3.org/2001/XMLSchema#double>
10	<https://mini.pw.edu.pl/kg/Artist/7...	Mellow Man Ace	"0.966e0"^^<http://www.w3.org/2001/XMLSchema#double>

Figure 4: Artists, sorted by average danceability of their songs.

```
SELECT DISTINCT (SAMPLE(?artist_id) as ?artist_id1) (SAMPLE(?artist_name) as ?artist_name1) (COUNT(?song_id) as ?songs_count) WHERE {
  ?artist_id rdf:type mo:MusicArtist ;
       foaf:name ?artist_name .
  ?song_id spo:artist ?artist_id.
}
GROUP BY ?artist_id
ORDER BY DESC(?songs_count)
```

Press CTRL + <spacebar> to autocomplete

Table | Response | 10692 results in 0.676 seconds | Simple view ☐ Ellipse ☑ | Filter query results | Page size: 50

	artist_id1	artist_name1	songs_count
1	<https://mini.pw.edu.pl/kg/Artist/1053>	Queen	"130"^^<http://www.w3.org/2001/XMLSchema#integer>
2	<https://mini.pw.edu.pl/kg/Artist/11>	Martin Garrix	"87"^^<http://www.w3.org/2001/XMLSchema#integer>
3	<https://mini.pw.edu.pl/kg/Artist/1215>	Don Omar	"84"^^<http://www.w3.org/2001/XMLSchema#integer>
4	<https://mini.pw.edu.pl/kg/Artist/13>	David Guetta	"81"^^<http://www.w3.org/2001/XMLSchema#integer>
5	<https://mini.pw.edu.pl/kg/Artist/1161>	Drake	"68"^^<http://www.w3.org/2001/XMLSchema#integer>
6	<https://mini.pw.edu.pl/kg/Artist/132>	Hardwell	"68"^^<http://www.w3.org/2001/XMLSchema#integer>
7	<https://mini.pw.edu.pl/kg/Artist/93>	Dimitri Vegas & Like Mike	"68"^^<http://www.w3.org/2001/XMLSchema#integer>
8	<https://mini.pw.edu.pl/kg/Artist/3>	The Chainsmokers	"66"^^<http://www.w3.org/2001/XMLSchema#integer>
9	<https://mini.pw.edu.pl/kg/Artist/1329>	Logic	"65"^^<http://www.w3.org/2001/XMLSchema#integer>

Figure 5: Artists, sorted by the number of songs they wrote.

```
SELECT (SAMPLE(?genre_id) as ?genre_id1) (SAMPLE(?genre_name) as ?genre_name1) (COUNT(?playlist_id) as ?num_playlists) WHERE
    ?genre_id rdf:type mo:Genre ;
          foaf:name ?genre_name .
    ?playlist_id spo:genre ?genre_id ;
                foaf:name ?playlist_name .
}
GROUP BY (?genre_id)
ORDER BY DESC(?num_playlists)
```

Table | Response | 6 results in 0.026 seconds | Simple view ☐ Ellipse ☑ | Filter query results | Page size: 50

	genre_id1	genre_name1	num_playlists
1	<https://mini.pw.edu.pl/kg/Genre/1>	rap	"80"^^<http://www.w3.org/2001/XMLSchema#integer>
2	<https://mini.pw.edu.pl/kg/Genre/2>	pop	"80"^^<http://www.w3.org/2001/XMLSchema#integer>
3	<https://mini.pw.edu.pl/kg/Genre/5>	rock	"79"^^<http://www.w3.org/2001/XMLSchema#integer>
4	<https://mini.pw.edu.pl/kg/Genre/3>	r&b	"78"^^<http://www.w3.org/2001/XMLSchema#integer>
5	<https://mini.pw.edu.pl/kg/Genre/4>	latin	"78"^^<http://www.w3.org/2001/XMLSchema#integer>
6	<https://mini.pw.edu.pl/kg/Genre/0>	edm	"76"^^<http://www.w3.org/2001/XMLSchema#integer>

Figure 6: Genres, sorted by the number of their playlists.

```
SELECT ?pl_id (SAMPLE(?name) as ?pl_name) (SAMPLE(?genre_name) as ?genre_name1) (SAMPLE(?subgenre_name) as ?subgenre_name1) (COUNT(?
song_id) as ?song_count) WHERE {
  ?pl_id rdf:type plo:Playlist ;
       foaf:name ?name ;
       spo:genre ?genre ;
       spo:subgenre ?subgenre .
  ?genre foaf:name ?genre_name .
  ?subgenre foaf:name ?subgenre_name .

  ?song_id spo:playlist ?pl_id .
}
GROUP BY ?pl_id
ORDER BY DESC(?song_count)
```

Table | Response | 471 results in 1.016 seconds | Simple view ☐ Ellipse ☑ | Filter query results | Page size: 50

	pl_id	pl_name	genre_name1	subgenre_name1	song_count
1	<https://mini.pw.edu.pl/kg/Pl...	2020 Hits & 2019 Hits – Top Global Tracks 🔥🔥🔥	latin	latin pop	"100"^^<http://www.w3.org/20...
2	<https://mini.pw.edu.pl/kg/Pl...	90s Dance Hits	pop	dance pop	"100"^^<http://www.w3.org/20...
3	<https://mini.pw.edu.pl/kg/Pl...	Big Room EDM	edm	big room	"100"^^<http://www.w3.org/20...
4	<https://mini.pw.edu.pl/kg/Pl...	Big Room House | Festival Bangers	edm	big room	"100"^^<http://www.w3.org/20...
5	<https://mini.pw.edu.pl/kg/Pl...	Chillout & Remixes 💜	pop	indie poptimism	"100"^^<http://www.w3.org/20...
6	<https://mini.pw.edu.pl/kg/Pl...	City Pop 1985 シティーポップ	rock	album rock	"100"^^<http://www.w3.org/20...

Figure 7: Playlists, sorted by their size (number of songs).

```
SELECT (SAMPLE(?song_id) as ?song_id1) (SAMPLE(?artist_name) as ?artist_name1) (COUNT(?playlist_id) as ?playlist_count) WHERE {
  ?song_id rdf:type mo:Track ;
           spo:playlist ?playlist_id ;
           spo:artist ?artist_id .
  ?artist_id foaf:name ?artist_name .
}
GROUP BY ?song_id
ORDER BY DESC(?playlist_count)
```

Table | Response | 28352 results in 2.393 seconds | Simple view ☐ Ellipse ☑ | Filter query results | Page size: 50

	song_id1	artist_name1	playlist_count
1	<https://mini.pw.edu.pl/kg/Song/2Fxmhks0bxGSBdJ92v...	Billie Eilish	"8"^^<http://www.w3.org/2001/XMLSchema#int...
2	<https://mini.pw.edu.pl/kg/Song/7BKLCZ1jbUBVqRi2FVl...	The Chainsmokers	"8"^^<http://www.w3.org/2001/XMLSchema#int...
3	<https://mini.pw.edu.pl/kg/Song/0sf12qNH5qcw8qpgy...	The Weeknd	"7"^^<http://www.w3.org/2001/XMLSchema#int...
4	<https://mini.pw.edu.pl/kg/Song/14sOS5L36385FJ3OL8...	Kygo	"7"^^<http://www.w3.org/2001/XMLSchema#int...
5	<https://mini.pw.edu.pl/kg/Song/3ZCTVFBt2Brf31RLEnC...	Billie Eilish	"7"^^<http://www.w3.org/2001/XMLSchema#int...
6	<https://mini.pw.edu.pl/kg/Song/3eekarcy7kvN4yt5ZFzl...	Travis Scott	"7"^^<http://www.w3.org/2001/XMLSchema#int...
7	<https://mini.pw.edu.pl/kg/Song/6Gg1gjgKi2AK4e0qzsR...	Juice WRLD	"7"^^<http://www.w3.org/2001/XMLSchema#int...
8	<https://mini.pw.edu.pl/kg/Song/6wo37KVqFJhtuxPTpLC...	The Chainsmokers	"7"^^<http://www.w3.org/2001/XMLSchema#int...
9	<https://mini.pw.edu.pl/kg/Song/7CHi4DtfK4heMlQaud...	MEDUZA	"7"^^<http://www.w3.org/2001/XMLSchema#int...

Figure 8: Songs, sorted by the number of playlists they belong to.

```
SELECT ?song_id ?song_name ?artist_name ?album_name ?releaseDate WHERE {
  ?song_id rdf:type mo:Track ;
       foaf:name ?song_name ;
       spo:album ?album_id ;
       spo:artist ?artist_id .
  ?artist_id foaf:name ?artist_name .
  ?album_id foaf:name ?album_name ;
            spo:releaseDate ?releaseDate .
  FILTER(?releaseDate > "2015-01-01"^^xsd:date)
}
```

Table | Response | 16441 results in 21.013 seconds | Simple view ☐ Ellipse ☑ | Filter query results | Page size: 50

	song_id	song_name	artist_name	album_name	releaseDate
1	<https://mini.pw.ed...	I Don't Care (with Justin Bieber) - Loud Lu...	Ed Sheeran	I Don't Care (with Justin Bieber) [Loud Luxu...	"2019-06-14"^^<
2	<https://mini.pw.ed...	Memories - Dillon Francis Remix	Maroon 5	Memories (Dillon Francis Remix)	"2019-12-13"^^<
3	<https://mini.pw.ed...	All the Time - Don Diablo Remix	Zara Larsson	All the Time (Don Diablo Remix)	"2019-07-05"^^<
4	<https://mini.pw.ed...	Call You Mine - Keanu Silva Remix	The Chainsmokers	Call You Mine - The Remixes	"2019-07-19"^^<
5	<https://mini.pw.ed...	Someone You Loved - Future Humans Re...	Lewis Capaldi	Someone You Loved (Future Humans Remix)	"2019-03-05"^^<
6	<https://mini.pw.ed...	Beautiful People (feat. Khalid) - Jack Wins ...	Ed Sheeran	Beautiful People (feat. Khalid) [Jack Wins R...	"2019-07-11"^^<
7	<https://mini.pw.ed...	Never Really Over - R3HAB Remix	Katy Perry	Never Really Over (R3HAB Remix)	"2019-07-26"^^<
8	<https://mini.pw.ed...	Post Malone (feat. RANI) - GATTÜSO Remix	Sam Feldt	Post Malone (feat. RANI) [GATTÜSO Remix]	"2019-08-29"^^<

Figure 9: Songs released after 01/01/2015.

```
SELECT ?song_id ?song_name ?artist_name ?album_name  ?popularity WHERE {
  ?song_id rdf:type mo:Track ;
       foaf:name ?song_name ;
       spo:album ?album_id ;
       spo:artist ?artist_id ;
       spo:popularity ?popularity .
  ?artist_id foaf:name ?artist_name .
  ?album_id foaf:name ?album_name .
}
ORDER BY DESC(?popularity)
```

Table | Response | 28352 results in 3.922 seconds | Simple view ☐ Ellipse ☑ | Filter query results | Page size: 50

	song_id	song_name	artist_name	album_name	popularity
1	<https://mini.pw.ed...	Dance Monkey	Tones and I	Dance Monkey (Stripped Back) / Dance Mo...	"100"^^<http://www...
2	<https://mini.pw.ed...	ROXANNE	Arizona Zervas	ROXANNE	"99"^^<http://www.w3...
3	<https://mini.pw.ed...	Blinding Lights	The Weeknd	Blinding Lights	"98"^^<http://www.w3...
4	<https://mini.pw.ed...	Memories	Maroon 5	Memories	"98"^^<http://www.w3...
5	<https://mini.pw.ed...	Circles	Post Malone	Hollywood's Bleeding	"98"^^<http://www.w3...
6	<https://mini.pw.ed...	The Box	Roddy Ricch	Please Excuse Me For Being Antisocial	"98"^^<http://www.w3...
7	<https://mini.pw.ed...	Tusa	KAROL G	Tusa	"98"^^<http://www.w3...
8	<https://mini.pw.ed...	Don't Start Now	Dua Lipa	Don't Start Now	"97"^^<http://www.w3...

Figure 10: Songs, sorted by their popularity.