Final Project

Group 6

Hari Joshithaa Aghilah Senthilprathiban, Mara-Iuliana Dragomir, Andreea Giurgiu, Rafaëlla van Nee

Goal

The main objective of this project is to provide more insights to journalists about the main artists in the music industry. Nowadays, music has become a major part of everyone's lives. People listen to music usually to feel calm or relaxed or even to just lift their spirits. This has led to a great rise in the music industry, there are many new rising artists, different genres, new songs and albums, this led us to explore more into the field of music. Moreover, due to the rise of the music industry, there is more and more news about music. Thus, we decided to explore the field of music and help journalists answer the question: Who are the top artists of the world?

Answering this question will help journalists find out who are the most popular artists based on their revenue, song prizes, how collaborative they are and on their experience. This data will help the journalists to know on which artist they should focus to write more about, as people tend to be more interested in the most popular ones.

Hence, we are interested in exploring and analysing the following data:

1) Top 10 artists which have the highest number of songs which are intitulated as an award winner. In order for a song to be an award winner, it needs to have at least one award.

- This information will help the journalist to write about the best artists as well as explaining how many award songs a top artist usually should have. A journalist could use this information to write a story about the most successful artists in the music industry. Additionally, this information could be used to compare different artists.

2) Top 10 artists who have the most experience. An artist's experience is determined by how many years they have spent in the field and how many songs/albums they have published.

- This dataset will give more input for the journalist to understand how much experience a successful singer has. At the same time the journalist can easily see who are the newest artists in the world of music. This way, a journalist can write about any patterns or trends that can be discerned from the data. For example, the journalist might notice that all of the top 10 artists have a lot of experience performing live, and this could be used to support a story about the importance of performance experience for up-and-coming artists.

3) Top 3 artists who have the largest variety of genres.

- This will give a great input of the variety a singer has. Usually artists focus just on one gender, so having the artist that is able to sing a different one will create a greate article. Additionally, the journalist can see which artists are the most versatile and popular among a wide range of listeners. This information could be used to explore how these artists are able to appeal to such a wide range of audiences, or what factors might contribute to their success.

4) The artist with the highest net worth.

- This information will suggest who are the most successful artists. A journalist could be interested about the wealth disparities within the music industry and this could be used as a way to compare and contrast different artists.

5) Top 10 artists who wrote the most songs

- This dataset will help the journalist to understand which artist really enjoys being part of the process of writing songs and making their song based on their story. This way, they can see who are the most prolific songwriters which could be used as part of an article about the music industry.

6) Top 10 artists who have the most collaborations

- This will give many artists who enjoy working together and forming collaborative relationships. For journalists, this would be a good source if they are interested in writing about how certain artists are more likely to collaborate with each other and how this affects their music. Alternatively, a journalist could be interested in writing how artists are more likely to collaborate with other artists from different genres, and how this affects the music that they create.

7) Top 10 artists who have the least collaborations

- This will give a great input of which artist enjoys creating collaboration with others and creating relations between them. At the same time this is a great source from which journalists could understand if 2 artists are good friends and can work together.

8) Which is the country with the most top artists?

- This information will help in order to determine which country has the most talented singers in the world. A journalist might be interested in the globalisation of music and exploring why certain countries produce more successful artists than others. This way the journalist can write about how the music industry is shifting and changing, and wants to use the data to support their claims. They could also show how music from different countries is becoming more popular around the world.

Stakeholders

Based on the goal we plan to achieve through our ontology, we can derive two main stakeholders of our system. The first one is represented by journalists, which would use the data analysis pipeline in order to report the musician ranking and all sorts of data in articles, and in order to find more popular subjects to write about (i.e. the most popular artists) in order to spark the interest of their readers. And the second one is represented by the readers of these articles, which are in fact people, of any age, gender, ethnicity, background, etc., who are interested in music and want to read about their favourite musicians, or about rankings of musicians based on specific criteria (financial success, experience, music genre(s), country, etc.).

Design and Walkthrough

First to create our ontology, we looked at previously created ontologies on the web inorder to be able to reuse the data available. We first thought of general classes and properties which are dominant in the field of music. The, we created a rough outline of a few classes such as Artist, Manager, Song, Album, and Band as well as some general properties such as partOfAlbum, partOfBand, playedBy, writtenBy, etc to have a general idea of linking the properties to the classes. We found two vocabularies online namely FOAF and DBO, which are mentioned later in the Integrating Existing Ontologies section and mapped relevant concepts from the vocabularies to our ontology. We also found four datasets in the form of CSV files, which are mentioned later and cleaned those datasets by removing rows with no information in them as well as by removing columns with information which was not relevant to our ontology. Then, we merged all the datasets together and added the relevant instances into our ontology and created additional classes in order to make our created ontology coherent with the data available in the CSV files. Additionally, we also found one external SPARQL endpoint, namely DBpedia inorder to be able to gather additional information such as the networth of the artists, or the number of awards won or nominations for a particular artist. Furthermore, we also wrote equivalent class restrictions for relevant classes in order to be able to infer class membership for the added instances. In addition, property characteristics were also added to our properties to be able to infer results from the inferred properties. Finally, after adding all this information in Protege, we ran the Pellet reasoner and obtained relevant inferences.

We created multiple queries, each of them pertaining to each of the goals mentioned earlier and we used the results of each of the queries to create visualisations for the journalists. For the visualization of our data we chose two main types of charts, namely bar charts and pie charts. Bar graphs are more suitable for viewing the best artists in ascending or descending order based on the specific criteria, while also having numerical data be visible. For example, we use this

method for portraying the number of awards, the number of written songs and the lowest numbers of collaborations. Meanwhile, pie charts are better for cases when it is needed to visually compare the data based on proportions instead of checking the numbers. This visualisation style is used for the number of released songs, genre variety, highest number of collaborations, and the countries with the highest number of popular artists. In addition to all of these, we also chose a secondary visualisation for the countries with top artists, namely a map, where the number of successful artists is represented by a blue circle, because it's intuitive and simple to read, as well as aesthetically pleasing.

Domain and Scope

We must first define our domain and ontology's scope in order to construct our ontology, which will serve as the foundation for our analysis. The main goal of this ontology is to help journalists to know on which artist they should focus to write more about, as people tend to be more interested in the most popular ones. Thus our main domain is music and our scope to explore this domain is to cover topics such as the artists with the highest number of awards, the years they have been active, the variety of genres of songs by each artist, artists with the highest net worth, the artists who wrote the most number of songs, the most and least collaborative artists and the country where the artists are from. These topics are crucial and will help journalists portray and provide the readers with a clear overview of all the important factors and the different singers in each factor. Moreover, using these scopes can help give an overview of the main artists in the music field.

Methodology

The approach we used to create our ontology was a middle-out approach. We first established the main ideas and concepts in the field of music, following a top-down approach. Then, we tried to establish these concepts into classes and properties based on whether the concepts worked better as nouns or verbs. These included very general terms such as Song, Artist, Album, playedBy, partOfAlbum. Finally, following a bottom-up approach we also looked at data available in previous existing ontologies as well as external sources such as the CSV files and DBpedia mentioned earlier. Thus, we also took the instances available also into consideration while trying to create out properties as to be able to include much more useful information such as the the Genre, website of the artists, other social media of the artists, top artists' lead streams, number of listeners for each artist as well as the number of scroggles for each artist. Therefore, we used a middle-out approach to create our music ontology by starting with the most fundamental topics and then working towards more abstract and specific terms.

Our Ontology

Our ontology consists of classes, properties (object and data type) as well as instances. We have one class for Artist and another one for Manager and these two are subclasses of the class Person. We also have separate classes for Song, Album, Award, Genres, and the Country of each singer. The data properties we have are the hasName relation, which links an instance of Person to the string value of their name in english. We also have the startedActivityIn relation which shows for how many years that an Artist has been singing songs. We also have the hasNetWorth relation which shows the total networth of an Artist. The object type properties in our ontology are the playedBy, writtenBy, managedBy, partOf, won, hasGenere, bornIn and collaborateWith. These object properties help link instances of two different classes. For example, the playbedBy relation links an Artist and to a Song which is played by that specific Artist.

Conceptualization of the Ontology

In our ontology for music, we have 19 classes including main classes as well as sub classes. In addition, we have 14 properties out of which 11 are object properties and 3 are data type properties. The list of classes and properties can we found below:

List of Classes:

- Person
    - Manager
    - Artist
        - SoloArtist
        - BandMember
        - Guitarist
        - Vocalist
        - Bassist
        - Drummer
        - Keyboardist
- Song
    - NominatedSong
    - AwardedSong
- Band
- Album
- Award
- Genre
- Country
- Tour

Object Properties

Table 1 below represents all the object properties in our ontology. The object properties can be found in the Predicate column and we also added the columns with subject and predicate which shows which classes can be the subjects and objects for each of the object properties.

Subject	**Predicate**	**Object**
Song	playedBy	Artist/Band
Song	writtenBy	Artist/Band
Artist/Band	managedBy	Manager
Song	partOfAlbum	Album
Band Member	partOfBand	Band
AwardedSong	wonAward	Award
Song	hasGenre	Genre
Artist	bornIn	Country
Artist/Band	collaboratedWith	Artist/Band
NominatedSong	nominatedForAward	Award
Artist/Band	playedInTour	Tour

Data Type Properties

Table 2 below represents the data type properties in our ontology. Again, the predicate represents the data type property and the subject refers to which classes can be used along with the property and the object refers to what values and data types the predicate can have.

Subject	**Predicate**	**Object**
Person	hasName	“Name” (literal)
Artist/Band	startedActivityIn	“Year” (nonNegativeInteger)
Person	hasNetWorth	“NetWorth” (nonNegativeInteger)

Figure 1 below illustrates the representation of our ontology. As described, the circles represent classes and the properties are represented by the arrows in the diagram. The rectangles represent literals and the data type of the values in the literals are shown inside the rectangle.

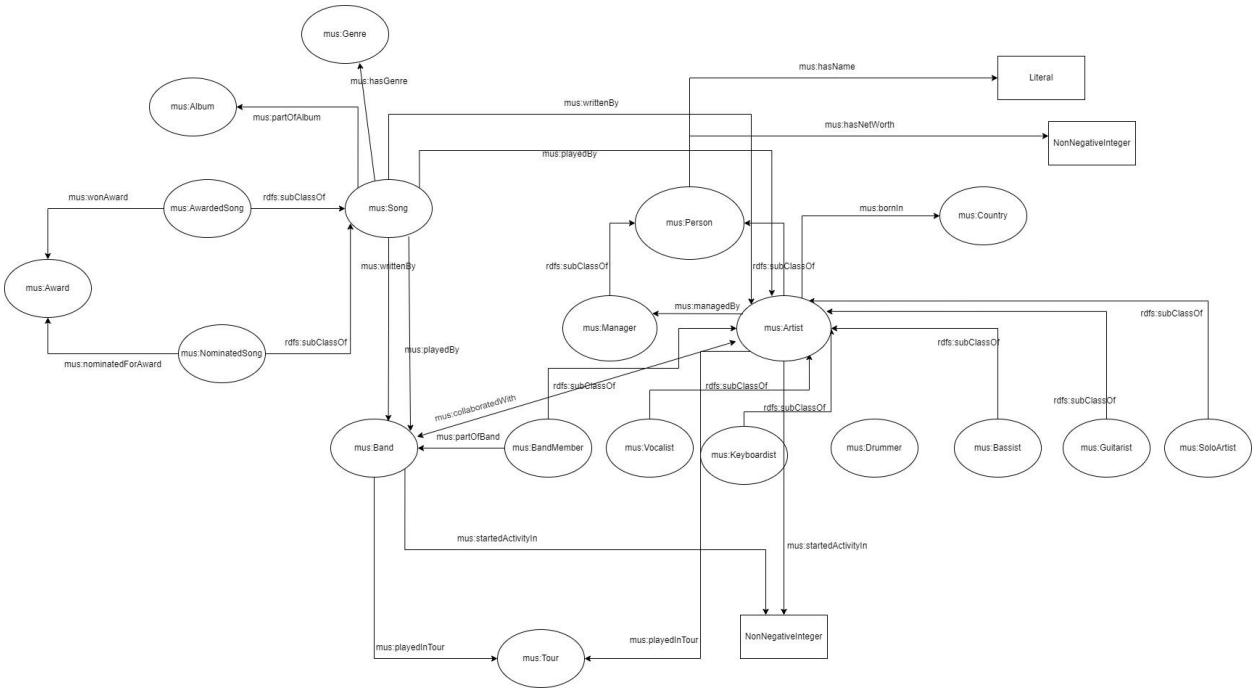


Figure 1: Representation of the Conceptualization of Our Ontology

Class Restrictions

We added the following three class restrictions to our ontology:

- BandMember Class is equivalent to an Artist that is part of some Band.
- NominatedSong Class is equivalent to a Song that was nominated for some Award.
- AwardedSong Class is equivalent to a Song that won some Award.

Integrating Existing Ontologies

A few of the ontologies we will use from the web are DBO, DBP, DBR, OWL, RDF/RDFS and FOAF. We chose to use DBO, DBP, and DBR because it has a lot of relevant and useful terms specific for our particular domain. Moreover, DBO, DBP, and DBR are also reliable as they contain useful information and data which supports our project. DBO is specific for ontologies, DBP is specifically for properties and DBR is specifically stated for DBpedia resources. We will also use the other general ontologies such as OWL, RDF and RDFS as these are general vocabularies which can help define general characteristics such as the type relation, the label relation, the equivalent class/property and subclass/subproperty relation. This will make it easier to link different sources and make much more clear and generalised relationships between classes, properties and instances. Additionally, we will also use the FOAF vocabulary in our ontology as this is also commonly used to define people (singers in our case) and describe their activities.

Integrating External Sources

We will use 2 external sources for our project, namely DBpedia and CSV files from various sources. One of the external sources we will use for this project is DBpedia. DBpedia is an external SPARQL endpoint because it is already in RDF. DBpedia has many relevant predicates as well as subjects which are specific for our specific domain. Some relevant predicates include dbp:writer which links a specific artist to the song he/she has written, dbp:mostNominations which show how successful each artist is. This allows us to focus more on our goals as DBpedia has a lot of relevant information about the net worth, awards won, the singers with the most number of songs, etc and this will help journalists analyse and interpret the music industry with much more data in a much more specific way.

The other external source which we will use for our ontology which is not in RDF are CSV files from various sources such as Kaggle, OpenAxis and GitHub. These CSV files include additional information such as the genre of the songs, the name of the albums, the artists for each song as

well as the number of listeners for each artist. This information for each of the artists is important as it helps journalists find out more about the top artists who the general public likes to listen to or the specific genres the top artists focused on, or the lead stream's for a particular artist. The four datasets used in our ontology can be accessed using the following links below:

1)https://gist.github.com/mbejda/9912f7a366c62c1f296c

2) https://app.openaxis.com/data/4044

3) https://www.kaggle.com/datasets/pieca111/music-artists-popularity

4)https://www.kaggle.com/datasets/leonardopena/top-spotify-songs-from-20102019-by-year

Data Reuse

We integrated the two ontologies mentioned above using owl restrictions. The two ontologies we integrated are DBpedia and foaf. We obtained these from the internet in the form of turtle files and then imported them into Protege along with our ontology. The integration and mapping of the two ontologies can be seen below. We mapped equivalent classes and properties as well as subclasses and subproperties from both the ontologies into our main ontology.

The four datasets that were also added in graphdb are in the form of csv files which contain relevant data about music artists. The csv files were uploaded and the columns were mapped as properties except the first one and the artists were mapped as instances. We added the triples into our ontology using Ontorefine. The subject of the triple is the instance and relates to the object as the property corresponding to that value in the row of the instance. Data from the CSV file was then inferred to the ontology's construction using rdf:type.

Mapping Constructs

Relevant DBO entities

- dbo:MusicalArtist (equivalent to mus:Artist)
- dbo:MusicGenre (equivalent to mus:Genre)
- dbo:musicBy (equivalent to mus:playedBy)

Relevant foaf entities:

- Foaf:Person (equivalent to mus:Person)
- Foaf:knows (mus:collaboratedWith is a subclass of foaf:knows)
- Foaf:name (equivalent to mus:name)

Inferences

The data was extracted into a file and then we imported it into protege. We have inferred instances for class memberships for songs and singers. Then, after running the reasoner, we obtained the following inferences for instances:

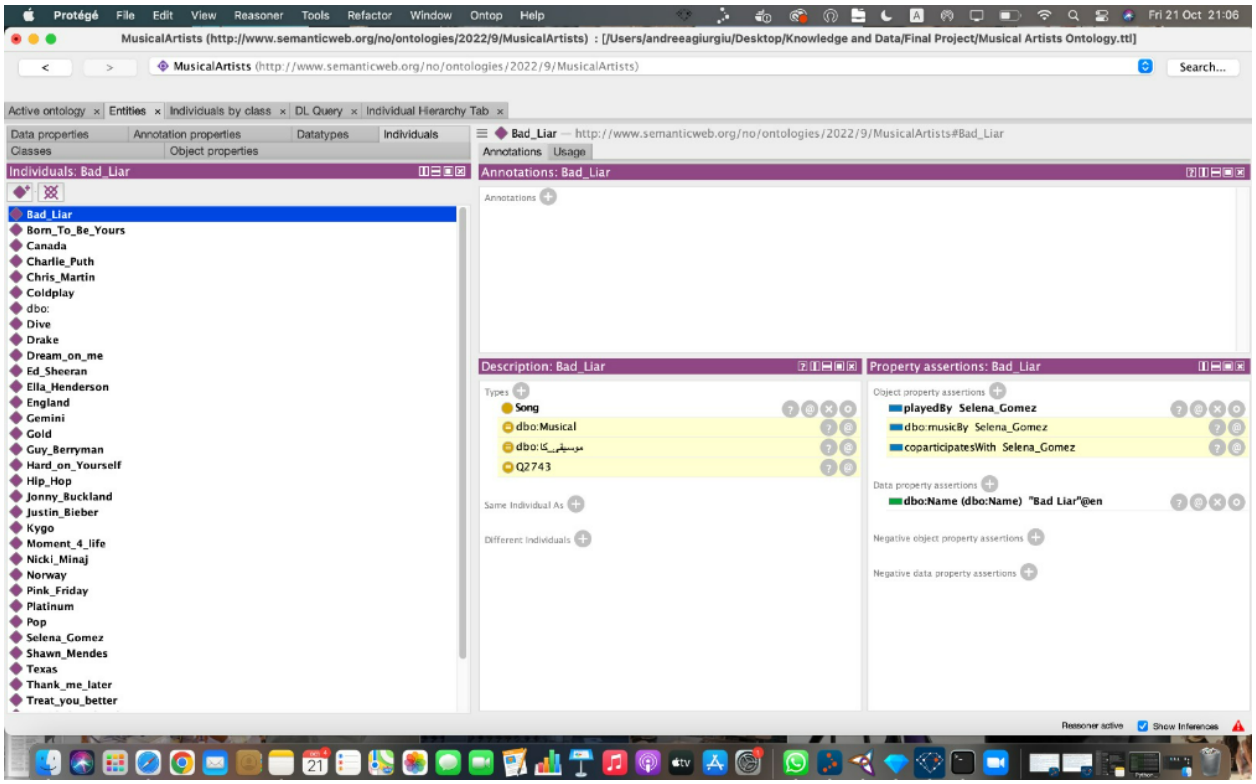


Figure 2: Inferences Showing Mapping Between Our Ontology and DBO

Figure 2 above shows inferences which have been obtained due to the mapping between our ontology and DBO. As stated in our mapping constructs, these inferences are obtained because the class dbo:Musical and others in Figure 2 are equivalent classes of our class mus:Song. This also means that the properties of those classes will also be inferred as dbo:musicBy and coparticipatesWith for the our object property mus:playedBy.

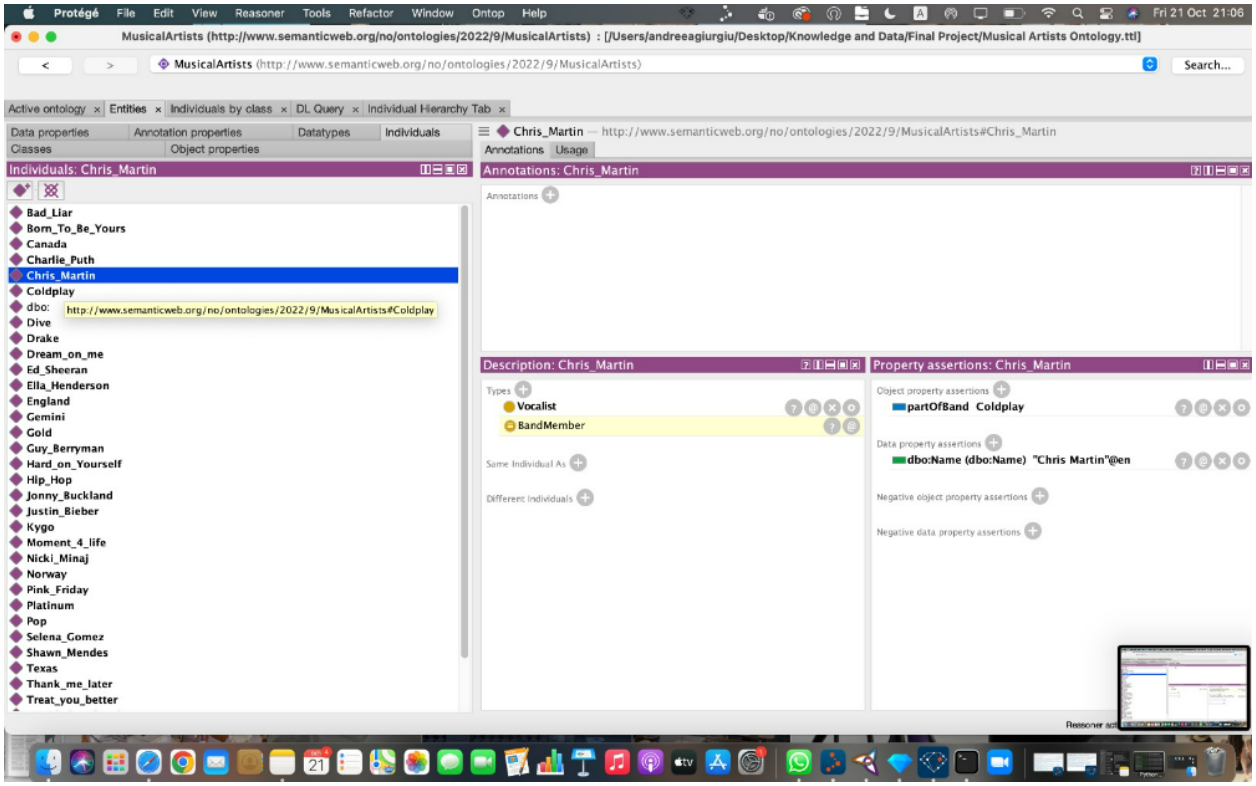


Figure 3: Inferring Class Membership for BandMember

Figure 3 above shows the class membership relation being inferred for the class BandMember. This is because we added a necessary and sufficient condition (equivalent class) which states that if a person is a part of a band, then he/she is a band member. As seen above, Chris_Martin is related to Coldplay by the property partOfBand, this then means that Chris_Martin is an instance of the class BandMember.

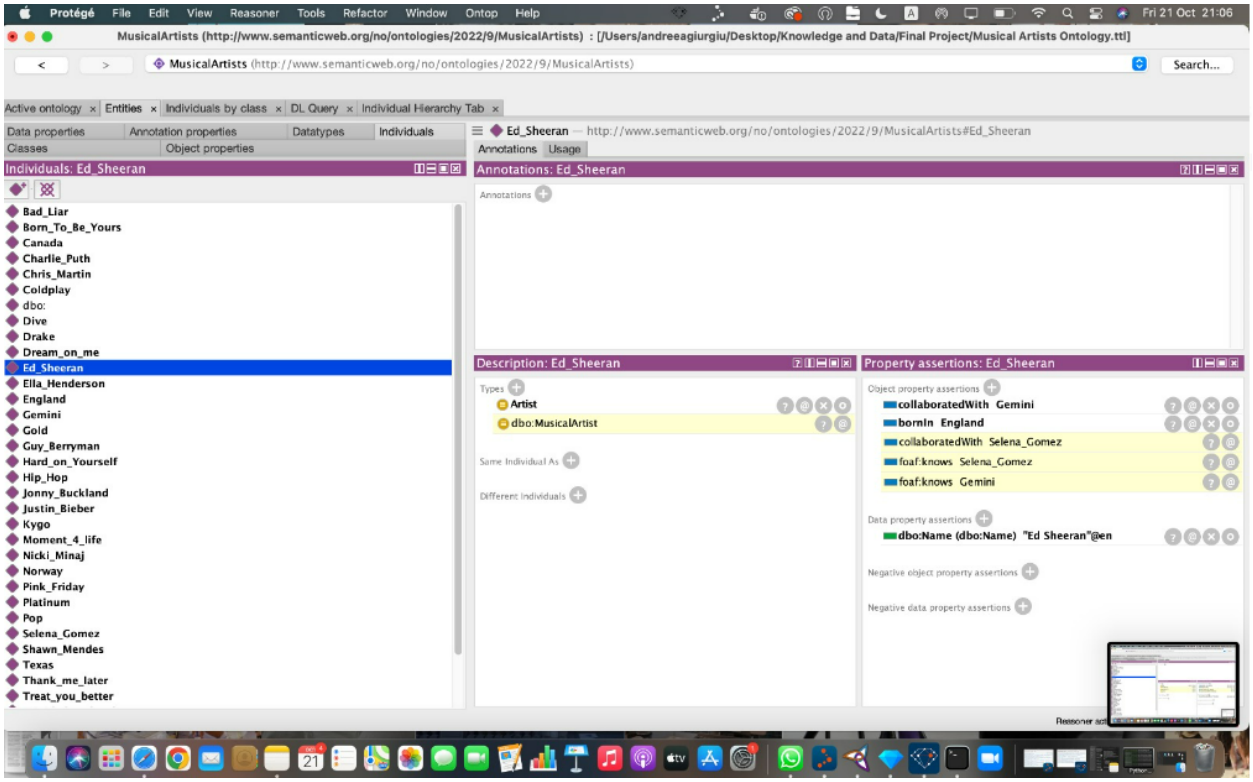


Figure 4: Inferring Property Relations for Artists

Figure 4 illustrates the protege screenshot which shows that an artist knows other artist(s) because they collaborated at least once, that an artist collaboration is mutual (if an artist collaborated with another artist, then the latter also collaborated with the former). This is because mus:collaboratedWith is a sub property of foaf:knows, thus if two artists collaborated with each other, then they must also know each other according to the RDFS rules.

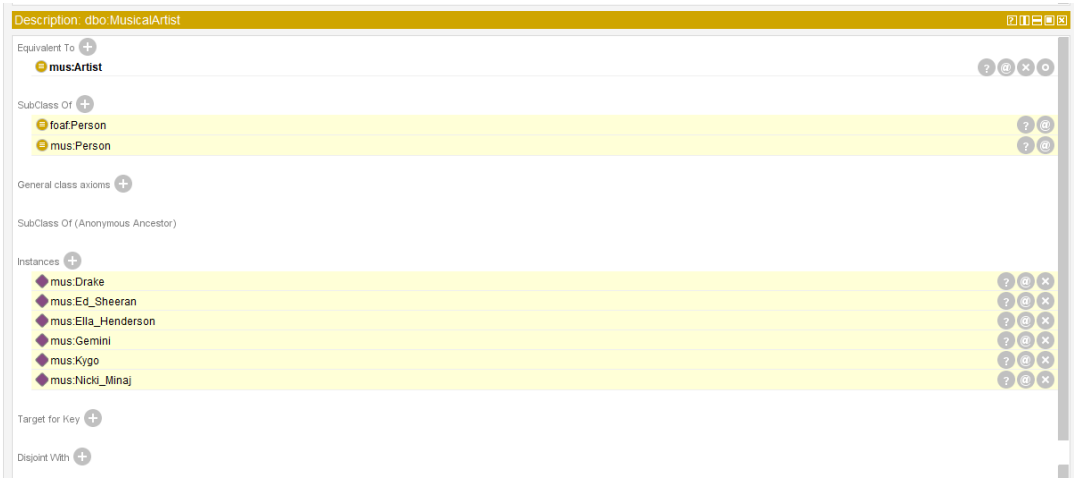


Figure 7: Inferring Class Membership for Instances of Artist

Figure 7 demonstrates the protege screenshot showing the mapping between our ontology & DBO. All the instances of the class of Artist have been inferred as rdf:type mus:Artist due to necessary and sufficient conditions added, which have been mentioned earlier.

SPARQL Queries

We created multiple queries corresponding to each of our goals and we extracted information from an external SPARQL Endpoint. We extracted information about the number of awards won by the top 10 artists, the artists who have the most experience, the artists with songs from various genres, the highest net worth of the artists, the artists who also wrote the most songs in addition to being singers, the most and least collaborative artists, and the country of origin for each of the artists. These queries contain important information which is useful for all the stakeholders.

This query gives the number of awards for each artist. This is important because it gives us the most successful artists, i.e. the artists who have won the most awards.

```
%%sparql http://localhost:7200/repositories/FinalProject -s courseload
PREFIX ex: <http://example.com/kad/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <https://dbpedia.org/property/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX mus: <http://www.semanticweb.org/no/ontologies/2022/9/MusicalArtists#>

Select ?artist (Count(?award) as ?wonAwards)
WHERE
{

    ?artist rdf:type mus:Artist.
    ?artist dbo:Name ?name.
    Service <https://dbpedia.org/sparql> {
        ?singer foaf:name ?name.
        ?singer dbo:award ?award.
        ?award dbo:wikiPageWikiLink ?wonAwards

    }

}
Group By ?artist
Order By desc(?wonAwards)
```

This query returns the top 10 artists who have released the most number of songs. This gives the journalists an intuition about the artists who have more songs released than the others.

```
%%sparql http://localhost:7200/repositories/FinalProject -s courseload
PREFIX ex: <http://example.com/kad/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <https://dbpedia.org/property/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX mus: <http://www.semanticweb.org/no/ontologies/2022/9/MusicalArtists#>

Select ?artist (Count(?Song) as ?HitNumber)
WHERE
{

    ?artist rdf:type mus:Artist.
    ?artist dbo:Name ?name.
    Service <https://dbpedia.org/sparql> {
        ?singer foaf:name ?name.
        ?Song dbo:artist ?singer.

    }

}
Group By ?artist ?award
Order By desc(?HitNumber)
LIMIT 10
```

This query below gives the top 3 artists with the most number of different genres each he/she has sung in. This gives more information to the journalists about the artists who have the most variety of songs.

```
%%sparql http://localhost:7200/repositories/FinalProject -s courseload
PREFIX ex: <http://example.com/kad/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <https://dbpedia.org/property/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX mus: <http://www.semanticweb.org/no/ontologies/2022/9/MusicalArtists#>

Select ?artist (Count(distinct ?genera) as ?HitNumber)
WHERE
{

    ?artist rdf:type mus:Artist.
    ?artist dbo:Name ?name.
    Service <https://dbpedia.org/sparql> {
        ?singer foaf:name ?name.
        ?Song dbo:artist ?singer.
        ?Song dbo:genre ?genera.

    }

}
Group By ?artist
Order By desc(?HitNumber)
LIMIT 3
```

This query shows the artist with the highest net worth. This will give more information to the journalists about the artist with the highest revenue and who has earnt the most as a singer.

```
%%sparql http://localhost:7200/repositories/FinalProject -s courseload
PREFIX ex: <http://example.com/kad/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <https://dbpedia.org/property/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX mus: <http://www.semanticweb.org/no/ontologies/2022/9/MusicalArtists#>

Select ?artist ?networth
WHERE
{

    ?artist rdf:type mus:Artist.
    ?artist dbo:Name ?name.
    Service <https://dbpedia.org/sparql> {
        ?singer foaf:name ?name.
        ?singer dbo:networth ?networth

    }

}
Order By desc(?networth)
LIMIT 1
```

This query shows the top 10 singers with the most number of songs they have written. This provides more information on the singers who also like to write songs, i.e. who are both singers and songwriters.

```
%%sparql http://localhost:7200/repositories/FinalProject -s courseload
PREFIX ex: <http://example.com/kad/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <https://dbpedia.org/property/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX mus: <http://www.semanticweb.org/no/ontologies/2022/9/MusicalArtists#>

Select ?artist (Count(?Song) as ?writtensongs)
WHERE
{

    ?artist rdf:type mus:Artist.
    ?artist dbo:Name ?name.
    Service <https://dbpedia.org/sparql> {
        ?singer foaf:name ?name.
        ?Song dbo:writer ?singer.

    }

}
Group By ?artist
Order By desc(?writtensongs)
LIMIT 10
```

This query below shows the artists with the most collaborations. This also gives an indication to the journalists as to which artists know a lot of other artists and have larger connections in the music industry.

```
%%sparql http://localhost:7200/repositories/FinalProject -s courseload
PREFIX ex: <http://example.com/kad/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <https://dbpedia.org/property/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX mus: <http://www.semanticweb.org/no/ontologies/2022/9/MusicalArtists#>

Select ?artist (Count(?artists) as ?collaborations)
WHERE
{

    ?artist rdf:type mus:Artist.
    ?artist dbo:Name ?name.
    Service <https://dbpedia.org/sparql> {
        ?singer foaf:name ?name.
        ?artists dbo:associatedMusicalArtist ?singer

    }

}
Group By ?artist
Order By desc(?collaborations)
LIMIT 10
```

This query below shows the artists with the least collaborations. This gives an indication to the journalists about the artists who would prefer to sing alone in the field of music.

```
%%sparql http://localhost:7200/repositories/FinalProject -s courseload
PREFIX ex: <http://example.com/kad/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <https://dbpedia.org/property/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX mus: <http://www.semanticweb.org/no/ontologies/2022/9/MusicalArtists#>

Select ?artist (Count(?artists) as ?collaborations)
WHERE
{

    ?artist rdf:type mus:Artist.
    ?artist dbo:Name ?name.
    Service <https://dbpedia.org/sparql> {
        ?singer foaf:name ?name.
        ?artists dbo:associatedMusicalArtist ?singer

    }

}
Group By ?artist
Order By ?collaborations
LIMIT 10
```

This query below shows the countries that most artists are from. This allows journalists to analyse more about the music industry and find out if certain countries have the most singers and the countries which dominate the music industry.

```
%%sparql http://localhost:7200/repositories/FinalProject -s courseload
PREFIX ex: <http://example.com/kad/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <https://dbpedia.org/property/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX mus: <http://www.semanticweb.org/no/ontologies/2022/9/MusicalArtists#>

Select ?country (Count(?artists) as ?bornincontry)
WHERE
{

    ?artists rdf:type mus:Artist.
    ?artists mus:bornIn ?country.




}
Group By ?country
Order By ?bornincountry
```

Inferred SPARQL Queries

We also wrote additional SPARQL Queries which produced inferred results when we switched the OWL reasoner ON. These queries can be seen below:

This query below first constructs two different sets of triples into our ontology, namely that if an artist is part of a band then, he/she is a BandMember. Then, it selects the artists who are part of a band.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX mus: <http://www.semanticweb.org/no/ontologies/2022/9/MusicalArtists#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dbp: <https://dbpedia.org/property/>

CONSTRUCT{?artist rdf:type mus:BandMember}
WHERE
{
    ?artist rdf:type mus:Artist.
    ?artist mus:partOfBand ?band.
}
```

This query adds the information obtained from the previous query and adds them to our ontology:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX mus: <http://www.semanticweb.org/no/ontologies/2022/9/MusicalArtists#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
CONSTRUCT{?artist rdf:type mus:bandMember}
WHERE
{
    ?artist rdf:type mus:Artist.
    ?artist mus:partOfBand ?band.
}
```

The query returns the following results when the reasoner is turned ON:

	subject	predicate	object
1	mus:Chris_Martin	rdf:type	mus:BandMember
2	mus:Guy_Berryman	rdf:type	mus:BandMember
3	mus:Jonny_Buckland	rdf:type	mus:BandMember
4	mus:Will_Champion	rdf:type	mus:BandMember

The query below is similar to the first one but it searches for all the artists in our ontology and then finds out the name of the band they are a part of, if they are a bandMember and then constructs these triples.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX mus: <http://www.semanticweb.org/no/ontologies/2022/9/MusicalArtists#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
CONSTRUCT{?artist mus:partOfBand ?band}
WHERE
{
    ?artist rdf:type mus:Artist.
    ?artist dbo:Name ?Artist.
    Service <https://dbpedia.org/sparql> {
        ?singer foaf:name ?Artist.
        ?band dbo:bandMember ?singer . }
}
```

This query inserts the triples obtained from the previous one and adds them into our ontology.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX mus: <http://www.semanticweb.org/no/ontologies/2022/9/MusicalArtists#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
INSERT{?artist mus:partOfBand ?band}
WHERE
{
    ?artist rdf:type mus:Artist.
    ?artist dbo:Name ?Artist.
    Service <https://dbpedia.org/sparql> {
        ?singer foaf:name ?Artist.
        ?band dbo:bandMember ?singer . }
}
```

When the reasoner is turned ON, it returns the name of the Band the artist is from using the external SPARQL endpoint, namely DBpedia.

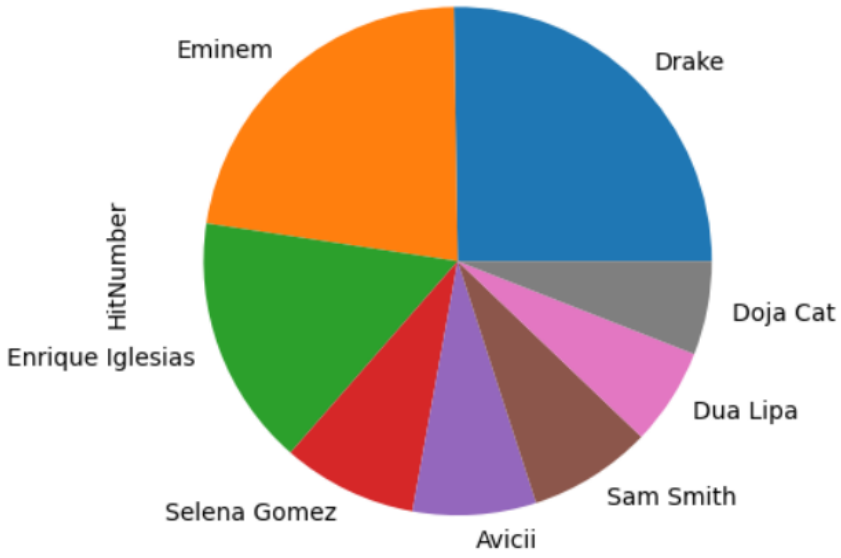
	subject	predicate	object
1	mus:Chris_Martin	mus:partOfBand	http://dbpedia.org/resource/Coldplay
2	mus:Eminem	mus:partOfBand	http://dbpedia.org/resource/Bad_Meets_Evil
3	mus:Guy_Berryman	mus:partOfBand	http://dbpedia.org/resource/Apparatjik
4	mus:Guy_Berryman	mus:partOfBand	http://dbpedia.org/resource/Coldplay
5	mus:Harry_Styles	mus:partOfBand	http://dbpedia.org/resource/One_Direction__One_Directi
6	mus:Inna	mus:partOfBand	http://dbpedia.org/resource/G_Girls__G_Girls__1
7	mus:Martin_Garrix	mus:partOfBand	http://dbpedia.org/resource/Area21
8	mus:Tiësto	mus:partOfBand	http://dbpedia.org/resource/Kamaya_Painters
9	mus:Will_Champion	mus:partOfBand	http://dbpedia.org/resource/Coldplay

This can again be verified by writing a SELECT query which returns artists who are part of a band, this query can be found in the Jupyter Notebook attached.

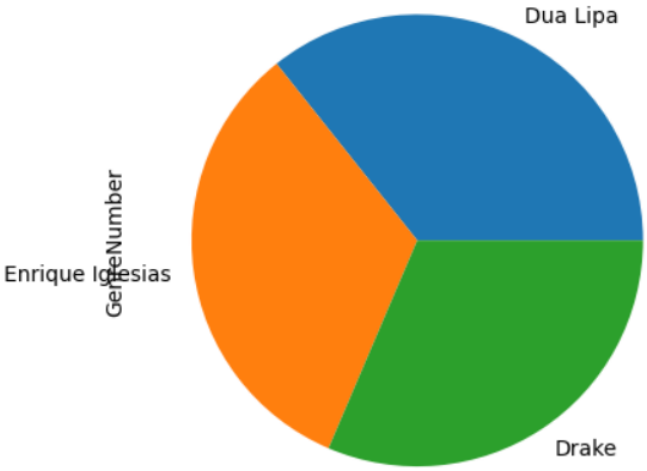
Analysis Pipeline and Conclusion

As mentioned earlier, the visualisations in the Jupyter Notebook created can be seen below. Each of the goals mentioned above are represented below in each of the visualisations. This makes it easier for the journalists to represent all the information to the readers in a much more effective way. This is because having visual representations allows people to quickly grasp the information represented rather than having to read an entire article.

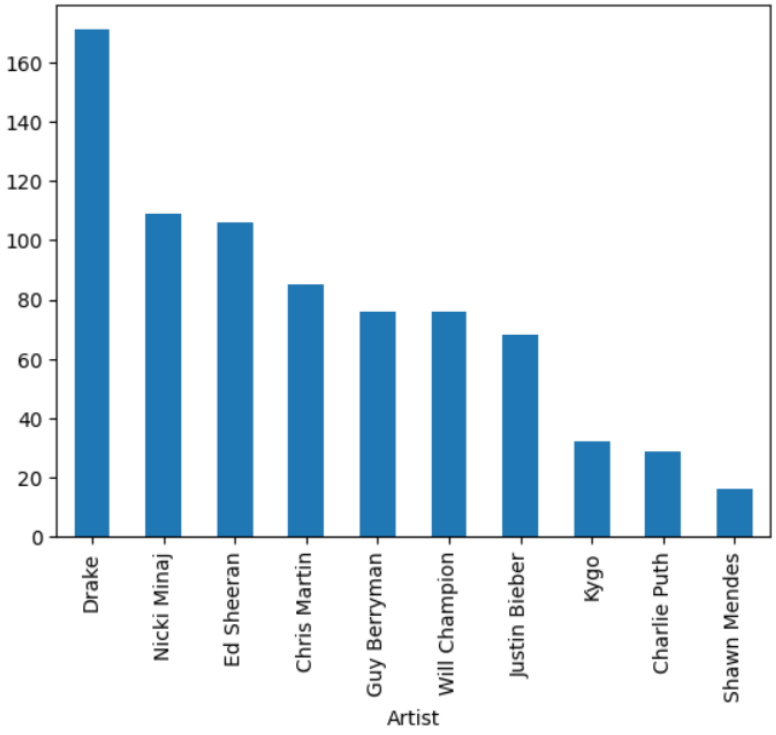
The figure below shows the number of songs sung by each artist. This information is represented as a pie chart and this makes it easier to compare the number of songs released for all artists in a much easier way by comparing the segments in the pie chart for the different artists.



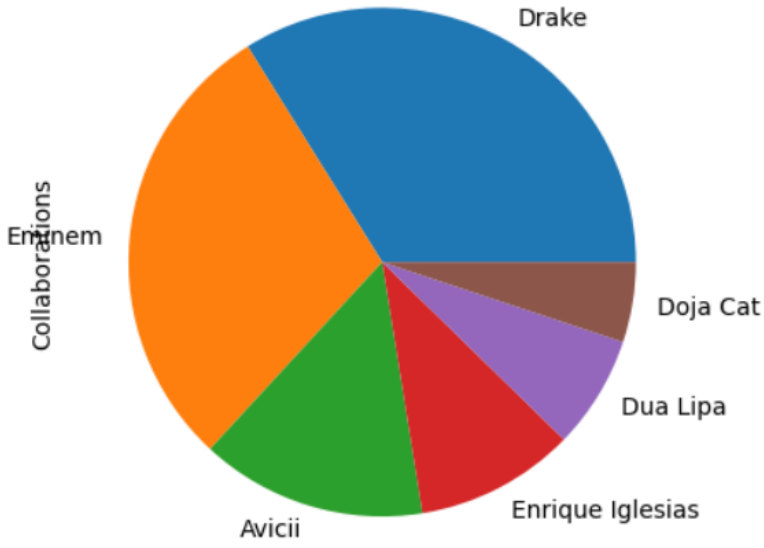
The pie chart below in Figure 9 represents the singers who have sung songs from many different genres. This information is useful and allows a much deeper understanding about the artists who have the songs with the most different variety.



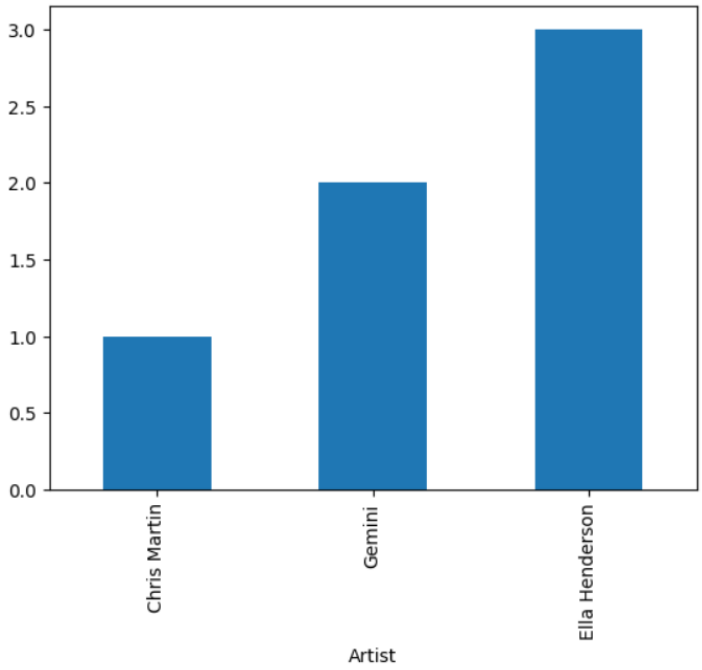
The bar graph below represents the number of songs written by each artist in the y-axis and the artist in the x-axis. This is represented in a bar graph as this allows the readers to easily identify the top 10 artists who have written the most songs by just looking at the length of the bar in the graph. This also allows us to easily identify the artist who has written the most number of songs.



The pie chart below shows the artists who have collaborated the most with other artists. This is represented in a pie chart because this makes it easier to spot the largest segment to find the singer who has collaborated the most with other artists.



The bar chart below shows the artists who have the least number of collaborations with other artists. This is represented in a bar chart because this makes it easier to spot the shortest bar to find the singer who has collaborated the least with other artists and thus has a smaller network in the music industry.



Finally, we also created a map as illustrated below where the circles represent the countries with the most number of artists.