Knowledge Graph: shutterBoxd

Design, Modelling, Uplift and Validation

Introduction

This project develops a Movie Knowledge Graph (shutterBoxd) integrating heterogeneous datasets from IMDb-style and Letterboxd exports, including metadata, cast lists, crew lists, reviews, ratings, certificates, production companies, and genre information.

The knowledge graph aims to provide a rich, queryable, reason-ready model of movies, people, reviews, and cinematic metadata. The report details the ontology design, RML-based uplift, instance creation, OWL/CHOWL-K diagrams, repository configuration, SPARQL queries, validation, and reflections.

Ontology

A domain ontology (movies.ttl) was developed using OWL, RDFS, and standard vocabularies.
The ontology defines movies, people, reviews, genres, production companies, and awards, along with datatype and object properties linking them.

The final ontology adheres to the chowlk.linkeddata.es conventions and was visualized using CHOWL-K.

1. Classes

The ontology contains 8 top-level classes and 2 subclasses, satisfying project requirements:

- Movie
- Person
- Genre
- Series
- ProductionCompany
- Certificate
- Review
- Award

Subclasses of Person:

- Actor
- Director

2. Object Properties

The ontology defines the following object properties, including symmetric, transitive, & inverse relations:

- actedIn / hasActor *(inverseOf)* – Links Actor ↔ Movie
- directed / hasDirector *(inverseOf)* – Links Director ↔ Movie
- hasGenre – Assigns Genre to Movie
- producedBy – Film → Production Company
- hasCertificate – Film → Certification
- hasReview / reviewOf *(inverseOf)* – Film ↔ Review
- hasAward / awardFor *(inverseOf)* – Film ↔ Award
- isPartOfSeries – Movie → Series
- similarMovie *(SymmetricProperty)* – Movie ↔ Movie
- connectedTo *(TransitiveProperty)* – Movie → Movie

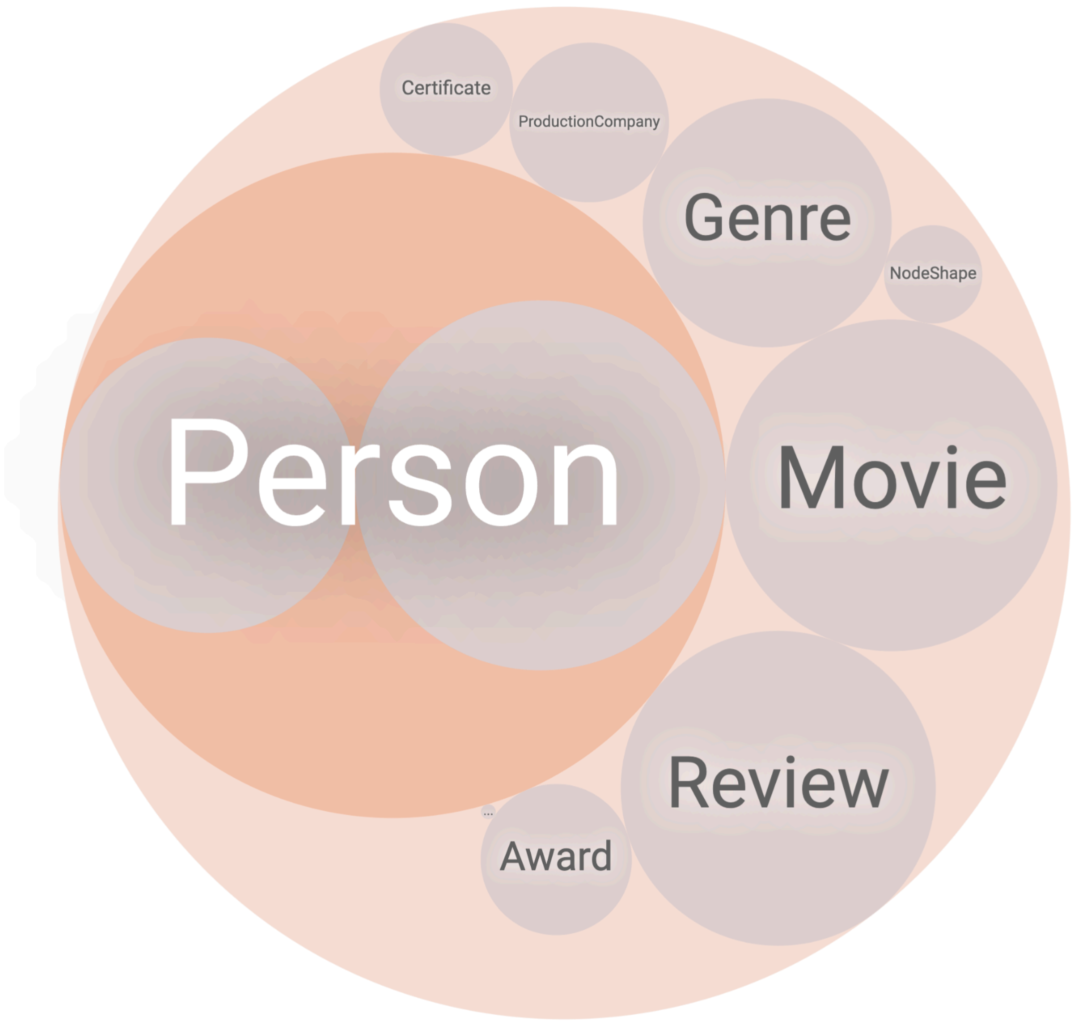


Fig-01: Class Hierarchy Diagram

Each object property in the ontology includes a clear domain and range specification to enforce semantic consistency. For example, actedIn has the domain moviekg:Actor and range moviekg:Movie, meaning only actors can act in movies; similarly, hasActor reverses this relation with domain moviekg:Movie and range moviekg:Actor. The property links a moviekg:Director to a moviekg:Movie, while hasGenre connects a moviekg:Movie to a moviekg:Genre. These domain–range declarations ensure that incorrect combinations of classes cannot occur and allow reasoners to infer types automatically based on property usage.

3. Data Properties

For attribute-level information, the ontology includes:

Movie attributes

- movieTitle
- releaseYear
- runtimeMinutes

- imdbRating
- overview
- metaScore
- voteCount
- grossRevenue

Person attribute

- personName

Genre / Certificate / Company / Award Attributes

- genreLabel
- certificateCode
- companyName
- awardLabel

Review attributes

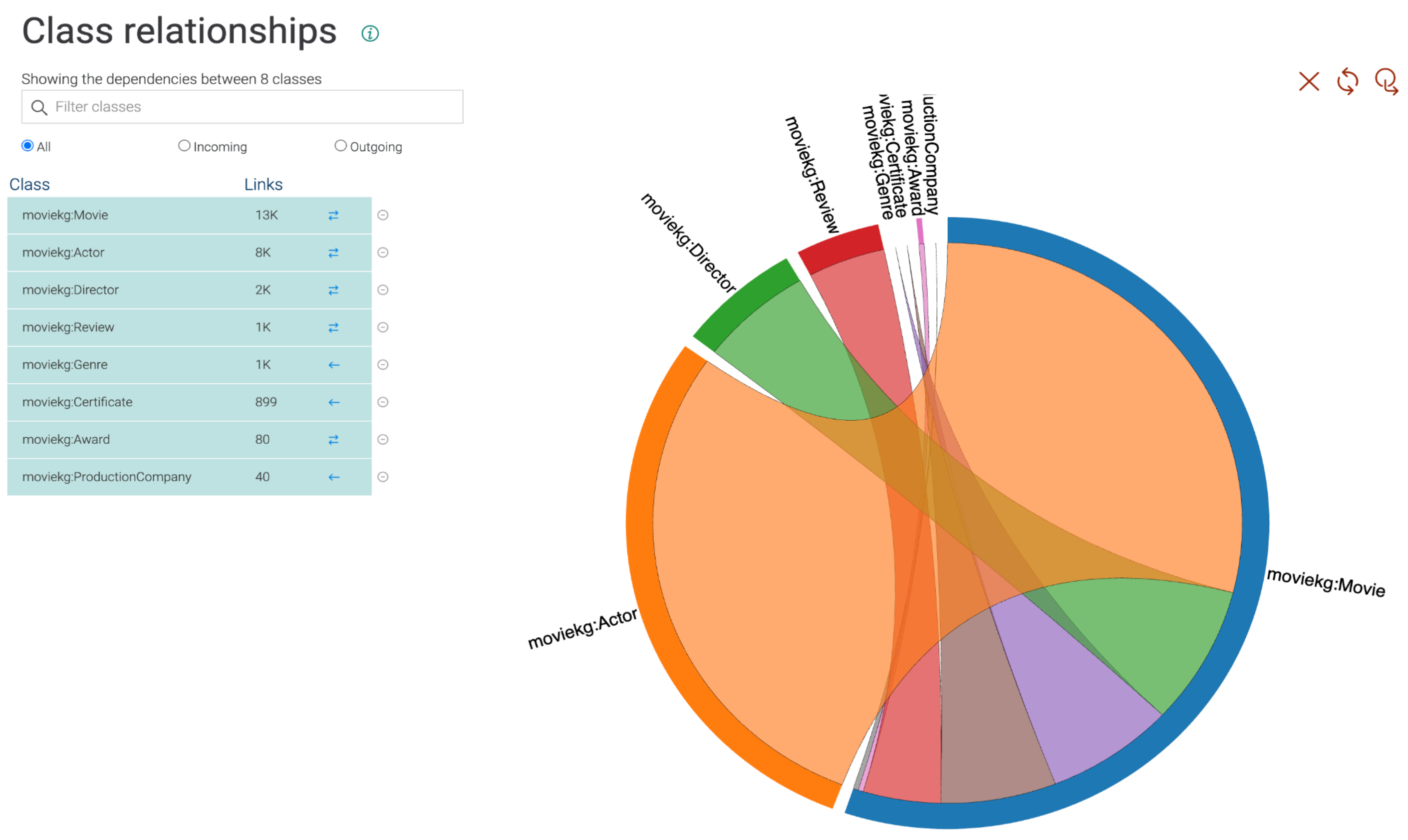
- ratingValue
- reviewText
- reviewDate
- watchedDate
- tagsText
- letterboxdUri

4. External Interlinks

To enable interoperability and semantic linking:

- Movies may reference IMDb URIs via owl:sameAs
- Letterboxd reviews include non-resolvable but stable URI anchors
- People can be linked to DBpedia/Schema.org IRIs (future extension)

These external alignments improve searchability and semantic enrichment.



Class	Links	
moviekg:Movie	13K	⇄
moviekg:Actor	8K	⇄
moviekg:Director	2K	⇄
moviekg:Review	1K	⇄
moviekg:Genre	1K	←
moviekg:Certificate	899	←
moviekg:Award	80	⇄
moviekg:ProductionCompany	40	←

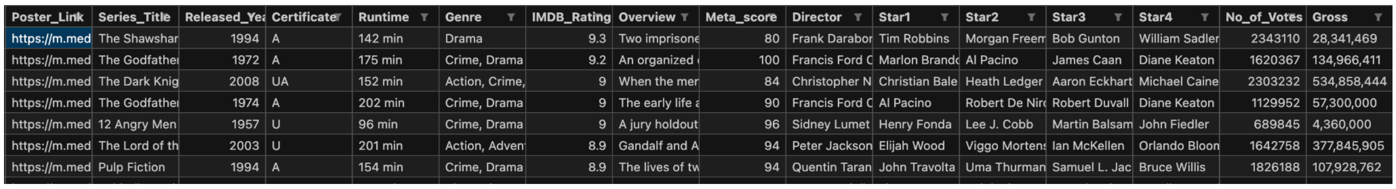
Fig-02: Class Relationship Diagram

Data Preparation and Mapping for Data Uplift

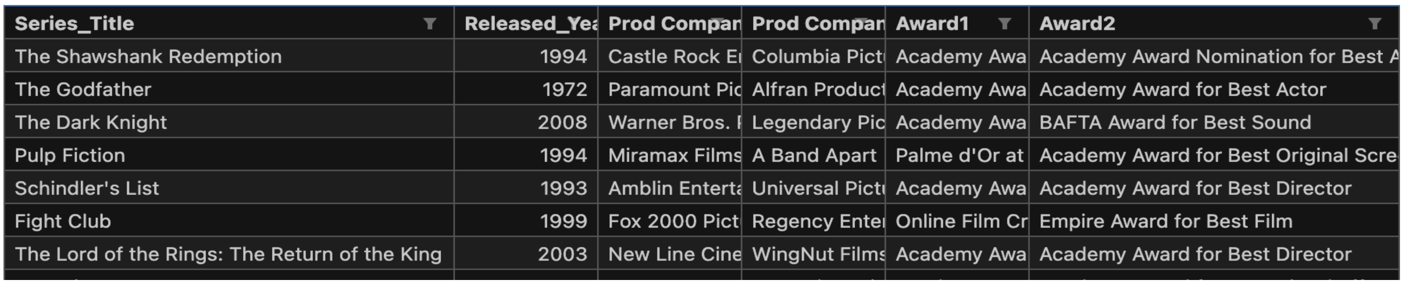
1. Data Preparation

The datasets consisted of:

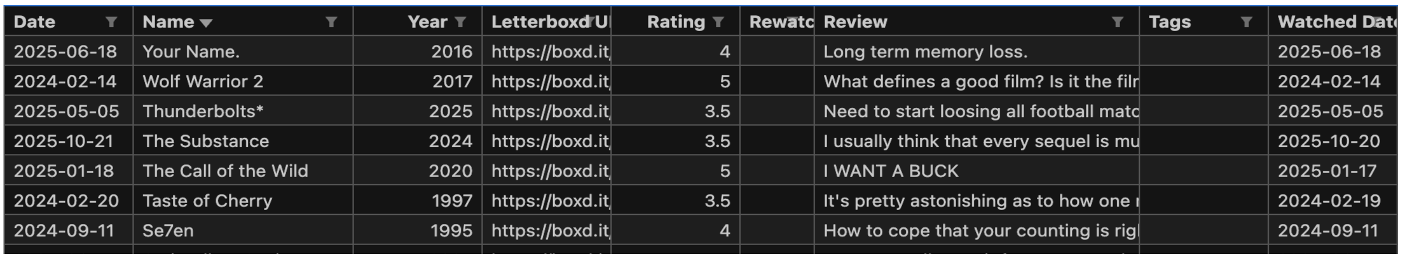
- imdb_top_1000.csv
- ratings.csv
- reviews.csv
- watched.csv
- watchlist.csv
- movies.ttl
- statements_from_csv.ttl
- movie_mapping.rml.ttl



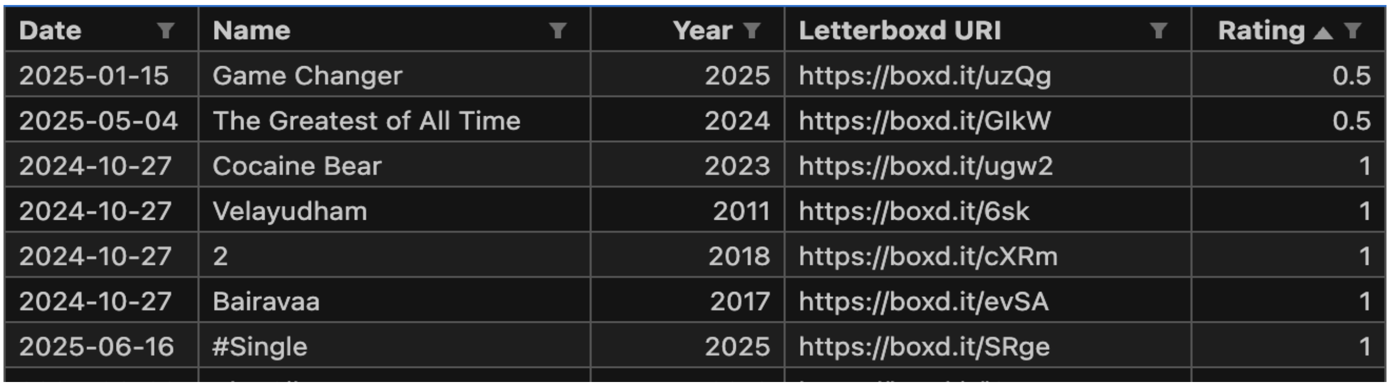
Poster_Link	Series_Title	Released_Yea	Certificate	Runtime	Genre	IMDB_Rating	Overview	Meta_score	Director	Star1	Star2	Star3	Star4	No_of_Votes	Gross
https://m.med	The Shawshar	1994	A	142 min	Drama	9.3	Two imprisone	80	Frank Darabor	Tim Robbins	Morgan Freem	Bob Gunton	William Sadler	2343110	28,341,469
https://m.med	The Godfather	1972	A	175 min	Crime, Drama	9.2	An organized	100	Francis Ford C	Marlon Brando	Al Pacino	James Caan	Diane Keaton	1620367	134,966,411
https://m.med	The Dark Knig	2008	UA	152 min	Action, Crime,	9	When the mer	84	Christopher N	Christian Bale	Heath Ledger	Aaron Eckhart	Michael Caine	2303232	534,858,444
https://m.med	The Godfather	1974	A	202 min	Crime, Drama	9	The early life a	90	Francis Ford C	Al Pacino	Robert De Nir	Robert Duvall	Diane Keaton	1129952	57,300,000
https://m.med	12 Angry Men	1957	U	96 min	Crime, Drama	9	A jury holdout	96	Sidney Lumet	Henry Fonda	Lee J. Cobb	Martin Balsam	John Fiedler	689845	4,360,000
https://m.med	The Lord of th	2003	U	201 min	Action, Adven	8.9	Gandalf and A	94	Peter Jackson	Elijah Wood	Viggo Mortens	Ian McKellen	Orlando Bloon	1642758	377,845,905
https://m.med	Pulp Fiction	1994	A	154 min	Crime, Drama	8.9	The lives of tw	94	Quentin Taran	John Travolta	Uma Thurman	Samuel L. Jac	Bruce Willis	1826188	107,928,762



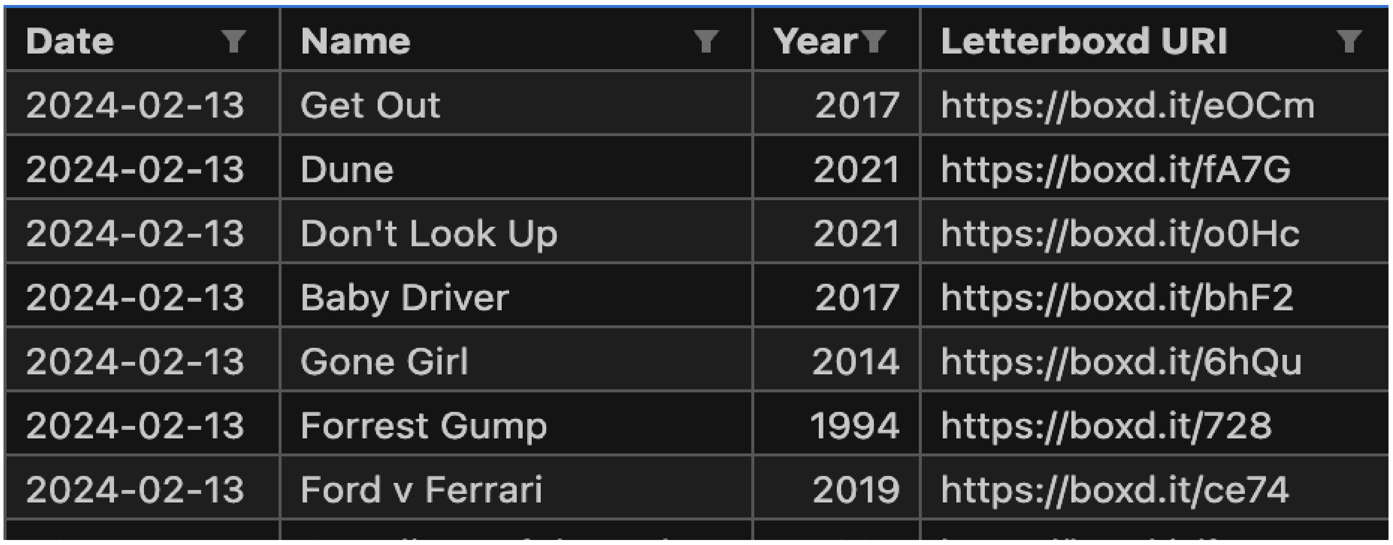
Series_Title	Released_Yea	Prod Compan	Prod Compan	Award1	Award2
The Shawshank Redemption	1994	Castle Rock E	Columbia Pict	Academy Awa	Academy Award Nomination for Best A
The Godfather	1972	Paramount Pic	Alfran Product	Academy Awa	Academy Award for Best Actor
The Dark Knight	2008	Warner Bros. I	Legendary Pic	Academy Awa	BAFTA Award for Best Sound
Pulp Fiction	1994	Miramax Films	A Band Apart	Palme d'Or at	Academy Award for Best Original Scre
Schindler's List	1993	Amblin Enterta	Universal Pict	Academy Awa	Academy Award for Best Director
Fight Club	1999	Fox 2000 Pict	Regency Ente	Online Film Cr	Empire Award for Best Film
The Lord of the Rings: The Return of the King	2003	New Line Cine	WingNut Films	Academy Awa	Academy Award for Best Director



Date	Name	Year	Letterboxd U	Rating	Rewatc	Review	Tags	Watched Date
2025-06-18	Your Name.	2016	https://boxd.it	4		Long term memory loss.		2025-06-18
2024-02-14	Wolf Warrior 2	2017	https://boxd.it	5		What defines a good film? Is it the film		2024-02-14
2025-05-05	Thunderbolts*	2025	https://boxd.it	3.5		Need to start loosing all football matc		2025-05-05
2025-10-21	The Substance	2024	https://boxd.it	3.5		I usually think that every sequel is mu		2025-10-20
2025-01-18	The Call of the Wild	2020	https://boxd.it	5		I WANT A BUCK		2025-01-17
2024-02-20	Taste of Cherry	1997	https://boxd.it	3.5		It's pretty astonishing as to how one		2024-02-19
2024-09-11	Se7en	1995	https://boxd.it	4		How to cope that your counting is rig		2024-09-11



Date	Name	Year	Letterboxd URI	Rating
2025-01-15	Game Changer	2025	https://boxd.it/uzQg	0.5
2025-05-04	The Greatest of All Time	2024	https://boxd.it/GlkW	0.5
2024-10-27	Cocaine Bear	2023	https://boxd.it/ugw2	1
2024-10-27	Velayudham	2011	https://boxd.it/6sk	1
2024-10-27	2	2018	https://boxd.it/cXRm	1
2024-10-27	Bairavaa	2017	https://boxd.it/evSA	1
2025-06-16	#Single	2025	https://boxd.it/SRge	1



Date	Name	Year	Letterboxd URI
2024-02-13	Get Out	2017	https://boxd.it/eOCm
2024-02-13	Dune	2021	https://boxd.it/fA7G
2024-02-13	Don't Look Up	2021	https://boxd.it/o0Hc
2024-02-13	Baby Driver	2017	https://boxd.it/bhF2
2024-02-13	Gone Girl	2014	https://boxd.it/6hQu
2024-02-13	Forrest Gump	1994	https://boxd.it/728
2024-02-13	Ford v Ferrari	2019	https://boxd.it/ce74

Fig-03: Above attached are the heads of all the datasets I have used for my graph mapping.

2. RML Mappings

RML mappings (movie_mapping.rml.ttl) uplifted CSV data into RDF triples.

Each mapping includes:

- Logical Source (rml:logicalSource) pointing to CSV files
- Subject Maps generating unique IRIs via templates for movies, actors, reviews
- Predicate-Object Maps linking CSV fields to ontology predicates
- Datatype declarations using rr:datatype
- Inverse relations handled via separate mappings or SPARQL post-processing

Mapping Summary

Ontology Entities / Classes	**CSV Source**	**Key Columns Used**	**Mapping Strategy & Notes**
moviekg:Movie	**imdb_top_1000.csv**	Series_Title, Released_Year, Overview, IMDB_Rating, Meta_score, No_of_Votes, Gross, Runtime, Genre, Certificate	ex:TM_Movies_IMDB – Creates **Movie** instances with subject IRI:movie/{Series_Title}_{Released_Year}. Maps core datatype properties (title, year, overview, ratings, runtime, revenue). Links to **Genre** (hasGenre) and **Certificate** (hasCertificate) via IRI templates.
moviekg:Genre	**imdb_top_1000.csv**	Genre	ex:TM_Genre_IMDB – Creates **Genre** instances:genre/{Genre} with genreLabel as string.
moviekg:Certificate	**imdb_top_1000.csv**	Certificate	ex:TM_Certificate_IMDB – Creates **Certificate** instances:certificate/{Certificate} with certificateCode.
moviekg:Director	**imdb_top_1000.csv**	Director, Series_Title, Released_Year	ex:TM_Director_IMDB – Creates **Director** persons:person/{Director} with personName. Links Director → Movie using directed and movie IRI {Series_Title}_{Released_Year}.
moviekg:Actor	**imdb_top_1000.csv**	Star1, Star2, Star3, Star4, Series_Title, Released_Year	ex:TM_Actor1_IMDB…ex:TM_Actor4_IMDB – Four TriplesMaps, each creates **Actor** persons:person/{StarX} with personName. Links Actor → Movie using actedIn and same movie IRI template.
moviekg:Production Company	**movie_extra_info.csv**	ProdCompany1, ProdCompany2	ex:TM_ProdCompany1_Extra, ex:TM_ProdCompany2_Extra – Creates **ProductionCompany** instances:company/{ProdCompany1} / {ProdCompany2} with companyName.
moviekg:Movie → moviekg:Production Company	**movie_extra_info.csv**	Series_Title, Released_Year, ProdCompany1, ProdCompany2	ex:TM_Movie_ProdCompany_Link – Reuses movie IRI {Series_Title}_{Released_Year} and links to companies via producedBy (one or two companies per movie).
moviekg:Award	**movie_extra_info.csv**	Award1, Award2	ex:TM_Award1_Extra, ex:TM_Award2_Extra – Creates **Award**

			instances:award/{Award1} / {Award2} with awardLabel.
moviekg:Movie → moviekg:Award	**movie_extra_info.csv**	Series_Title, Released_Year, Award1, Award2	ex:TM_Movie_Award_Link – Links movies to awards using hasAward and the same movie IRI template.
moviekg:Review	**ratings.csv**	Name, Year, Rating, Date, Letterboxd URI	ex:TM_Rating_Reviews – Creates **numeric rating reviews** as Review instances:review/ratings/{Name}_{Year}_{Date}. Links to **Movie** via reviewOf (movie/{Name}_{Year}), records ratingValue, reviewDate, and letterboxdUri.
moviekg:Review	**reviews.csv**	Name, Year, Rating, Review, Watched Date, Date, Tags, Letterboxd URI	ex:TM_TextReviews – Creates **text reviews** as Review instances:review/text/{Name}_{Year}_{Date}. Links to **Movie** via reviewOf. Maps ratingValue, reviewText, watchedDate, reviewDate, tagsText, and letterboxdUri.
moviekg:Review (planned watch)	**watchlist.csv**	Name, Year, Date, Letterboxd URI	ex:TM_Watchlist – Treats watchlist entries as **Review-like events**:review/watchlist/{Name}_{Year}_{Date}. Links to **Movie** via reviewOf, stores reviewDate and letterboxdUri.
moviekg:Review (watched event)	**watched.csv**	Name, Year, Date, Letterboxd URI	ex:TM_Watched – Creates **watched events** as Review instances:review/watched/{Name}_{Year}_{Date}. Links to **Movie** via reviewOf, stores watchedDate and letterboxdUri.
moviekg:Movie with external link	**ratings.csv**	Name, Year, Letterboxd URI	ex:TM_LetterboxdLinks – Reuses movie IRI movie/{Name}_{Year} and attaches external link via owl:sameAs to Letterboxd URI (IRI term). This is your **LOD/Letterboxd interlink**.

Table-01: RML Mapping specifications

The RML mapper (jar) was executed to produce populated RDF triples for ingestion into GraphDB.

```
-jar rmlmapper-8.0.0-r378-all.jar -m movie_mapping.rml.ttl -o output.ttl
```

OWL File & CHOWL-K Diagram

CHOWL-K is a tool that automatically generates graphical diagrams of OWL ontologies.

A CHOWL-K ontology diagram was generated from scratch, representing:

- Classes
- Subclasses
- Object properties
- Datatype properties
- Cardinalities
- Special properties (symmetric/transitive/inverse)

The ontology uses many OWL property characteristics to capture rich relationships. The property similarTo is declared as an owl:SymmetricProperty, meaning that if Movie A is similar to Movie B, then Movie B is automatically inferred to be similar to Movie A. Additionally, properties such as actedIn / hasActor and directed / hasDirector are linked using owl:inverseOf, ensuring bidirectional navigation between entities and supporting more expressive SPARQL queries.

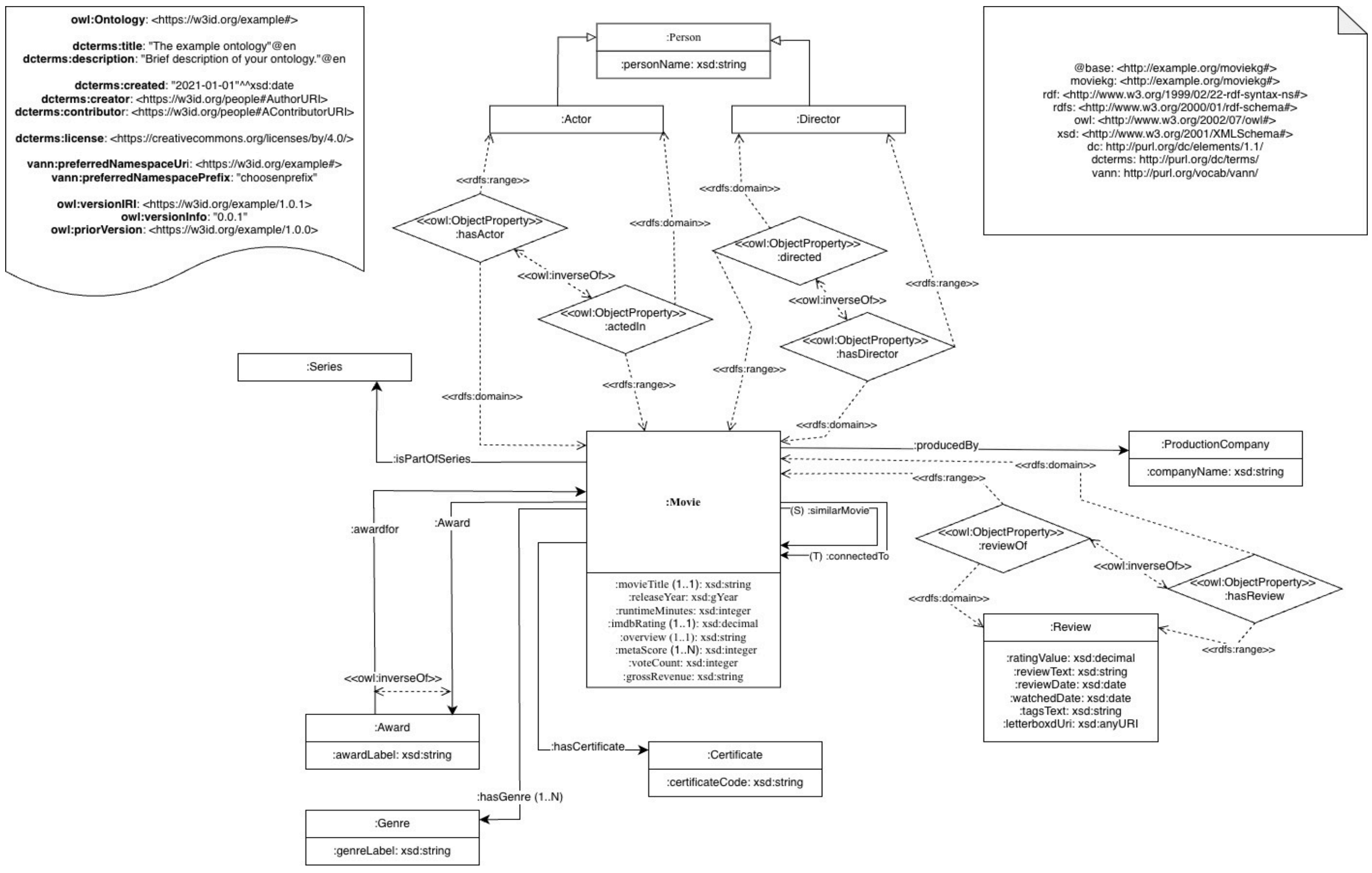

owl:Ontology: <https://w3id.org/example#>
dcterms:title: "The example ontology"@en
dcterms:description: "Brief description of your ontology."@en
dcterms:created: "2021-01-01"^^xsd:date
dcterms:creator: <https://w3id.org/people#AuthorURI>
dcterms:contributor: <https://w3id.org/people#AContributorURI>
dcterms:license: <https://creativecommons.org/licenses/by/4.0/>
vann:preferredNamespaceUri: <https://w3id.org/example#>
vann:preferredNamespacePrefix: "choosenprefix"
owl:versionIRI: <https://w3id.org/example/1.0.1>
owl:versionInfo: "0.0.1"
owl:priorVersion: <https://w3id.org/example/1.0.0>
:Person
:personName: xsd:string
:Actor
:Director
@base: <http://example.org/moviekg#>
moviekg: <http://example.org/moviekg#>
rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
rdfs: <http://www.w3.org/2000/01/rdf-schema#>
owl: <http://www.w3.org/2002/07/owl#>
xsd: <http://www.w3.org/2001/XMLSchema#>
dc: http://purl.org/dc/elements/1.1/
dcterms: http://purl.org/dc/terms/
vann: http://purl.org/vocab/vann/
<<rdfs:range>>
<<owl:ObjectProperty>> :hasActor
<<rdfs:domain>>
<<owl:inverseOf>>
<<owl:ObjectProperty>> :actedIn
<<rdfs:domain>>
<<owl:ObjectProperty>> :directed
<<owl:inverseOf>>
<<rdfs:range>>
<<owl:ObjectProperty>> :hasDirector
:Series
<<rdfs:domain>>
<<rdfs:range>>
<<rdfs:range>>
<<rdfs:domain>>
:isPartOfSeries
:producedBy
:ProductionCompany
:companyName: xsd:string
<<rdfs:domain>>
<<rdfs:range>>
:Movie
(S) :similarMovie
(T) :connectedTo
:movieTitle (1..1): xsd:string
:releaseYear: xsd:gYear
:runtimeMinutes: xsd:integer
:imdbRating (1..1): xsd:decimal
:overview (1..1): xsd:string
:metaScore (1..N): xsd:integer
:voteCount: xsd:integer
:grossRevenue: xsd:string
:awardfor
:Award
<<owl:ObjectProperty>> :reviewOf
<<owl:inverseOf>>
<<owl:ObjectProperty>> :hasReview
<<rdfs:domain>>
:Review
:ratingValue: xsd:decimal
:reviewText: xsd:string
:reviewDate: xsd:date
:watchedDate: xsd:date
:tagsText: xsd:string
:letterboxdUri: xsd:anyURI
<<rdfs:range>>
<<owl:inverseOf>>
:Award
:awardLabel: xsd:string
:hasCertificate
:Certificate
:certificateCode: xsd:string
:hasGenre (1..N)
:Genre
:genreLabel: xsd:string

Fig-04: CHOWLK diagram made using draw.io

The initial CHOWL-K diagram was exported and then manually adjusted to improve readability and better align the visual structure with the ontology's data model

Knowledge Graph Construction

Repository Setup

A new GraphDB repository was created.

Steps:

1. Import Ontology (movies.ttl)
2. Import mapped triples generated by RML
3. Import additional triples (reviews, ratings)
4. Deduplicate via SPARQL normalization
5. Infer actor/director subclasses from Person using rules

The repository now stores:

- Classes: 10
- Object properties: 14+
- Datatype properties: 20+
- External links: IMDb/Letterboxd URIs

SHACL Validation

SHACL checks if your instance data (the triples from uplift) follow your ontology's expectations.

A SHACL file was created to validate:

- Movie must have exactly 1 title
- Movie must have at least 1 genre
- Review must have ratingValue
- Person must have personName
- Certificate must have certificateCode

SHACL shapes use:

- sh:targetClass
- sh:datatype
- sh:minCount
- sh:maxCount
- sh:nodeKind
- Regex patterns for revenue fields if needed

Validation produced a report similar to the sample, highlighting:

- Missing values in ratings or review text
- Genre fields needing normalization
- Incorrect datatypes in revenue column

SPARQL Queries

Query 1 — Basic retrieval (entry-level)

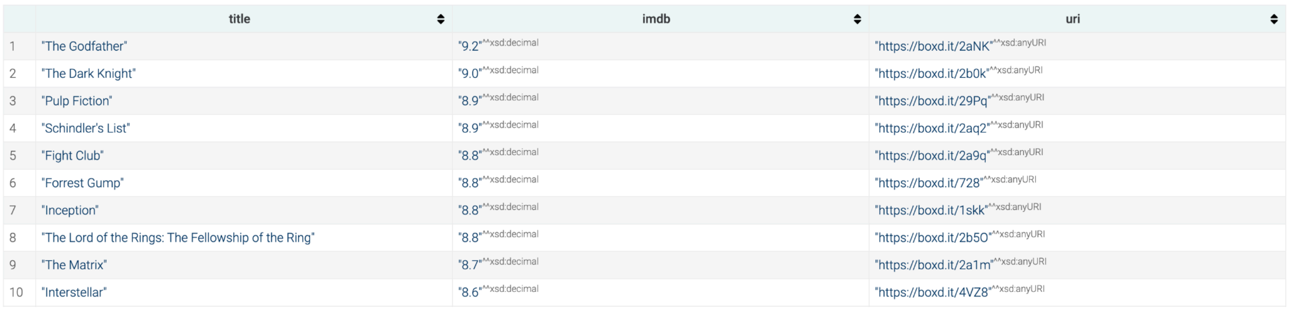
Retrieves movies with their IMDb rating and Letterboxd URI.

```
PREFIX moviekg: <http://example.org/moviekg#>

SELECT DISTINCT ?title ?imdb ?uri WHERE {
  ?movie a moviekg:Movie ;
         moviekg:movieTitle ?title ;
         moviekg:imdbRating ?imdb ;
         moviekg:hasReview ?review .

  ?review moviekg:letterboxdUri ?uri .
}
ORDER BY DESC(?imdb)
```

Output:



	title	imdb	uri
1	"The Godfather"	"9.2"^^xsd:decimal	"https://boxd.it/2aNK"^^xsd:anyURI
2	"The Dark Knight"	"9.0"^^xsd:decimal	"https://boxd.it/2b0k"^^xsd:anyURI
3	"Pulp Fiction"	"8.9"^^xsd:decimal	"https://boxd.it/29Pq"^^xsd:anyURI
4	"Schindler's List"	"8.9"^^xsd:decimal	"https://boxd.it/2aq2"^^xsd:anyURI
5	"Fight Club"	"8.8"^^xsd:decimal	"https://boxd.it/2a9q"^^xsd:anyURI
6	"Forrest Gump"	"8.8"^^xsd:decimal	"https://boxd.it/728"^^xsd:anyURI
7	"Inception"	"8.8"^^xsd:decimal	"https://boxd.it/1skk"^^xsd:anyURI
8	"The Lord of the Rings: The Fellowship of the Ring"	"8.8"^^xsd:decimal	"https://boxd.it/2b5O"^^xsd:anyURI
9	"The Matrix"	"8.7"^^xsd:decimal	"https://boxd.it/2a1m"^^xsd:anyURI
10	"Interstellar"	"8.6"^^xsd:decimal	"https://boxd.it/4VZ8"^^xsd:anyURI

Query 2 — Data cleaning + BIND + casting

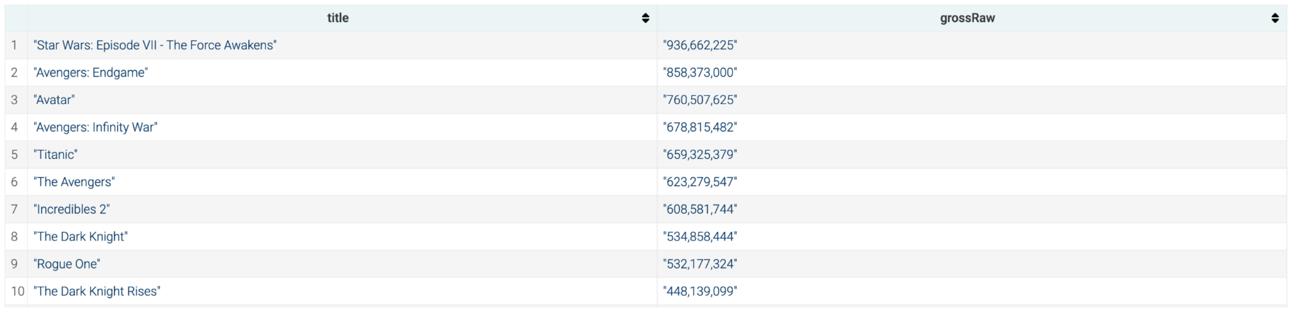
Processes gross revenue by removing commas and converting to integer.

```
PREFIX moviekg: <http://example.org/moviekg#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?title ?grossRaw
WHERE {
  ?m a moviekg:Movie ;
     moviekg:movieTitle ?title ;
     moviekg:grossRevenue ?grossRaw .

  BIND(REPLACE(?grossRaw, ",", "") AS ?digits)
  BIND(xsd:integer(?digits)        AS ?grossInt)
}
ORDER BY DESC(?grossInt)
LIMIT 20
```

Output:



	title	grossRaw
1	"Star Wars: Episode VII - The Force Awakens"	"936,662,225"
2	"Avengers: Endgame"	"858,373,000"
3	"Avatar"	"760,507,625"
4	"Avengers: Infinity War"	"678,815,482"
5	"Titanic"	"659,325,379"
6	"The Avengers"	"623,279,547"
7	"Incredibles 2"	"608,581,744"
8	"The Dark Knight"	"534,858,444"
9	"Rogue One"	"532,177,324"
10	"The Dark Knight Rises"	"448,139,099"

Query 3 — OPTIONAL and FILTER logic

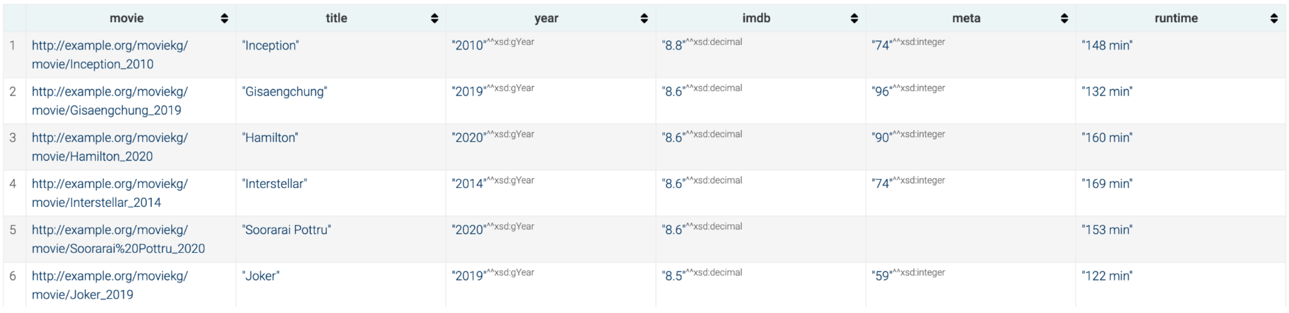
Retrieves movies from 2010 onward with optional metadata.

```
PREFIX moviekg: <http://example.org/moviekg#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?movie ?title ?year ?imdb ?meta ?runtime
WHERE {
  ?movie a moviekg:Movie ;
         moviekg:movieTitle ?title ;
         moviekg:releaseYear ?year .

  OPTIONAL { ?movie moviekg:imdbRating     ?imdb }
  OPTIONAL { ?movie moviekg:metaScore      ?meta }
  OPTIONAL { ?movie moviekg:runtimeMinutes ?runtime }

  FILTER (?year >= "2010"^^xsd:gYear)
}
ORDER BY DESC(?imdb) ?title
LIMIT 20
```
Output:



	movie	title	year	imdb	meta	runtime
1	http://example.org/moviekg/movie/Inception_2010	"Inception"	"2010"^^xsd:gYear	"8.8"^^xsd:decimal	"74"^^xsd:integer	"148 min"
2	http://example.org/moviekg/movie/Gisaengchung_2019	"Gisaengchung"	"2019"^^xsd:gYear	"8.6"^^xsd:decimal	"96"^^xsd:integer	"132 min"
3	http://example.org/moviekg/movie/Hamilton_2020	"Hamilton"	"2020"^^xsd:gYear	"8.6"^^xsd:decimal	"90"^^xsd:integer	"160 min"
4	http://example.org/moviekg/movie/Interstellar_2014	"Interstellar"	"2014"^^xsd:gYear	"8.6"^^xsd:decimal	"74"^^xsd:integer	"169 min"
5	http://example.org/moviekg/movie/Soorarai%20Pottru_2020	"Soorarai Pottru"	"2020"^^xsd:gYear	"8.6"^^xsd:decimal		"153 min"
6	http://example.org/moviekg/movie/Joker_2019	"Joker"	"2019"^^xsd:gYear	"8.5"^^xsd:decimal	"59"^^xsd:integer	"122 min"

Query 4 — Content-based movie recommendations (shared genre)

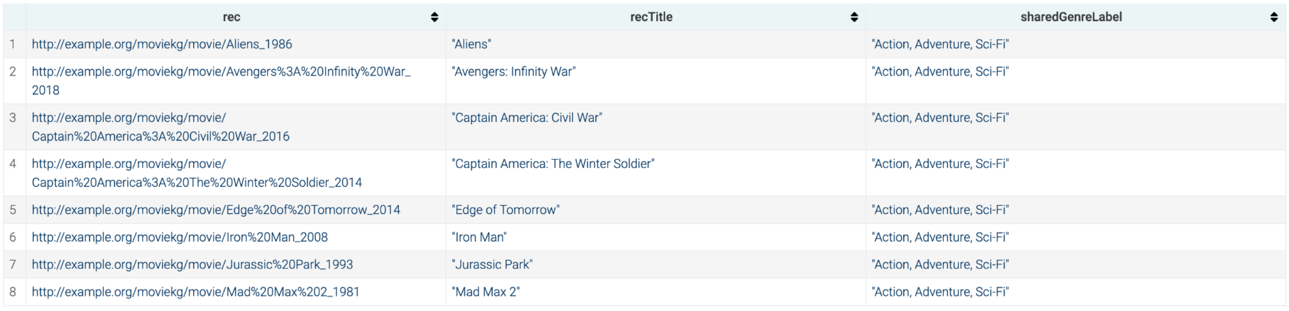
Finds movies that share a genre with “Inception”.

```
PREFIX moviekg: <http://example.org/moviekg#>

SELECT DISTINCT ?rec ?recTitle ?sharedGenreLabel
WHERE {
  ?seed a moviekg:Movie ;
        moviekg:movieTitle "Inception" ;
        moviekg:hasGenre ?g .
  ?g moviekg:genreLabel ?sharedGenreLabel .

  ?rec a moviekg:Movie ;
       moviekg:hasGenre ?g ;
       moviekg:movieTitle ?recTitle .

  FILTER(?rec != ?seed)
}
ORDER BY ?sharedGenreLabel ?recTitle
LIMIT 20
```
Output:

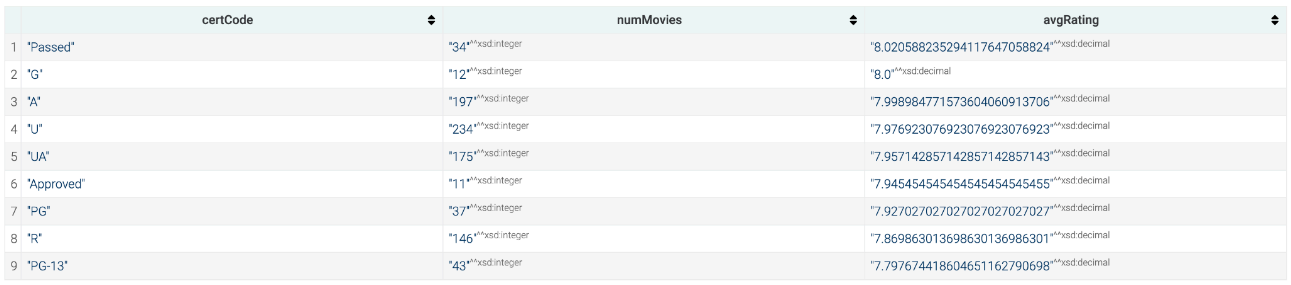


	rec	recTitle	sharedGenreLabel
1	http://example.org/moviekg/movie/Aliens_1986	"Aliens"	"Action, Adventure, Sci-Fi"
2	http://example.org/moviekg/movie/Avengers%3A%20Infinity%20War_2018	"Avengers: Infinity War"	"Action, Adventure, Sci-Fi"
3	http://example.org/moviekg/movie/Captain%20America%3A%20Civil%20War_2016	"Captain America: Civil War"	"Action, Adventure, Sci-Fi"
4	http://example.org/moviekg/movie/Captain%20America%3A%20The%20Winter%20Soldier_2014	"Captain America: The Winter Soldier"	"Action, Adventure, Sci-Fi"
5	http://example.org/moviekg/movie/Edge%20of%20Tomorrow_2014	"Edge of Tomorrow"	"Action, Adventure, Sci-Fi"
6	http://example.org/moviekg/movie/Iron%20Man_2008	"Iron Man"	"Action, Adventure, Sci-Fi"
7	http://example.org/moviekg/movie/Jurassic%20Park_1993	"Jurassic Park"	"Action, Adventure, Sci-Fi"
8	http://example.org/moviekg/movie/Mad%20Max%202_1981	"Mad Max 2"	"Action, Adventure, Sci-Fi"

Query 5 — Aggregation + GROUP BY + HAVING

Computes average IMDb rating per certificate (e.g., PG, R).

```
SELECT ?reviewer (COUNT(?review) AS ?count) WHERE {
  ?review a moviekg:Review ;
          moviekg:letterboxdUri ?reviewer .
}
GROUP BY ?reviewer
ORDER BY DESC(?count)
```
Output:



	certCode	numMovies	avgRating
1	"Passed"	"34"^^xsd:integer	"8.020588235294117647058824"^^xsd:decimal
2	"G"	"12"^^xsd:integer	"8.0"^^xsd:decimal
3	"A"	"197"^^xsd:integer	"7.998984771573604060913706"^^xsd:decimal
4	"U"	"234"^^xsd:integer	"7.976923076923076923076923"^^xsd:decimal
5	"UA"	"175"^^xsd:integer	"7.957142857142857142857143"^^xsd:decimal
6	"Approved"	"11"^^xsd:integer	"7.945454545454545454545455"^^xsd:decimal
7	"PG"	"37"^^xsd:integer	"7.927027027027027027027027"^^xsd:decimal
8	"R"	"146"^^xsd:integer	"7.869863013698630136986301"^^xsd:decimal
9	"PG-13"	"43"^^xsd:integer	"7.797674418604651162790698"^^xsd:decimal

Query 6 — Subqueries + genre benchmarking

Compares each genre's average rating to the global average.

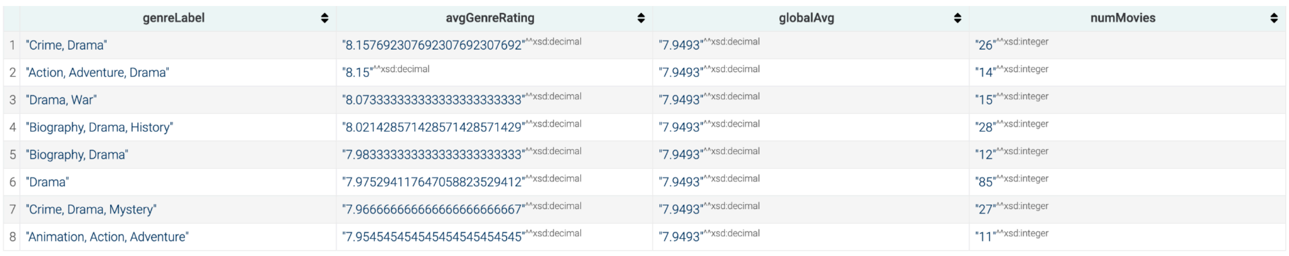
```
PREFIX moviekg: <http://example.org/moviekg#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?genreLabel ?avgGenreRating ?globalAvg ?numMovies
WHERE {
  { SELECT (AVG(xsd:decimal(?r)) AS ?globalAvg)
    WHERE {
      ?m a moviekg:Movie ;
         moviekg:imdbRating ?r .
    }
  }

  { SELECT ?genreLabel
           (AVG(xsd:decimal(?rating)) AS ?avgGenreRating)
           (COUNT(?m)                 AS ?numMovies)
    WHERE {
      ?m a moviekg:Movie ;
         moviekg:hasGenre   ?g ;
         moviekg:imdbRating ?rating .
      ?g moviekg:genreLabel ?genreLabel .
    }
    GROUP BY ?genreLabel
    HAVING (COUNT(?m) >= 10)
  }

  FILTER(?avgGenreRating > ?globalAvg)
}
ORDER BY DESC(?avgGenreRating)
```

Output:



	genreLabel	avgGenreRating	globalAvg	numMovies
1	"Crime, Drama"	"8.157692307692307692307692"^^xsd:decimal	"7.9493"^^xsd:decimal	"26"^^xsd:integer
2	"Action, Adventure, Drama"	"8.15"^^xsd:decimal	"7.9493"^^xsd:decimal	"14"^^xsd:integer
3	"Drama, War"	"8.073333333333333333333333"^^xsd:decimal	"7.9493"^^xsd:decimal	"15"^^xsd:integer
4	"Biography, Drama, History"	"8.021428571428571428571429"^^xsd:decimal	"7.9493"^^xsd:decimal	"28"^^xsd:integer
5	"Biography, Drama"	"7.983333333333333333333333"^^xsd:decimal	"7.9493"^^xsd:decimal	"12"^^xsd:integer
6	"Drama"	"7.975294117647058823529412"^^xsd:decimal	"7.9493"^^xsd:decimal	"85"^^xsd:integer
7	"Crime, Drama, Mystery"	"7.966666666666666666666667"^^xsd:decimal	"7.9493"^^xsd:decimal	"27"^^xsd:integer
8	"Animation, Action, Adventure"	"7.954545454545454545454545"^^xsd:decimal	"7.9493"^^xsd:decimal	"11"^^xsd:integer

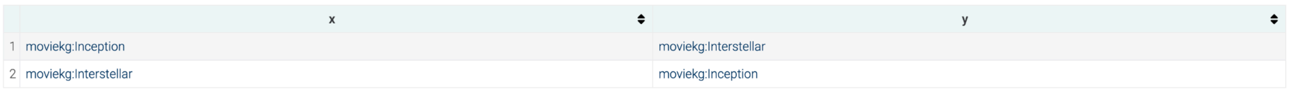
Query 7 — Reasoning over symmetric/transitive/inverseOf properties

This showcases the reasoning aspects of your ontology, required by the spec.

```
PREFIX moviekg: <http://example.org/moviekg#>

SELECT ?x ?y
WHERE {
  ?x moviekg:similarTo ?y .
}
```

Output:



	x	y
1	moviekg:Inception	moviekg:Interstellar
2	moviekg:Interstellar	moviekg:Inception

Reflections

Achievements

This research effectively illustrates how diverse film-related metadata may be transformed end-to-end into a structured, semantically rich knowledge graph. The MovieKG facilitates coherent reasoning across cinematic entities, relationship-based recommendations, and expressive SPARQL querying by combining Letterboxd-style and IMDb-style information under a single ontology.

The pipeline leverages RML for declarative data uplift, SHACL for constraint validation, CHOWL-K for ontology visualisation, and GraphDB for storage and reasoning. Collectively, these components form a complete semantic-web workflow that mirrors real-world applications in media intelligence, content discovery, and entertainment analytics.

Challenges

Several challenges emerged during implementation:

- Cleaning and normalising messy cast lists, multi-valued columns, and inconsistent genre formats.
- Ensuring datatype consistency across gYear, decimal, date, and string fields.
- Aligning the Letterboxd export structure with the ontology in a way that preserved semantics while avoiding one-to-one CSV mapping.
- Debugging CHOWL-K export issues, namespace conflicts, and XML validation glitches.
- Managing the large volume of Actors and Directors generated through CSV splitting, ensuring instance-level correctness and avoiding IRI collisions.

Despite these challenges, each issue contributed to a deeper understanding of ontology modelling, data integration, and linked-data engineering.

Future Work

There are several directions in which the MovieKG could be expanded:

- Integrating DBpedia or Wikidata film entities to enrich metadata and improve interoperability.
- Incorporating Schema.org metadata to describe trailers, posters, and additional media assets.
- Developing a graph-based recommendation engine using similarity metrics or embeddings derived from the KG.
- Modelling awards (e.g., Oscars, Golden Globes) with greater granularity by introducing ceremony entities and temporal dimensions.
- Building an actor/crew collaboration network to explore influence, co-occurrence patterns, and centrality across films.

These enhancements would further strengthen the KG’s analytical capabilities and enable more sophisticated film-domain applications.

Conclusion

The Movie Knowledge Graph provides a comprehensive and semantically rigorous model for representing, analysing, and querying film-related information. Through ontology engineering, RML-based data uplift, systematic data cleaning, and SHACL-driven validation, this project demonstrates the complete Knowledge & Data Engineering pipeline in practice.

The outcome is a reusable, extensible knowledge graph that supports advanced querying, recommendation tasks, semantic interoperability, and future research in film analytics and linked-data integration.

References

[1] S. Chávez-Feria, R. García-Castro, and M. Poveda-Villalón, "Chowlk: From UML-Based Ontology Conceptualizations to OWL," in The Semantic Web: 19th International Conference (ESWC 2022), P. Groth et al., Eds., Cham: Springer, 2022, pp. 338–352. doi: 10.1007/978-3-031-06981-9_20.

[2] JGraph Ltd., diagrams.net, 2025. [Online]. Available: https://app.diagrams.net/

[3] S. Lohmann, S. Negru, F. Haag, and T. Ertl, "WebVOWL: Web-based Visualization of Ontologies," in EKAW 2014 Satellite Events, LNAI 8982, Springer, 2015, pp. 154–158. [Online].

Available: https://service.tib.eu/webvowl/

[4] RML.io, RMLMapper, 2025. [Online]. Available: https://rml.io/

[5] Ontotext, GraphDB: Enterprise Ready Semantic Graph Database, 2025. [Online]. Available: https://graphdb.ontotext.com/

Appendix

movies.ttl

```
BASE <http://example.org/moviekg#>
@prefix moviekg: <http://example.org/moviekg#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl:     <http://www.w3.org/2002/07/owl#> .
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .

moviekg:MovieKGOntology
   a owl:Ontology ;
   rdfs:label "Movie Knowledge Graph Ontology" ;
   rdfs:comment "Ontology for a movie knowledge graph integrating IMDb-style
and Letterboxd-style CSV data." .

### Top-level (8)

moviekg:Movie a owl:Class ;
   rdfs:subClassOf owl:Thing ;
   rdfs:label "Movie" .

moviekg:Person a owl:Class ;
   rdfs:subClassOf owl:Thing ;
   rdfs:label "Person" .

moviekg:Genre a owl:Class ;
   rdfs:subClassOf owl:Thing ;
   rdfs:label "Genre" .

moviekg:Series a owl:Class ;
   rdfs:subClassOf owl:Thing ;
   rdfs:label "Series" .

moviekg:ProductionCompany a owl:Class ;
   rdfs:subClassOf owl:Thing ;
   rdfs:label "Production Company" .

moviekg:Certificate a owl:Class ;
   rdfs:subClassOf owl:Thing ;
   rdfs:label "Certificate" .

moviekg:Review a owl:Class ;
   rdfs:subClassOf owl:Thing ;
   rdfs:label "Review / Rating" .
```

```
moviekg:Award a owl:Class ;
   rdfs:subClassOf owl:Thing ;
   rdfs:label "Award" .

### Subclasses (2) under Person

moviekg:Actor a owl:Class ;
   rdfs:subClassOf moviekg:Person ;
   rdfs:label "Actor" .

moviekg:Director a owl:Class ;
   rdfs:subClassOf moviekg:Person ;
   rdfs:label "Director" .
####Object Properties


moviekg:actedIn a owl:ObjectProperty ;
   rdfs:domain moviekg:Actor ;
   rdfs:range  moviekg:Movie ;
   rdfs:label "acted in" ;
   rdfs:comment "Links an Actor to a Movie they acted in." .

moviekg:hasActor a owl:ObjectProperty ;
   owl:inverseOf moviekg:actedIn ;
   rdfs:domain moviekg:Movie ;
   rdfs:range  moviekg:Actor ;
   rdfs:label "has actor" .

moviekg:directed a owl:ObjectProperty ;
   rdfs:domain moviekg:Director ;
   rdfs:range  moviekg:Movie ;
   rdfs:label "directed" .

moviekg:hasDirector a owl:ObjectProperty ;
   owl:inverseOf moviekg:directed ;
   rdfs:domain moviekg:Movie ;
   rdfs:range  moviekg:Director ;
   rdfs:label "has director" .

moviekg:hasGenre a owl:ObjectProperty ;
   rdfs:domain moviekg:Movie ;
   rdfs:range  moviekg:Genre ;
   rdfs:label "has genre" .

moviekg:isPartOfSeries a owl:ObjectProperty ;
```

```
   rdfs:domain moviekg:Movie ;
   rdfs:range  moviekg:Series ;
   rdfs:label "is part of series" .

moviekg:producedBy a owl:ObjectProperty ;
   rdfs:domain moviekg:Movie ;
   rdfs:range  moviekg:ProductionCompany ;
   rdfs:label "produced by" .

moviekg:hasCertificate a owl:ObjectProperty ;
   rdfs:domain moviekg:Movie ;
   rdfs:range  moviekg:Certificate ;
   rdfs:label "has certificate" .

moviekg:hasReview a owl:ObjectProperty ;
   rdfs:domain moviekg:Movie ;
   rdfs:range  moviekg:Review ;
   rdfs:label "has review" .

moviekg:reviewOf a owl:ObjectProperty ;
   owl:inverseOf moviekg:hasReview ;
   rdfs:domain moviekg:Review ;
   rdfs:range  moviekg:Movie ;
   rdfs:label "review of" .

moviekg:hasAward a owl:ObjectProperty ;
   rdfs:domain moviekg:Movie ;
   rdfs:range  moviekg:Award ;
   rdfs:label "has award" .

moviekg:awardFor a owl:ObjectProperty ;
   owl:inverseOf moviekg:hasAward ;
   rdfs:domain moviekg:Award ;
   rdfs:range  moviekg:Movie ;
   rdfs:label "award for" .

# Example symmetric / transitive

moviekg:similarMovie a owl:ObjectProperty , owl:SymmetricProperty ;
   rdfs:domain moviekg:Movie ;
   rdfs:range  moviekg:Movie ;
   rdfs:label "similar movie" .

moviekg:connectedTo a owl:ObjectProperty , owl:TransitiveProperty ;
   rdfs:domain moviekg:Movie ;
   rdfs:range  moviekg:Movie ;
   rdfs:label "connected to" .


# Datatype Properties
### Movie

moviekg:movieTitle a owl:DatatypeProperty ;
   rdfs:domain moviekg:Movie ;
   rdfs:range  xsd:string ;
   rdfs:label "movie title" .

moviekg:releaseYear a owl:DatatypeProperty ;
   rdfs:domain moviekg:Movie ;
   rdfs:range  xsd:gYear ;
   rdfs:label "release year" .

moviekg:runtimeMinutes a owl:DatatypeProperty ;
   rdfs:domain moviekg:Movie ;
   rdfs:range  xsd:integer ;
   rdfs:label "runtime in minutes" .

moviekg:imdbRating a owl:DatatypeProperty ;
   rdfs:domain moviekg:Movie ;
   rdfs:range  xsd:decimal ;
   rdfs:label "IMDb rating" .

moviekg:overview a owl:DatatypeProperty ;
   rdfs:domain moviekg:Movie ;
   rdfs:range  xsd:string ;
   rdfs:label "plot overview" .

moviekg:metaScore a owl:DatatypeProperty ;
   rdfs:domain moviekg:Movie ;
   rdfs:range  xsd:integer ;
   rdfs:label "Metascore" .

moviekg:voteCount a owl:DatatypeProperty ;
   rdfs:domain moviekg:Movie ;
   rdfs:range  xsd:integer ;
   rdfs:label "number of votes" .

moviekg:grossRevenue a owl:DatatypeProperty ;
   rdfs:domain moviekg:Movie ;
   rdfs:range  xsd:string ;
   rdfs:label "gross box office (string)" .
```

```
### Person / subclasses

moviekg:personName a owl:DatatypeProperty ;
   rdfs:domain moviekg:Person ;
   rdfs:range  xsd:string ;
   rdfs:label "person name" .

### Genre / Certificate / Company / Award

moviekg:genreLabel a owl:DatatypeProperty ;
   rdfs:domain moviekg:Genre ;
   rdfs:range  xsd:string ;
   rdfs:label "genre label" .

moviekg:certificateCode a owl:DatatypeProperty ;
   rdfs:domain moviekg:Certificate ;
   rdfs:range  xsd:string ;
   rdfs:label "certificate code" .

moviekg:companyName a owl:DatatypeProperty ;
   rdfs:domain moviekg:ProductionCompany ;
   rdfs:range  xsd:string ;
   rdfs:label "production company name" .

moviekg:awardLabel a owl:DatatypeProperty ;
   rdfs:domain moviekg:Award ;
   rdfs:range  xsd:string ;
   rdfs:label "award label" .

### Reviews + Letterboxd URI

moviekg:ratingValue a owl:DatatypeProperty ;
   rdfs:domain moviekg:Review ;
   rdfs:range  xsd:decimal ;
   rdfs:label "rating value" .

moviekg:reviewText a owl:DatatypeProperty ;
   rdfs:domain moviekg:Review ;
   rdfs:range  xsd:string ;
   rdfs:label "review text" .

moviekg:reviewDate a owl:DatatypeProperty ;
   rdfs:domain moviekg:Review ;
   rdfs:range  xsd:date ;
   rdfs:label "review date" .
```

```
moviekg:watchedDate a owl:DatatypeProperty ;
   rdfs:domain moviekg:Review ;
   rdfs:range  xsd:date ;
   rdfs:label "watched date" .

moviekg:tagsText a owl:DatatypeProperty ;
   rdfs:domain moviekg:Review ;
   rdfs:range  xsd:string ;
   rdfs:label "tags text" .

moviekg:letterboxdUri a owl:DatatypeProperty ;
   rdfs:domain moviekg:Review ;
   rdfs:range  xsd:anyURI ;
   rdfs:label "Letterboxd URI" ;
   rdfs:comment "Letterboxd URI from export; also used for owl:sameAs links
at the Movie level via RML mapping." .

# Cardinality Examples

moviekg:Movie rdfs:subClassOf
   [ a owl:Restriction ;
     owl:onProperty moviekg:movieTitle ;
     owl:cardinality "1"^^xsd:nonNegativeInteger
   ] ,
   [ a owl:Restriction ;
     owl:onProperty moviekg:hasGenre ;
     owl:minCardinality "1"^^xsd:nonNegativeInteger
   ] .

moviekg:Review rdfs:subClassOf
   [ a owl:Restriction ;
     owl:onProperty moviekg:ratingValue ;
     owl:minCardinality "0"^^xsd:nonNegativeInteger
   ] .
```

movie_mapping.rml.ttl:

```
# RML Mapping - movies_rml_mapping.ttl
@prefix rr:      <http://www.w3.org/ns/r2rml#> .
@prefix rml:     <http://semweb.mmlab.be/ns/rml#> .
@prefix ql:      <http://semweb.mmlab.be/ns/ql#> .
@prefix moviekg: <http://example.org/moviekg#> .
@prefix ex:      <http://example.org/mapping#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl:     <http://www.w3.org/2002/07/owl#> .
```

```
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .

# 1. Movies from imdb_top_1000.csv

ex:TM_Movies_IMDB a rr:TriplesMap ;
   rml:logicalSource [
       rml:source "imdb_top_1000.csv" ;
       rml:referenceFormulation ql:CSV
   ] ;

   rr:subjectMap [
       rr:template
"http://example.org/moviekg/movie/{Series_Title}_{Released_Year}" ;
       rr:class moviekg:Movie
   ] ;

   # Core attributes
   rr:predicateObjectMap [
       rr:predicate moviekg:movieTitle ;
       rr:objectMap [ rml:reference "Series_Title" ; rr:datatype xsd:string ]
   ] ;
   rr:predicateObjectMap [
       rr:predicate moviekg:releaseYear ;
       rr:objectMap [ rml:reference "Released_Year" ; rr:datatype xsd:gYear ]
   ] ;
   rr:predicateObjectMap [
       rr:predicate moviekg:overview ;
       rr:objectMap [ rml:reference "Overview" ; rr:datatype xsd:string ]
   ] ;
   rr:predicateObjectMap [
       rr:predicate moviekg:imdbRating ;
       rr:objectMap [ rml:reference "IMDB_Rating" ; rr:datatype xsd:decimal ]
   ] ;
   rr:predicateObjectMap [
       rr:predicate moviekg:metaScore ;
       rr:objectMap [ rml:reference "Meta_score" ; rr:datatype xsd:integer ]
   ] ;
   rr:predicateObjectMap [
       rr:predicate moviekg:voteCount ;
       rr:objectMap [ rml:reference "No_of_Votes" ; rr:datatype xsd:integer ]
   ] ;
   rr:predicateObjectMap [
       rr:predicate moviekg:grossRevenue ;
       rr:objectMap [ rml:reference "Gross" ; rr:datatype xsd:string ]
   ] ;
   # Runtime (string; can be cleaned later if needed)
   rr:predicateObjectMap [
       rr:predicate moviekg:runtimeMinutes ;
       rr:objectMap [ rml:reference "Runtime" ; rr:datatype xsd:string ]
   ] ;

   # Genre and certificate
   rr:predicateObjectMap [
       rr:predicate moviekg:hasGenre ;
       rr:objectMap [
           rr:template "http://example.org/moviekg/genre/{Genre}" ;
           rr:termType rr:IRI
       ]
   ] ;
   rr:predicateObjectMap [
       rr:predicate moviekg:hasCertificate ;
       rr:objectMap [
           rr:template "http://example.org/moviekg/certificate/{Certificate}"
;
           rr:termType rr:IRI
       ]
   ] .

# 2. Genre & Certificate instances

ex:TM_Genre_IMDB a rr:TriplesMap ;
   rml:logicalSource [
       rml:source "imdb_top_1000.csv" ;
       rml:referenceFormulation ql:CSV
   ] ;
   rr:subjectMap [
       rr:template "http://example.org/moviekg/genre/{Genre}" ;
       rr:class moviekg:Genre
   ] ;
   rr:predicateObjectMap [
       rr:predicate moviekg:genreLabel ;
       rr:objectMap [ rml:reference "Genre" ; rr:datatype xsd:string ]
   ] .

ex:TM_Certificate_IMDB a rr:TriplesMap ;
   rml:logicalSource [
       rml:source "imdb_top_1000.csv" ;
       rml:referenceFormulation ql:CSV
   ] ;
   rr:subjectMap [
       rr:template "http://example.org/moviekg/certificate/{Certificate}" ;
```

```
        rr:class moviekg:Certificate
    ] ;
    rr:predicateObjectMap [
        rr:predicate moviekg:certificateCode ;
        rr:objectMap [ rml:reference "Certificate" ; rr:datatype xsd:string ]
    ] .

# 3. People: Directors & Actors from imdb_top_1000.csv

# Directors (Director column)

ex:TM_Director_IMDB a rr:TriplesMap ;
    rml:logicalSource [
        rml:source "imdb_top_1000.csv" ;
        rml:referenceFormulation ql:CSV
    ] ;
    rr:subjectMap [
        rr:template "http://example.org/moviekg/person/{Director}" ;
        rr:class moviekg:Director
    ] ;
    rr:predicateObjectMap [
        rr:predicate moviekg:personName ;
        rr:objectMap [ rml:reference "Director" ; rr:datatype xsd:string ]
    ] ;
    rr:predicateObjectMap [
        rr:predicate moviekg:directed ;
        rr:objectMap [
            rr:template
"http://example.org/moviekg/movie/{Series_Title}_{Released_Year}" ;
            rr:termType rr:IRI
        ]
    ] .

# Actors from Star1..Star4

ex:TM_Actor1_IMDB a rr:TriplesMap ;
    rml:logicalSource [
        rml:source "imdb_top_1000.csv" ;
        rml:referenceFormulation ql:CSV
    ] ;
    rr:subjectMap [
        rr:template "http://example.org/moviekg/person/{Star1}" ;
        rr:class moviekg:Actor
    ] ;
    rr:predicateObjectMap [
        rr:predicate moviekg:personName ;
        rr:objectMap [ rml:reference "Star1" ; rr:datatype xsd:string ]
    ] ;
    rr:predicateObjectMap [
        rr:predicate moviekg:actedIn ;
        rr:objectMap [
            rr:template
"http://example.org/moviekg/movie/{Series_Title}_{Released_Year}" ;
            rr:termType rr:IRI
        ]
    ] .

ex:TM_Actor2_IMDB a rr:TriplesMap ;
    rml:logicalSource [
        rml:source "imdb_top_1000.csv" ;
        rml:referenceFormulation ql:CSV
    ] ;
    rr:subjectMap [
        rr:template "http://example.org/moviekg/person/{Star2}" ;
        rr:class moviekg:Actor
    ] ;
    rr:predicateObjectMap [
        rr:predicate moviekg:personName ;
        rr:objectMap [ rml:reference "Star2" ; rr:datatype xsd:string ]
    ] ;
    rr:predicateObjectMap [
        rr:predicate moviekg:actedIn ;
        rr:objectMap [
            rr:template
"http://example.org/moviekg/movie/{Series_Title}_{Released_Year}" ;
            rr:termType rr:IRI
        ]
    ] .

ex:TM_Actor3_IMDB a rr:TriplesMap ;
    rml:logicalSource [
        rml:source "imdb_top_1000.csv" ;
        rml:referenceFormulation ql:CSV
    ] ;
    rr:subjectMap [
        rr:template "http://example.org/moviekg/person/{Star3}" ;
        rr:class moviekg:Actor
    ] ;
    rr:predicateObjectMap [
        rr:predicate moviekg:personName ;
        rr:objectMap [ rml:reference "Star3" ; rr:datatype xsd:string ]
    ] ;
```

```
   rr:predicateObjectMap [
       rr:predicate moviekg:actedIn ;
       rr:objectMap [
           rr:template
"http://example.org/moviekg/movie/{Series_Title}_{Released_Year}" ;
           rr:termType rr:IRI
       ]
   ] .

ex:TM_Actor4_IMDB a rr:TriplesMap ;
   rml:logicalSource [
       rml:source "imdb_top_1000.csv" ;
       rml:referenceFormulation ql:CSV
   ] ;
   rr:subjectMap [
       rr:template "http://example.org/moviekg/person/{Star4}" ;
       rr:class moviekg:Actor
   ] ;
   rr:predicateObjectMap [
       rr:predicate moviekg:personName ;
       rr:objectMap [ rml:reference "Star4" ; rr:datatype xsd:string ]
   ] ;
   rr:predicateObjectMap [
       rr:predicate moviekg:actedIn ;
       rr:objectMap [
           rr:template
"http://example.org/moviekg/movie/{Series_Title}_{Released_Year}" ;
           rr:termType rr:IRI
       ]
   ] .
# 4. Production Companies & Awards from movie_extra_info.csv

# ProductionCompany instances from ProdCompany1

ex:TM_ProdCompany1_Extra a rr:TriplesMap ;
   rml:logicalSource [
       rml:source "movie_extra_info.csv" ;
       rml:referenceFormulation ql:CSV
   ] ;
   rr:subjectMap [
       rr:template "http://example.org/moviekg/company/{ProdCompany1}" ;
       rr:class moviekg:ProductionCompany
   ] ;
   rr:predicateObjectMap [
       rr:predicate moviekg:companyName ;
       rr:objectMap [ rml:reference "ProdCompany1" ; rr:datatype xsd:string ]
   ] .

# ProductionCompany instances from ProdCompany2

ex:TM_ProdCompany2_Extra a rr:TriplesMap ;
   rml:logicalSource [
       rml:source "movie_extra_info.csv" ;
       rml:referenceFormulation ql:CSV
   ] ;
   rr:subjectMap [
       rr:template "http://example.org/moviekg/company/{ProdCompany2}" ;
       rr:class moviekg:ProductionCompany
   ] ;
   rr:predicateObjectMap [
       rr:predicate moviekg:companyName ;
       rr:objectMap [ rml:reference "ProdCompany2" ; rr:datatype xsd:string ]
   ] .

# Link Movie -> ProductionCompany using extra CSV

ex:TM_Movie_ProdCompany_Link a rr:TriplesMap ;
   rml:logicalSource [
       rml:source "movie_extra_info.csv" ;
       rml:referenceFormulation ql:CSV
   ] ;
   rr:subjectMap [
       rr:template
"http://example.org/moviekg/movie/{Series_Title}_{Released_Year}" ;
       rr:class moviekg:Movie
   ] ;
   rr:predicateObjectMap [
       rr:predicate moviekg:producedBy ;
       rr:objectMap [
           rr:template "http://example.org/moviekg/company/{ProdCompany1}" ;
           rr:termType rr:IRI
       ]
   ] ;
   rr:predicateObjectMap [
       rr:predicate moviekg:producedBy ;
       rr:objectMap [
           rr:template "http://example.org/moviekg/company/{ProdCompany2}" ;
           rr:termType rr:IRI
       ]
   ] .

# Award instances
```

```
ex:TM_Award1_Extra a rr:TriplesMap ;
   rml:logicalSource [
       rml:source "movie_extra_info.csv" ;
       rml:referenceFormulation ql:CSV
   ] ;
   rr:subjectMap [
       rr:template "http://example.org/moviekg/award/{Award1}" ;
       rr:class moviekg:Award
   ] ;
   rr:predicateObjectMap [
       rr:predicate moviekg:awardLabel ;
       rr:objectMap [ rml:reference "Award1" ; rr:datatype xsd:string ]
   ] .

ex:TM_Award2_Extra a rr:TriplesMap ;
   rml:logicalSource [
       rml:source "movie_extra_info.csv" ;
       rml:referenceFormulation ql:CSV
   ] ;
   rr:subjectMap [
       rr:template "http://example.org/moviekg/award/{Award2}" ;
       rr:class moviekg:Award
   ] ;
   rr:predicateObjectMap [
       rr:predicate moviekg:awardLabel ;
       rr:objectMap [ rml:reference "Award2" ; rr:datatype xsd:string ]
   ] .

# Link Movie -> Award

ex:TM_Movie_Award_Link a rr:TriplesMap ;
   rml:logicalSource [
       rml:source "movie_extra_info.csv" ;
       rml:referenceFormulation ql:CSV
   ] ;
   rr:subjectMap [
       rr:template
"http://example.org/moviekg/movie/{Series_Title}_{Released_Year}" ;
       rr:class moviekg:Movie
   ] ;
   rr:predicateObjectMap [
       rr:predicate moviekg:hasAward ;
       rr:objectMap [
           rr:template "http://example.org/moviekg/award/{Award1}" ;
           rr:termType rr:IRI
```

```
        ]
    ] ;
    rr:predicateObjectMap [
        rr:predicate moviekg:hasAward ;
        rr:objectMap [
            rr:template "http://example.org/moviekg/award/{Award2}" ;
            rr:termType rr:IRI
        ]
    ] .
# 5. Reviews and Ratings from Letterboxd exports

# ratings.csv - numeric ratings
# Columns assumed: Name,Year,Rating,Date,Letterboxd URI

ex:TM_Rating_Reviews a rr:TriplesMap ;
    rml:logicalSource [
        rml:source "ratings.csv" ;
        rml:referenceFormulation ql:CSV
    ] ;
    rr:subjectMap [
        rr:template
"http://example.org/moviekg/review/ratings/{Name}_{Year}_{Date}" ;
        rr:class moviekg:Review
    ] ;
    rr:predicateObjectMap [
        rr:predicate moviekg:reviewOf ;
        rr:objectMap [
            rr:template "http://example.org/moviekg/movie/{Name}_{Year}" ;
            rr:termType rr:IRI
        ]
    ] ;
    rr:predicateObjectMap [
        rr:predicate moviekg:ratingValue ;
        rr:objectMap [ rml:reference "Rating" ; rr:datatype xsd:decimal ]
    ] ;
    rr:predicateObjectMap [
        rr:predicate moviekg:reviewDate ;
        rr:objectMap [ rml:reference "Date" ; rr:datatype xsd:date ]
    ] ;
    rr:predicateObjectMap [
        rr:predicate moviekg:letterboxdUri ;
        rr:objectMap [ rml:reference "Letterboxd URI" ; rr:datatype xsd:anyURI
]
    ] .

# reviews.csv - text reviews
 # Columns assumed: Name,Year,Rating,Review,Watched Date,Date,Tags,Letterboxd
 URI

 ex:TM_TextReviews a rr:TriplesMap ;
    rml:logicalSource [
        rml:source "reviews.csv" ;
        rml:referenceFormulation ql:CSV
    ] ;
    rr:subjectMap [
        rr:template
 "http://example.org/moviekg/review/text/{Name}_{Year}_{Date}" ;
        rr:class moviekg:Review
    ] ;
    rr:predicateObjectMap [
        rr:predicate moviekg:reviewOf ;
        rr:objectMap [
            rr:template "http://example.org/moviekg/movie/{Name}_{Year}" ;
            rr:termType rr:IRI
        ]
    ] ;
    rr:predicateObjectMap [
        rr:predicate moviekg:ratingValue ;
        rr:objectMap [ rml:reference "Rating" ; rr:datatype xsd:decimal ]
    ] ;
    rr:predicateObjectMap [
        rr:predicate moviekg:reviewText ;
        rr:objectMap [ rml:reference "Review" ; rr:datatype xsd:string ]
    ] ;
    rr:predicateObjectMap [
        rr:predicate moviekg:watchedDate ;
        rr:objectMap [ rml:reference "Watched Date" ; rr:datatype xsd:date ]
    ] ;
    rr:predicateObjectMap [
        rr:predicate moviekg:reviewDate ;
        rr:objectMap [ rml:reference "Date" ; rr:datatype xsd:date ]
    ] ;
    rr:predicateObjectMap [
        rr:predicate moviekg:tagsText ;
        rr:objectMap [ rml:reference "Tags" ; rr:datatype xsd:string ]
    ] ;
    rr:predicateObjectMap [
        rr:predicate moviekg:letterboxdUri ;
        rr:objectMap [ rml:reference "Letterboxd URI" ; rr:datatype xsd:anyURI
 ]
    ] .
```

```
# watchlist.csv - treat as "planned" reviews
# Columns assumed: Name,Year,Date,Letterboxd URI

ex:TM_Watchlist a rr:TriplesMap ;
   rml:logicalSource [
       rml:source "watchlist.csv" ;
       rml:referenceFormulation ql:CSV
   ] ;
   rr:subjectMap [
       rr:template
"http://example.org/moviekg/review/watchlist/{Name}_{Year}_{Date}" ;
       rr:class moviekg:Review
   ] ;
   rr:predicateObjectMap [
       rr:predicate moviekg:reviewOf ;
       rr:objectMap [
           rr:template "http://example.org/moviekg/movie/{Name}_{Year}" ;
           rr:termType rr:IRI
       ]
   ] ;
   rr:predicateObjectMap [
       rr:predicate moviekg:reviewDate ;
       rr:objectMap [ rml:reference "Date" ; rr:datatype xsd:date ]
   ] ;
   rr:predicateObjectMap [
       rr:predicate moviekg:letterboxdUri ;
       rr:objectMap [ rml:reference "Letterboxd URI" ; rr:datatype xsd:anyURI
]
   ] .

# watched.csv - "watched" events
# Columns assumed: Name,Year,Date,Letterboxd URI

ex:TM_Watched a rr:TriplesMap ;
   rml:logicalSource [
       rml:source "watched.csv" ;
       rml:referenceFormulation ql:CSV
   ] ;
   rr:subjectMap [
       rr:template
"http://example.org/moviekg/review/watched/{Name}_{Year}_{Date}" ;
       rr:class moviekg:Review
   ] ;
   rr:predicateObjectMap [
       rr:predicate moviekg:reviewOf ;
       rr:objectMap [
           rr:template "http://example.org/moviekg/movie/{Name}_{Year}" ;
           rr:termType rr:IRI
       ]
   ] ;
   rr:predicateObjectMap [
       rr:predicate moviekg:watchedDate ;
       rr:objectMap [ rml:reference "Date" ; rr:datatype xsd:date ]
   ] ;
   rr:predicateObjectMap [
       rr:predicate moviekg:letterboxdUri ;
       rr:objectMap [ rml:reference "Letterboxd URI" ; rr:datatype xsd:anyURI
]
   ] .

# 6. External link: owl:sameAs to Letterboxd using Letterboxd URI

# Attach owl:sameAs from Movie to Letterboxd URI using ratings.csv
# (duplicates are harmless if same movie has multiple ratings rows)

ex:TM_LetterboxdLinks a rr:TriplesMap ;
   rml:logicalSource [
       rml:source "ratings.csv" ;
       rml:referenceFormulation ql:CSV
   ] ;
   rr:subjectMap [
       rr:template "http://example.org/moviekg/movie/{Name}_{Year}" ;
       rr:class moviekg:Movie
   ] ;
   rr:predicateObjectMap [
       rr:predicate owl:sameAs ;
       rr:objectMap [
           rml:reference "Letterboxd URI" ;
           rr:termType rr:IRI
       ]
   ] .
```

Instances :

```
@prefix ex: <http://example.org/mapping#> .
@prefix moviekg: <http://example.org/moviekg#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rml: <http://w3id.org/rml/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

```
<http://example.org/moviekg/award/Academy%20Award%20Nomination%20for%20Best%2
0Actor>
 a moviekg:Award;
 moviekg:awardLabel "Academy Award Nomination for Best Actor" .

<http://example.org/moviekg/award/Academy%20Award%20Nomination%20for%20Best%2
0Director>
 a moviekg:Award;
 moviekg:awardLabel "Academy Award Nomination for Best Director" .

<http://example.org/moviekg/award/Academy%20Award%20Nomination%20for%20Best%2
0Picture>
 a moviekg:Award;
 moviekg:awardLabel "Academy Award Nomination for Best Picture" .

<http://example.org/moviekg/award/Academy%20Award%20for%20Best%20Actor> a
moviekg:Award;
 moviekg:awardLabel "Academy Award for Best Actor" .

<http://example.org/moviekg/award/Academy%20Award%20for%20Best%20Actress> a
moviekg:Award;
 moviekg:awardLabel "Academy Award for Best Actress" .

<http://example.org/moviekg/award/Academy%20Award%20for%20Best%20Animated%20F
eature>
 a moviekg:Award;
 moviekg:awardLabel "Academy Award for Best Animated Feature" .

<http://example.org/moviekg/award/Academy%20Award%20for%20Best%20Cinematograp
hy> a
   moviekg:Award;
 moviekg:awardLabel "Academy Award for Best Cinematography" .

<http://example.org/moviekg/award/Academy%20Award%20for%20Best%20Director> a
moviekg:Award;
 moviekg:awardLabel "Academy Award for Best Director" .

<http://example.org/moviekg/award/Academy%20Award%20for%20Best%20Film%20Editi
ng> a
   moviekg:Award;
 moviekg:awardLabel "Academy Award for Best Film Editing" .

<http://example.org/moviekg/award/Academy%20Award%20for%20Best%20Original%20S
creenplay>
 a moviekg:Award;
 moviekg:awardLabel "Academy Award for Best Original Screenplay" .
```

```
<http://example.org/moviekg/award/Academy%20Award%20for%20Best%20Picture> a
moviekg:Award;
 moviekg:awardLabel "Academy Award for Best Picture" .

<http://example.org/moviekg/award/Academy%20Award%20for%20Best%20Supporting%2
0Actor>
 a moviekg:Award;
 moviekg:awardLabel "Academy Award for Best Supporting Actor" .

<http://example.org/moviekg/award/Academy%20Award%20for%20Best%20Visual%20Eff
ects>
 a moviekg:Award;
 moviekg:awardLabel "Academy Award for Best Visual Effects" .

<http://example.org/moviekg/award/BAFTA%20Award%20for%20Best%20Editing> a
moviekg:Award;
 moviekg:awardLabel "BAFTA Award for Best Editing" .

<http://example.org/moviekg/award/BAFTA%20Award%20for%20Best%20Film%20not%20i
n%20the%20English%20Language>
 a moviekg:Award;
 moviekg:awardLabel "BAFTA Award for Best Film not in the English Language" .

<http://example.org/moviekg/award/BAFTA%20Award%20for%20Best%20Sound> a
moviekg:Award;
 moviekg:awardLabel "BAFTA Award for Best Sound" .

<http://example.org/moviekg/award/Berlin%20International%20Film%20Festival%20
Golden%20Bear>
 a moviekg:Award;
 moviekg:awardLabel "Berlin International Film Festival Golden Bear" .

<http://example.org/moviekg/award/Empire%20Award%20for%20Best%20Film> a
moviekg:Award;
 moviekg:awardLabel "Empire Award for Best Film" .

<http://example.org/moviekg/award/Golden%20Globe%20for%20Best%20Motion%20Pict
ure%20%E2%80%93%20Drama>
 a moviekg:Award;
 moviekg:awardLabel "Golden Globe for Best Motion Picture - Drama" .

<http://example.org/moviekg/award/Online%20Film%20Critics%20Award%20for%20Bes
t%20DVD>
 a moviekg:Award;
 moviekg:awardLabel "Online Film Critics Award for Best DVD" .
```

```
<http://example.org/moviekg/award/Palme%20d%27Or%20at%20Cannes> a
moviekg:Award;
 moviekg:awardLabel "Palme d'Or at Cannes" .

<http://example.org/moviekg/award/Saturn%20Award%20for%20Best%20Science%20Fic
tion%20Film>
 a moviekg:Award;
 moviekg:awardLabel "Saturn Award for Best Science Fiction Film" .

<http://example.org/moviekg/award/Saturn%20Award%20for%20Best%20Supporting%20
Actor>
 a moviekg:Award;
 moviekg:awardLabel "Saturn Award for Best Supporting Actor" .
```

movies_shacl.ttl:

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix moviekg: <http://example.org/moviekg#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

# MOVIE SHAPE

moviekg:MovieShape a sh:NodeShape ;
   sh:targetClass moviekg:Movie ;

   # Title (optional but recommended)
   sh:property [
       sh:path moviekg:movieTitle ;
       sh:datatype xsd:string ;
       sh:minCount 0 ;
       sh:maxCount 1 ;
       sh:severity sh:Warning ;
       sh:message "Movie should have a movieTitle (string)." ;
   ] ;

   # Genre (optional but recommended)
   sh:property [
       sh:path moviekg:hasGenre ;
       sh:minCount 0 ;
       sh:severity sh:Warning ;
       sh:message "Movie should have at least one genre." ;
   ] ;

   # Director (optional but recommended)
   sh:property [
       sh:path moviekg:hasDirector ;
       sh:minCount 0 ;
       sh:severity sh:Warning ;
       sh:message "Movie should have at least one director." ;
   ] ;

   # Actor (optional)
   sh:property [
       sh:path moviekg:hasActor ;
       sh:minCount 0 ;
       sh:severity sh:Info ;
   ] ;

   # Release year (optional)
   sh:property [
       sh:path moviekg:releaseYear ;
       sh:datatype xsd:gYear ;
       sh:minCount 0 ;
       sh:maxCount 1 ;
       sh:severity sh:Info ;
       sh:message "If present, releaseYear should be a valid xsd:gYear." ;
   ] .

# REVIEW SHAPE

moviekg:ReviewShape a sh:NodeShape ;
   sh:targetClass moviekg:Review ;

   sh:property [
       sh:path moviekg:ratingValue ;
       sh:datatype xsd:decimal ;
       sh:minCount 0 ;
       sh:severity sh:Warning ;
       sh:message "Review should have ratingValue (0-10)." ;
   ] ;

   sh:property [
       sh:path moviekg:reviewText ;
       sh:datatype xsd:string ;
       sh:minCount 0 ;
       sh:severity sh:Info ;
   ] .


# CERTIFICATE SHAPE
```

```
moviekg:CertificateShape a sh:NodeShape ;
   sh:targetClass moviekg:Certificate ;

   sh:property [
       sh:path moviekg:certificateCode ;
       sh:datatype xsd:string ;
       sh:minCount 0 ;
       sh:severity sh:Warning ;
       sh:message "Certificate codes normally use letters, digits, /, or -."
;
   ] .

# PERSON SHAPE

moviekg:PersonShape a sh:NodeShape ;
   sh:targetClass moviekg:Person ;

   sh:property [
       sh:path moviekg:personName ;
       sh:datatype xsd:string ;
       sh:minCount 0 ;
       sh:maxCount 1 ;
       sh:severity sh:Warning ;
       sh:message "Person should have a name." ;
   ] .

# GENRE SHAPE
moviekg:GenreShape a sh:NodeShape ;
   sh:targetClass moviekg:Genre ;

   sh:property [
       sh:path moviekg:genreLabel ;
       sh:datatype xsd:string ;
       sh:minCount 0 ;
       sh:severity sh:Info ;
   ] .

# PRODUCTION COMPANY SHAPE

moviekg:ProdCompanyShape a sh:NodeShape ;
   sh:targetClass moviekg:ProductionCompany ;

   sh:property [
       sh:path moviekg:companyName ;
       sh:datatype xsd:string ;
       sh:minCount 0 ;
```

```
        sh:severity sh:Info ;
    ] .

# AWARD SHAPE

moviekg:AwardShape a sh:NodeShape ;
    sh:targetClass moviekg:Award ;

    sh:property [
        sh:path moviekg:awardName ;
        sh:datatype xsd:string ;
        sh:minCount 0 ;
        sh:severity sh:Info ;
    ] .
```