



Università degli Studi dell' Aquila

Open Data and Web Services (ODWS)

Final Project

by

Jai Jobe, Lamin Ann & Michael Dubem Igbomezie

jai.jobe@student.univaq.it

lamin.ann@student.univaq.it

michaeldubem.igbomezie@student.univaq.it

Lecturers: Marco Autili & Claudio Pompilio

marco.autili@univaq.it , claudio.pompilio@univaq.it

Table of Contents

1. Introduction
2. Objectives
 - 2.1 Primary Objectives
 - 2.2 Secondary Objectives
3. Methodology
 - 3.1 Knowledge Domain Selection
 - 3.2 Data Gathering and Project Creation on OpenRefine
 - 3.3 Data Cleaning using Open Refine
 - 3.4 Data Enriching and Linking
 - 3.5 RDF Mapping and Exportation
 - 3.6 SPARQL Queries
 - 3.7 REST Web Service
 - 3.8 Java Client
4. Results
5. Conclusion
6. Appendix
 - 6.1 Dataset Description and License
7. References

1. Introduction

Datafication, Digitisation and System Integration has become a crucial part of today's world and connected systems, especially when we see the great potentials of integrated web services, third party libraries, model engineering and Open Data. These areas have collectively reduced the time it takes to set up functioning complex systems. It has reduced the error that comes from spreading efforts too thin in trying to "reinvent the wheel", and has further facilitated knowledge sharing and expertise integration.

Open Data refers to the information collected, produced or paid for by the public bodies and made freely available for re-use for any purpose. Its benefits are diverse and range from improved efficiency of public administrations, economic growth in the private sector to wider social welfare.

Thanks to Web services, clients or other web-based programs can access these data or information using standardized messaging protocols. Web services are built using different protocols in order to integrate with various applications. However, we use Representational State Transfer (REST) which is a set of guidelines that offers flexible implementation. While not all web services use the REST protocol, applications built with REST APIs are more lightweight, manageable and scalable.

The aim of this project is to implement a web service which harnesses Open Data to make some information available to users. This involved the gathering of licensed data in a certain knowledge domain, from open data platforms and communities, cleaning, enriching and linking of these datasets, and converting them into RDF graph so as to query these graphs and make their content available on a certain endpoint.

2. Objective

2.1 Primary Objective

The primary objective of this project is to successfully implement a REST service using the specified software which includes the OpenRefine, Eclipse IDE, Apache Tomcat, Apache CXF and Apache Maven, which exposes an RDF graph (extracted from the cleaned and converted data loaded in OpenRefine) using SPARQL queries.

2.2 Secondary Objective

The secondary objective of this project is focused on the type of information to be made available through the REST service. The area of interest of this project was the economic impact of the COVID-19 pandemic on 8 major countries in the world, the health impact on their population, their readiness to such an event prior to the outbreak and their reaction during the outbreak.

3. Methodology

3.1 Knowledge Domain Selection

Covid-19 is the world's greatest pandemic in the recent years. It has led to a dramatic loss of human life worldwide and presents an unprecedented challenge to public health, food systems and the world of work. The economic and social disruption caused by the pandemic is devastating: tens of millions of people are at risk of falling into extreme poverty, while the number of undernourished people, currently estimated at nearly 690 million, could increase by up to 132 million by the end of the year. These factors triggered our interest in picking different datasets about COVID-19.

3.2 Data Gathering and Project Creation on Open Refine

Kaggle is an online community platform that allow users to find and publish datasets. Below are the selected Datasets downloaded from Kaggle:

- COVID 19 Impact on Global Economy

Covid 19 Impact On Global Economy

Impact of covid 19 on economy

Data Code (0) Discussion (2) Metadata

About Dataset

Usability ⓘ
7.65

Context
This dataset has GDP and covid 19 cases information from many country.

Acknowledgements
Cite this dataset
Vitenu-Sackey, Prince Asare (2020), "The Impact of Covid-19 Pandemic on the Global Economy: Emphasis on Poverty Alleviation and Economic Growth", Mendeley Data, V1, doi: 10.17632/b2wvnbnpj9.1
<http://dx.doi.org/10.17632/b2wvnbnpj9.1>

License
Data files © Original Authors

Expected update frequency
Never

- Economic Data for - 9 Countries (1980-2020)

Economic Data - 9 Countries (1980-2020)

Movement of Stock Market with Macroeconomic changes



Data Code (0) Discussion (0) Metadata

About Dataset

The dataset contains data for 8 countries and one special administrative region (China, France, Germany, Hong Kong, India, Japan, Spain, United Kingdom and United States of America) from 1980 through 2020. It includes major macroeconomic factors like inflation, unemployment, GDP, exchange rate (base USD) and per capita income. Apart from that it has the stock prices of the respective country's major stock index which can help in analysing the data set to identify the impact of major macroeconomic variables on the movement of stock index prices.

Usability ⓘ

9.12

License

CC BY-NC-SA 4.0

Expected update frequency

Annually

Investing Economics

Economic Data - 9 Countries (1980-2020).csv (34 KIB)

Download

Detail Compact Column

10 of 14 columns

About this file

From 1980 through 2020, data is available for eight nations and one special administrative region (China, France, Germany, Hong Kong, India, Japan, Spain, the United Kingdom, and the United States of America). Major macroeconomic variables such as inflation, unemployment, GDP, exchange rate (base USD), and per capita income are included.

Data Explorer

34 KIB

Economic Data - 9 Countries...

- Countries by Government Budget per Capita

List_of_countries_by_government_budget_per_capita

List_of_countries_by_government_budget_per_capita.csv



Data Code (0) Discussion (0) Metadata

About Dataset

Usability ⓘ

5.88

License

CC0: Public Domain

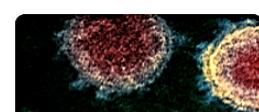
Expected update frequency

Not specified

- COVID 19 Case Reported in 2020

COVID-19 Dataset

Number of Confirmed, Death and Recovered cases every day across the globe



Data Code (581) Discussion (41) Metadata

About Dataset

Usability ⓘ

10.00

License

Other (specified in description)

Expected update frequency

Daily

Context

- A new coronavirus designated 2019-nCoV was first identified in Wuhan, the capital of China's Hubei province
- People developed pneumonia without a clear cause and for which existing vaccines or treatments were not effective.
- The virus has shown evidence of human-to-human transmission
- Transmission rate (rate of infection) appeared to escalate in mid-January 2020
- As of 30 January 2020, approximately 8,243 cases have been confirmed

Content

- full_grouped.csv - Day to day country wise no. of cases (Has County/State/Province level data)
- covid19clean_complete.csv - Day to day country wise no. of cases (Doesn't have County/State/Province level data)

- Public Health Safety measure and severity

World Bank WDI 2.12 - Health Systems

World Development Indicators for 2016

Data Code (30) Discussion (0) Metadata

About Dataset

World Bank - World Development Indicators: Health Systems

This is a digest of the information described at <http://wdi.worldbank.org/table/2.12#> It describes various health spending per capita by Country, as well as doctors, nurses and midwives, and specialist surgical staff per capita

Content

Notes, explanations, etc.

1. There are countries/regions in the World Bank data not in the Covid-19 data, and countries/regions in the Covid-19 data with no World Bank data. This is unavoidable.
2. There were political decisions made in both datasets that may cause problems. I chose to go forward with the data as presented, and did not attempt to modify the decisions made by the dataset creators (e.g., the names of countries, what is and is not a country, etc.).

Columns are as follows:

1. Country_Region: the region as used in Kaggle Covid-19 spread data challenges.

To create a project on OpenRefine; we imported the dataset and select ‘Create Project’ at the top right corner.

Project created with dataset that contains 369 number of rows.

3.3 Data Cleaning using Open Refine

Almost every dataset we encountered was messy. There were inconsistencies in the way the data was entered, this made the data difficult to analyse and that made it compulsory for us to clean it.

Text facet was used on this dataset to correct country column ‘United States of America’ and multiple representation of the same category (misspelling). It also confirmed that all dates were having the same format.

We used **Cell transformation** on the date column. The aim was to convert the date column to Datetime format.

DATE	HDI	TC	TD	STI	POP	GDP CAP
Facet		295836866	0	0	21.08743924	9.636177357
Text filter		295836866	0	0	21.08743924	9.636177357
Edit cells		Transform...			21.08743924	9.636177357
Edit column		Common transforms			Trim leading and trailing whitespace	
Transpose					Collapse consecutive whitespace	
Sort...					Unescape HTML entities	
View					Replace Smart quotes with ascii	
Reconcile					To titlecase	
10/01/2020	0.752				To uppercase	
11/01/2020	0.752				To lowercase	
12/01/2020	0.752				To number	
13/01/2020	0.752	4.077537444	0	1.022	To date	
14/01/2020	0.752	4.077537444	0	1.022	To text	
15/01/2020	0.752	4.077537444	0.693147181	2.119	To null	
16/01/2020	0.752	4.077537444	0.693147181	2.119	To empty string	
17/01/2020	0.752	4.143134726	0.693147181	2.119		
18/01/2020	0.752	4.382026635	0.693147181	2.119		

Redundant data: this means there are multiple copies of the same information spread over the dataset.

We **starred** all redundant rows on this particular dataset, applied '**Facet by star**' and then **removed matching rows** from this particular dataset.

CODE	COUNTRY	DATE	HDI
All	Afghanistan	31/12/2019	0.498
Transform	Afghanistan	01/01/2020	0.498
Edit all columns	Afghanistan	02/01/2020	0.498
Facet	Afghanistan	03/01/2020	0.498
Edit rows	Star rows		
Edit columns	Unstar rows		
View	Flag rows		
	Unflag rows		
	Remove matching rows		
9. AFG	Afghanistan	10/01/2020	0.498
10. AFG	Afghanistan	11/01/2020	0.498
11. AFG	Afghanistan	12/01/2020	0.498

CODE	COUNTRY	DATE	HDI
All	Afghanistan	31/12/2019	0.498
Transform	Afghanistan	01/01/2020	0.498
Edit all columns	Afghanistan	02/01/2020	0.498
Facet	Afghanistan	03/01/2020	0.498
Edit rows	Star rows		
Edit columns	Unstar rows		
View	Flag rows		
	Unflag rows		
	Remove matching rows		
9. AFG	Afghanistan	10/01/2020	0.498
10. AFG	Afghanistan	11/01/2020	0.498
11. AFG	Afghanistan	12/01/2020	0.498

2402 rows

Show as: rows records Show: 5 10 25 50 100 500 1000 rows < first < previous

All	CODE	COUNTRY	DATE	HDI	TC	TD	STI	POP	GDP_CAP
1. CHN	China	31/12/2019	0.752	3.295836866	0	0	21.08743924	9.636177357	
2. CHN	China	01/01/2020	0.752	3.295836866	0	0	21.08743924	9.636177357	
3. CHN	China	02/01/2020	0.752	3.295836866	0	0	21.08743924	9.636177357	
4. CHN	China	03/01/2020	0.752	3.784189634	0	0	21.08743924	9.636177357	
5. CHN	China	04/01/2020	0.752	3.784189634	0	0	21.08743924	9.636177357	
6. CHN	China	05/01/2020	0.752	4.077537444	0	1.022450928	21.08743924	9.636177357	
7. CHN	China	06/01/2020	0.752	4.077537444	0	1.022450928	21.08743924	9.636177357	
8. CHN	China	07/01/2020	0.752	4.077537444	0	1.022450928	21.08743924	9.636177357	
9. CHN	China	08/01/2020	0.752	4.077537444	0	1.022450928	21.08743924	9.636177357	
10. CHN	China	09/01/2020	0.752	4.077537444	0	1.022450928	21.08743924	9.636177357	
11. CHN	China	10/01/2020	0.752	4.077537444	0	1.022450928	21.08743924	9.636177357	
12. CHN	China	11/01/2020	0.752	4.077537444	0	1.022450928	21.08743924	9.636177357	
13. CHN	China	12/01/2020	0.752	4.077537444	0	1.022450928	21.08743924	9.636177357	
14. CHN	China	13/01/2020	0.752	4.077537444	0	1.022450928	21.08743924	9.636177357	
15. CHN	China	14/01/2020	0.752	4.077537444	0	1.022450928	21.08743924	9.636177357	
16. CHN	China	15/01/2020	0.752	4.077537444	0.693147181	2.119863456	21.08743924	9.636177357	
17. CHN	China	16/01/2020	0.752	4.077537444	0.693147181	2.119863456	21.08743924	9.636177357	

This is the data with no redundancy. A total of 48016 matching rows were removed reducing the dataset to 2402 number of rows.

Similar actions were carried out on the remaining datasets.

3.4 Data Enriching and Linking

Reconciliation is the process of matching a dataset with external datasets. We Reconciled the country column by adding the Wikidata Standard, and selected the Wikidata Standard service, followed by the below steps:

Country	USD	Year
of China (100)	2735	2017-01-01T00:00:00Z
Germany (100)	20960	2017-01-01T00:00:00Z
Hong Kong (100)		
Japan (100)		
Spain (100)	10863	2017-01-01T00:00:00Z

OpenRefine Countries by Govt Budget per Capita csv

Facet / Filter Undo / Redo 15 / 15

7 matching rows (9 total)

Country	USD	Year
People's Republic of China	2735	2017-01-01T00:00:00Z
France	20960	2017-01-01T00:00:00Z
Germany	19550	2017-01-01T00:00:00Z
district of Hong Kong	8714	2017-01-01T00:00:00Z
India	559	2017-01-01T00:00:00Z
Japan	14964	2017-01-01T00:00:00Z
United Kingdom	10863	2017-01-01T00:00:00Z
United Kingdom	17202	2018-01-01T00:00:00Z
United States of America	20674	2017-01-01T00:00:00Z

Extensions: RDF ▾ Wikidata ▾

Show as: rows records Show: 5 10 25 50 100 500 1000 rows

« first < previous 1 of 1 page next > last »

Facet / Filter Undo / Redo 16 / 16

9 rows

Country	USD	Year
1. People's Republic of China	2735	2017-01-01T00:00:00Z
2. France	20960	2017-01-01T00:00:00Z
3. Germany	19550	2017-01-01T00:00:00Z
4. district of Hong Kong	8714	2017-01-01T00:00:00Z
5. India	559	2017-01-01T00:00:00Z
6. Japan	14964	2017-01-01T00:00:00Z
7. United Kingdom	10863	2017-01-01T00:00:00Z
8. United Kingdom	17202	2018-01-01T00:00:00Z
9. United States of America	20674	2017-01-01T00:00:00Z

We then enriched the dataset with the Wikidata URLs

OpenRefine Countries by Govt Budget per Capita csv

Facet / Filter Undo / Redo 16 / 16

9 rows

Country	USD	Year
1. People's Republic of China	2735	2017-01-01T00:00:00Z
2. France	20960	2017-01-01T00:00:00Z
3. Germany	19550	2017-01-01T00:00:00Z
4. district of Hong Kong	8714	2017-01-01T00:00:00Z
5. India	559	2017-01-01T00:00:00Z
6. Japan	14964	2017-01-01T00:00:00Z
7. United Kingdom	10863	2017-01-01T00:00:00Z
8. United Kingdom	17202	2018-01-01T00:00:00Z
9. United States of America	20674	2017-01-01T00:00:00Z

Facet / Filter Undo / Redo 16 / 16

Add column based on column Country

New column name: wikidata_url

On error: set to blank store error copy value from original column

Expression: Language: General Refine Expression Language (GREL)

"https://wikidata.org/wiki/*cell.recon.match.id"

Preview History Starred Help

row	value
1.	https://wikidata.org/wiki/Q148
2.	https://wikidata.org/wiki/Q142
3.	https://wikidata.org/wiki/Q183
4.	https://wikidata.org/wiki/Q50256
5.	https://wikidata.org/wiki/Q668
6.	https://wikidata.org/wiki/Q17
7.	https://wikidata.org/wiki/Q29
8.	https://wikidata.org/wiki/Q145
9.	https://wikidata.org/wiki/Q30

OpenRefine Countries by Govt Budget per Capita CSV

Facet / Filter Undo / Redo 17 / 17

9 rows

Country	wikidata_url	USD	Year
1. People's Republic of China	https://wikidata.org/wiki/Q148	2735	2017-01-01T00:00:00Z
2. France	https://wikidata.org/wiki/Q142	20960	2017-01-01T00:00:00Z
3. Germany	https://wikidata.org/wiki/Q183	19550	2017-01-01T00:00:00Z
4. district of Hong Kong	https://wikidata.org/wiki/Q50256	8714	2017-01-01T00:00:00Z
5. India	https://wikidata.org/wiki/Q668	559	2017-01-01T00:00:00Z
6. Japan	https://wikidata.org/wiki/Q17	14964	2017-01-01T00:00:00Z
7. Spain	https://wikidata.org/wiki/Q29	10863	2017-01-01T00:00:00Z
8. United Kingdom	https://wikidata.org/wiki/Q145	17202	2018-01-01T00:00:00Z
9. United States of America	https://wikidata.org/wiki/Q30	20674	2017-01-01T00:00:00Z

This is the resulting dataset after enriching and linking.

3.5 RDF Mapping and Exportation

RDF schema alignment

RDF Schema alignment

The RDF schema alignment skeleton below specifies how the RDF data that will get generated from your grid-shaped data. The cells in each record of your data will get placed into nodes within the skeleton. Configure the skeleton by specifying which column to substitute into which node.

Base URI: <http://localhost:3333/> Edit

RDF skeleton **RDF Preview**

Available prefixes: [rdf](#) [owl](#) [rdfs](#) [foaf](#) [+Add](#) [Manage](#)

(Row index) URI Add type	<input checked="" type="checkbox"/> X → property? → <input checked="" type="checkbox"/> X → property? → <input checked="" type="checkbox"/> X → property? → <input checked="" type="checkbox"/> X → property? → Add property	<input type="checkbox"/> Country Cell <input type="checkbox"/> wikidata_url Cell <input type="checkbox"/> USD Cell <input type="checkbox"/> Year Cell
-----------------------------	--	--

Add another root node **Save**

OK **Cancel**

We edited the Base URI

RDF Schema alignment

The RDF schema alignment skeleton below specifies how the RDF data that will get generated from your grid-shaped data. The cells in each record of your data will get placed into nodes within the skeleton. Configure the skeleton by specifying which column to substitute into which node.

Base URI: <http://localhost:3333/> Edit <http://odws.univqa.it/countriesbudget/>

RDF skeleton **RDF Preview** **Apply** **Cancel**

Available prefixes: [rdf](#) [owl](#) [rdfs](#) [foaf](#) [+Add](#) [Manage](#)

(Row index) URI Add type	<input checked="" type="checkbox"/> X → property? → <input checked="" type="checkbox"/> X → property? → <input checked="" type="checkbox"/> X → property? → <input checked="" type="checkbox"/> X → property? → Add property	<input type="checkbox"/> Country Cell <input type="checkbox"/> wikidata_url Cell <input type="checkbox"/> USD Cell <input type="checkbox"/> Year Cell
-----------------------------	--	--

Add another root node **Save**

OK **Cancel**

We then set up the identity in URIs

RDF node

Use content...

(Row index)
 Country
 wikidata_url
 USD
 Year
 Constant value

Content used ...

URI
 Text
 Language
 Integer
 Non-integer
 Date (YYYY-MM-DD)
 Date/time (YYYY-MM-DD HH:MM:SS)
 Boolean
 Custom (specify type URI)

 Blank

Use expression...

value
Preview edit

OK **Cancel**

Linked Data Principles

Preview URI value

General Refine Expression Language (GREL) ▾

row	value	value	resolved against the base URI
1.	China	China	http://odws.univaq.it/countriesbudget/China
2.	France	France	http://odws.univaq.it/countriesbudget/France
3.	Germany	Germany	http://odws.univaq.it/countriesbudget/Germany
4.	Hong Kong	Hong Kong	http://odws.univaq.it/Hong%20Kong
5.	India	India	http://odws.univaq.it/countriesbudget/India
6.	Japan	Japan	http://odws.univaq.it/countriesbudget/Japan
7.	Spain	Spain	http://odws.univaq.it/countriesbudget/Spain

Default prefixes

RDF Schema alignment

The RDF schema alignment skeleton below specifies how the RDF data that will get generated from your grid-shaped data. The cells in each record of your data will get placed into nodes within the skeleton. Configure the skeleton by specifying which column to substitute into which node.

Base URI: <http://odws.univaq.it/countriesbudget/> Edit

RDF skeleton RDF Preview

Available prefixes: [rdf](#) [owl](#) [rdfs](#) [foaf](#) + Add ⚙ Manage

(Row index) URI Add type

List of defined prefixes

Add prefix

Prefix URI	Delete	Refresh
rdf http://www.w3.org/1999/02/22-rdf-syntax-ns#	Delete	Refresh
owl http://www.w3.org/2002/07/owl#	Delete	Refresh
rdfs http://www.w3.org/2000/01/rdf-schema#	Delete	Refresh
foaf http://xmlns.com/foaf/0.1/	Delete	Refresh

OK Cancel

Add another root node Save

OK Cancel

We added prefixes for dbpedia.org and schema.org

Add new prefix

Prefix:

URI: (a suggestion by [prefix.cc](#) is provided)

OK Cancel Vocabulary import...

Add new prefix

Prefix:

URI:

OK Cancel Vocabulary import...

Then we mapped the country cell

RDF Schema alignment

The RDF schema alignment skeleton below specifies how the RDF data that will get generated from your grid-shaped data. The cells in each record of your data will get placed into nodes within the skeleton. Configure the skeleton by specifying which column to substitute into which node.

Base URI: <http://odws.univaq.it/countriesbudget/> Edit

RDF skeleton RDF Preview

Available prefixes: [rdf](#) [owl](#) [rdfs](#) [foaf](#) [dbp](#) [sch](#) + Add ⚙ Manage

(Row index) URI Add type

☰ ×→ property?→ Search for property:
 ×→ property?→
 ×→ property?→
 ×→ property?→
 Add property

Select an item from the list:

- dbp:country <http://dbpedia.org/ontology/c>
- dbp:countryCode <http://dbpedia.org/ontology/c>
- dbp:countryOrigin <http://dbpedia.org/ontology/c>
- dbp:countryRank <http://dbpedia.org/ontology/c>
- dbp:countryWithFirstAstronaut <http://dbpedia.org/ontology/c>
- dbp:countryWithFirstSatellite <http://dbpedia.org/ontology/c>
- dbp:countryWithFirstSatelliteLa <http://dbpedia.org/ontology/c>
- dbp:countryWithFirstSpaceflight <http://dbpedia.org/ontology/c>

Your item not in the list?

Save

Add another root node

OK Cancel

Similarly, wikidata_url cell, USD cell and Year cell were all mapped in the same way.

Below is the resulting mapping and the RDF preview

RDF Schema alignment

The RDF schema alignment skeleton below specifies how the RDF data that will get generated from your grid-shaped data. The cells in each record of your data will get placed into nodes within the skeleton. Configure the skeleton by specifying which column to substitute into which node.

Base URI: <http://odws.univaq.it/countriesbudget/> Edit

RDF skeleton **RDF Preview**

Available prefixes: rdf owl rdfs foaf dbp sch + Add Manage

(Row index) URI	dbp:country →	Country Cell
Add type	dbp:sameas →	wikidata_url Cell
	dbp:budgetpercapita →	USD Cell
	dbp:year →	Year Cell
	Add property	

Add another root node Save

OK **Cancel**

RDF Schema alignment

The RDF schema alignment skeleton below specifies how the RDF data that will get generated from your grid-shaped data. The cells in each record of your data will get placed into nodes within the skeleton. Configure the skeleton by specifying which column to substitute into which node.

Base URI: <http://odws.univaq.it/countriesbudget/> Edit

RDF skeleton **RDF Preview**

This is a sample turtle representation of (up-to) the first 10 rows

```

@prefix dbp: <http://dbpedia.org/ontology/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix sch: <https://schema.org/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

<http://odws.univaq.it/countriesbudget/0> dbp:country "China".
dbp:sameas "https://wikidata.org/wiki/Q148".
dbp:budgetpercapita "2735"^^<http://www.w3.org/2001/XMLSchema#int>.
dbp:year "2017-01-01T00:00Z"^^<http://www.w3.org/2001/XMLSchema#dateTime>.
dbp:country "France".
dbp:sameas "https://wikidata.org/wiki/Q142".
<http://odws.univaq.it/countriesbudget/1> dbp:country "Germany".
dbp:sameas "https://wikidata.org/wiki/Q183".
dbp:budgetpercapita "2960"^^<http://www.w3.org/2001/XMLSchema#int>.
dbp:year "2017-01-01T00:00Z"^^<http://www.w3.org/2001/XMLSchema#dateTime>.
<http://odws.univaq.it/countriesbudget/2> dbp:country "India".
dbp:sameas "https://wikidata.org/wiki/Q685".

```

OK **Cancel**

Exportation

We exported the data as RDF/XML

OpenRefine project archive to file	<?xml version="1.0" encoding="UTF-8"?> <rdf:RDF xmlns:dbp="http://dbpedia.org/ontology/" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:owl="http://www.w3.org/2002/07/owl#" xmlns:sch="https://schema.org/" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" xmlns:foaf="http://xmlns.com/foaf/0.1/">
Tab-separated value	<rdf:Description rdf:about="http://odws.univaq.it/countriesbudget/0"> <dbp:country>China</dbp:country> <dbp:sameas> https://wikidata.org/wiki/Q148 </dbp:sameas> <dbp:budgetpercapita rdf:datatype="http://www.w3.org/2001/XMLSchema#int">2735</dbp:budgetpercapita> <dbp:year rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2017-01-01T00:00Z</dbp:year> </rdf:Description>
Comma-separated value	<rdf:Description rdf:about="http://odws.univaq.it/countriesbudget/1"> <dbp:country>France</dbp:country> <dbp:sameas> https://wikidata.org/wiki/Q142 </dbp:sameas> <dbp:budgetpercapita rdf:datatype="http://www.w3.org/2001/XMLSchema#int">2960</dbp:budgetpercapita> <dbp:year rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2017-01-01T00:00Z</dbp:year> </rdf:Description>
HTML table	<rdf:Description rdf:about="http://odws.univaq.it/countriesbudget/2"> <dbp:country>Germany</dbp:country> <dbp:sameas> https://wikidata.org/wiki/Q183 </dbp:sameas> <dbp:budgetpercapita rdf:datatype="http://www.w3.org/2001/XMLSchema#int">19550</dbp:budgetpercapita> <dbp:year rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2017-01-01T00:00Z</dbp:year> </rdf:Description>
Excel (xls)	<rdf:Description rdf:about="http://odws.univaq.it/countriesbudget/3"> <dbp:country>India</dbp:country> <dbp:sameas> https://wikidata.org/wiki/Q685 </dbp:sameas> <dbp:budgetpercapita rdf:datatype="http://www.w3.org/2001/XMLSchema#int">559</dbp:budgetpercapita> <dbp:year rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2017-01-01T00:00Z</dbp:year> </rdf:Description>
Excel 2007+ (xlsx)	<rdf:Description rdf:about="http://odws.univaq.it/countriesbudget/4"> <dbp:country>Japan</dbp:country> <dbp:sameas> https://wikidata.org/wiki/Q17 </dbp:sameas> <dbp:budgetpercapita rdf:datatype="http://www.w3.org/2001/XMLSchema#int">14964</dbp:budgetpercapita> <dbp:year rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2017-01-01T00:00Z</dbp:year> </rdf:Description>
ODF spreadsheet	<rdf:Description rdf:about="http://odws.univaq.it/countriesbudget/5"> <dbp:country>Spain</dbp:country> <dbp:sameas> https://wikidata.org/wiki/Q29 </dbp:sameas> <dbp:budgetpercapita rdf:datatype="http://www.w3.org/2001/XMLSchema#int">10863</dbp:budgetpercapita> <dbp:year rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2017-01-01T00:00Z</dbp:year> </rdf:Description>
Custom tabular exporter...	<rdf:Description rdf:about="http://odws.univaq.it/countriesbudget/6">
SQL Exporter...	
Templating...	
RDF as RDF/XML	
RDF as Turtle	
OpenRefine project archive to Google Drive...	
Google Sheets	
Wikibase edits...	
QuickStatements file	
Wikibase schema	

3.6 SPARQL Queries

SPARQL is fundamentally a language for formulating queries, through which information can be retrieved from datasets. However, since it is targeted at datasets published on the World Wide Web, it also provides a protocol for specifying SPARQL commands.

The following query types were used in the Apache Jena to retrieve the important information we needed to know from the linked datasets:

SELECT: the query returns a table in which columns represent variables and rows represent variable bindings matching the search pattern.

FROM clause: defines the graph to query. By default, if a query omits FROM clauses, the scope of the query is limited to the default graph (DEFAULTSET).

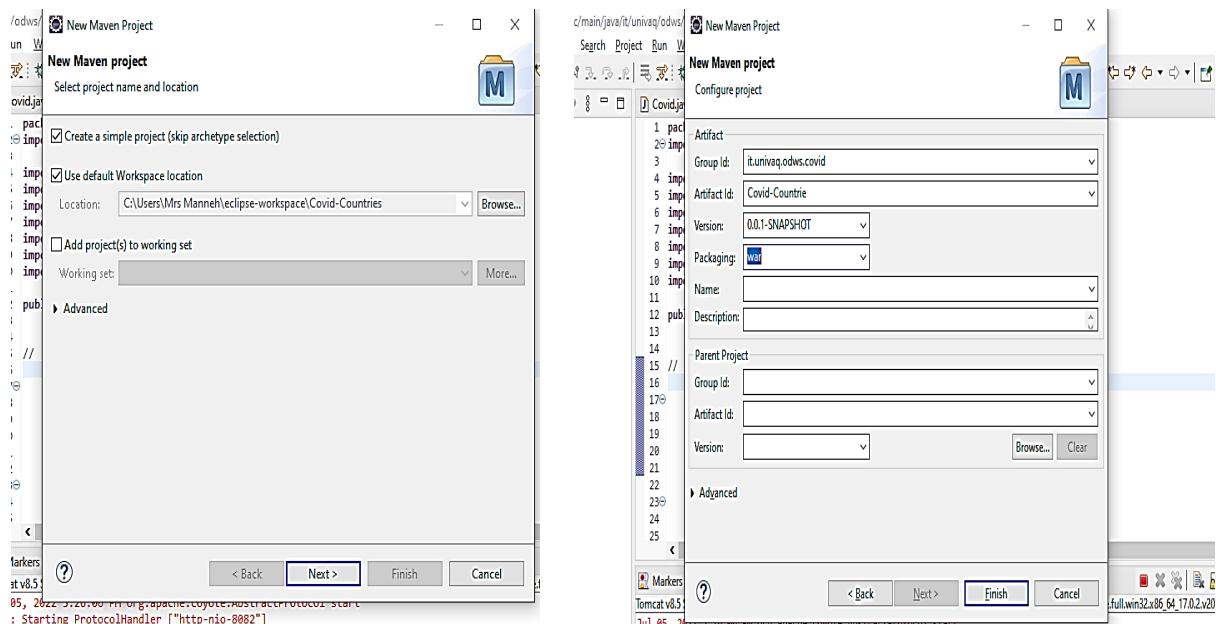
In the result content, we showed the query of individual graphs and a combined graph.

3.7 REST Web Service: Is an architectural style based on transferring representations of resources from a server to a client. We use rest and maven for the implementation of the project. The maven help us to load and install some of the dependencies that will be needed throughout the project. The rest help us to render our project as a client service in order to extract data from the RDF files.

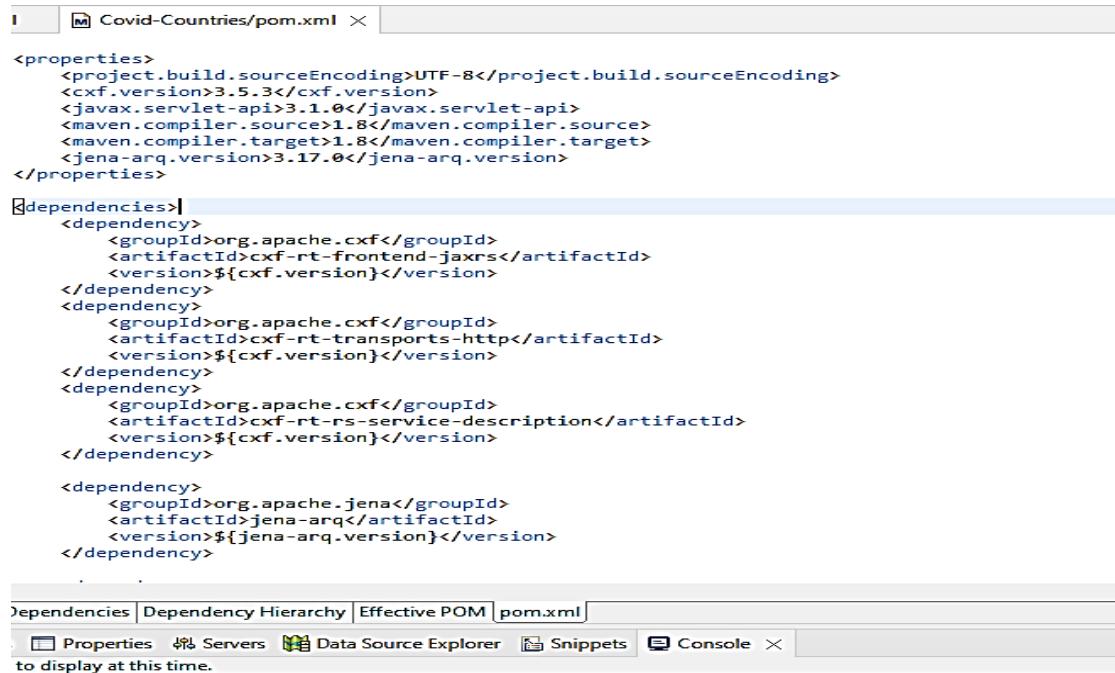
The plugins that were installed includes:

- Eclipse IDE
- Apache Tomcat
- Apache CXF
- Maven

We created a new Maven Project



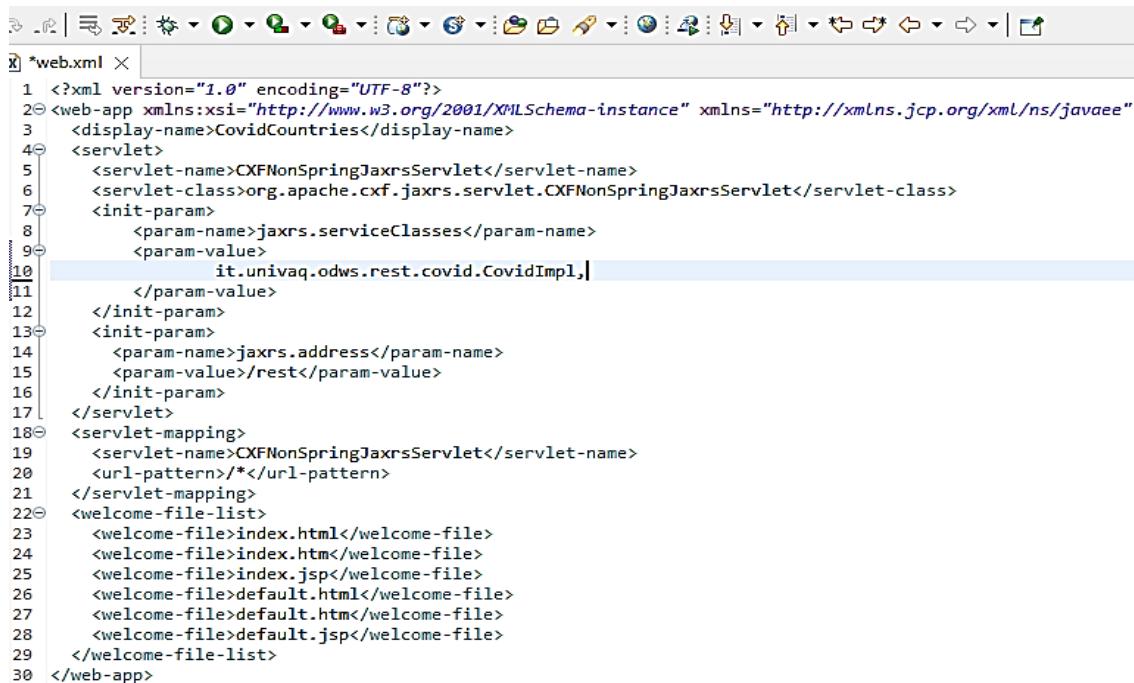
We edited the POM (Project Object Model) file



```
<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <cxf.version>3.5.3</cxf.version>
    <javax.servlet-api>3.1.0</javax.servlet-api>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
    <jena-arq.version>3.17.0</jena-arq.version>
</properties>

<dependencies>
    <dependency>
        <groupId>org.apache.cxf</groupId>
        <artifactId>cxfrt-frontend-jaxrs</artifactId>
        <version>${cxn.version}</version>
    </dependency>
    <dependency>
        <groupId>org.apache.cxf</groupId>
        <artifactId>cxfrt-transports-http</artifactId>
        <version>${cxn.version}</version>
    </dependency>
    <dependency>
        <groupId>org.apache.cxf</groupId>
        <artifactId>cxfrt-rs-service-description</artifactId>
        <version>${cxn.version}</version>
    </dependency>
    <dependency>
        <groupId>org.apache.jena</groupId>
        <artifactId>jena-arq</artifactId>
        <version>${jena-arq.version}</version>
    </dependency>
    .
    .
    .
</dependencies>
```

We edited the web XML file



```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee">
    <display-name>CovidCountries</display-name>
    <servlet>
        <servlet-name>CXFNonSpringJaxrsServlet</servlet-name>
        <servlet-class>org.apache.cxf.jaxrs.servlet.CXFNonSpringJaxrsServlet</servlet-class>
        <init-param>
            <param-name>jaxrs.serviceClasses</param-name>
            <param-value>
                it.univaq.odws.rest.covid.CovidImpl,
            </param-value>
        </init-param>
        <init-param>
            <param-name>jaxrs.address</param-name>
            <param-value>/rest</param-value>
        </init-param>
    </servlet>
    <servlet-mapping>
        <servlet-name>CXFNonSpringJaxrsServlet</servlet-name>
        <url-pattern>/*</url-pattern>
    </servlet-mapping>
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
        <welcome-file>index.htm</welcome-file>
        <welcome-file>index.jsp</welcome-file>
        <welcome-file>default.html</welcome-file>
        <welcome-file>default.htm</welcome-file>
        <welcome-file>default.jsp</welcome-file>
    </welcome-file-list>
</web-app>
```

We created a COVID interface

The screenshot shows two windows side-by-side. On the left is the 'New Java Interface' dialog, which allows creating a new Java interface. It has fields for 'Source folder' (Covid-Countries/src/main/java), 'Package' (it.univaq.odws.rest.covid), and 'Name' (Covid). It also includes sections for 'Modifiers' (public selected), 'Extended interfaces', and a checkbox for 'Generate comments'. On the right is the generated code for `Covid.java`, which defines a RESTful interface with several methods using annotations like @Path and @GET.

```

1 package it.univaq.odws.rest.covid;
2
3 import javax.ws.rs.GET;
4 import javax.ws.rs.Path;
5 import javax.ws.rs.PathParam;
6 import javax.ws.rs.Produces;
7 import javax.ws.rs.core.MediaType;
8
9 @Path("/countrieseconomicdata")
10 public interface Covid {
11
12
13    @GET
14    @Path("/{id}")
15    @Produces({MediaType.APPLICATION_JSON})
16    String getCovidById (@PathParam("id")int id);
17
18    @GET
19    @Path("/country/{country}")
20    @Produces({MediaType.APPLICATION_JSON})
21    String getCovidByCountry (@PathParam("country")String country);
22
23    @GET
24    @Path("/stockindex/{stockindex}")
25    @Produces({MediaType.APPLICATION_JSON})
26    String getCovidByStockIndex (@PathParam("stockindex")String stockindex);
27
28    @GET
29    @Path("/gdppercents/{gdppercents}")
30    @Produces({MediaType.APPLICATION_JSON})
31    String getCovidByGDPPercent (@PathParam("gdppercents")String gdppercents);

```

We created the implementation class, (CovidImpl), so as to implement the Covid interface.

The screenshot shows the Eclipse IDE with the `CovidImpl.java` file open. The code implements the `Covid` interface, loading an RDF dataset and converting result sets to JSON. It overrides the `getCovidById` method to return a dataset. Below the code editor, the Tomcat v8.5 server status shows it is running at `localhost:8082`.

```

1 package it.univaq.odws.rest.covid;
2
3 import org.apache.jena.query.Dataset;
4 import org.apache.jena.query.QueryExecution;
5 import org.apache.jena.query.QueryExecutionFactory;
6 import org.apache.jena.query.ReadWrite;
7 import org.apache.jena.query.ResultSet;
8 import org.apache.jena.query.ResultSetFormatter;
9 import org.apache.jena.riot.RDFDataMgr;
10
11 public class CovidImpl implements Covid{
12
13     private static final String COVID_DATASET = "Economic-Datas.ttl";
14     // private static final String COVID_DATASET = "Countries-Budget.rdf";
15
16     private Dataset loadDataset() {
17         Dataset dataset = RDFDataMgr.LoadDataset(COVID_DATASET);
18         dataset.begin(ReadWrite.READ);
19         return dataset;
20     }
21
22
23     private String convertResultSetToString	ResultSet resultSet) {
24         ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
25         ResultSetFormatter.outputAsJSON(byteArrayOutputStream, resultSet);
26         return byteArrayOutputStream.toString();
27     }
28
29
30     @Override
31     public String getCovidById(int id) {
32         Dataset dataset = loadDataset();
33
34         StringBuilder query = new StringBuilder();
35         query.append("PREFIX dbr: <http://dbpedia.org/resource/> \n");

```

We deployed our service

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```

<application xmlns="http://wadl.dev.java.net/2009/02" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <grammars/>
  <resources base="http://localhost:8082/Covid-Countries/rest">
    <resource path="/covidinfo">
      <resource path="/country/{country}">
        <param name="country" style="template" type="xs:string"/>
        <method name="GET">
          <request/>
          <response>
            <representation mediaType="application/json">
              <param name="result" style="plain" type="xs:string"/>
            </representation>
          </response>
        </method>
      </resource>
      <resource path="/countryCode/{countryCode}">
        <param name="countryCode" style="template" type="xs:string"/>
        <method name="GET">
          <request/>
          <response>
            <representation mediaType="application/json">
              <param name="result" style="plain" type="xs:string"/>
            </representation>
          </response>
        </method>
      </resource>
      <resource path="/covidbycountry/{country}">
        <param name="country" style="template" type="xs:string"/>
        <method name="GET">
          <request/>
          <response>
            <representation mediaType="application/json">
              <param name="result" style="plain" type="xs:string"/>
            </representation>
          </response>
        </method>
      </resource>
      <resource path="/stockindex/{stockindex}">

```

Available RESTful services:

Endpoint address: <http://localhost:8082/Covid-Countries/rest>
WADL : http://localhost:8082/Covid-Countries/rest?_wadl

We then queried this dataset by index 22

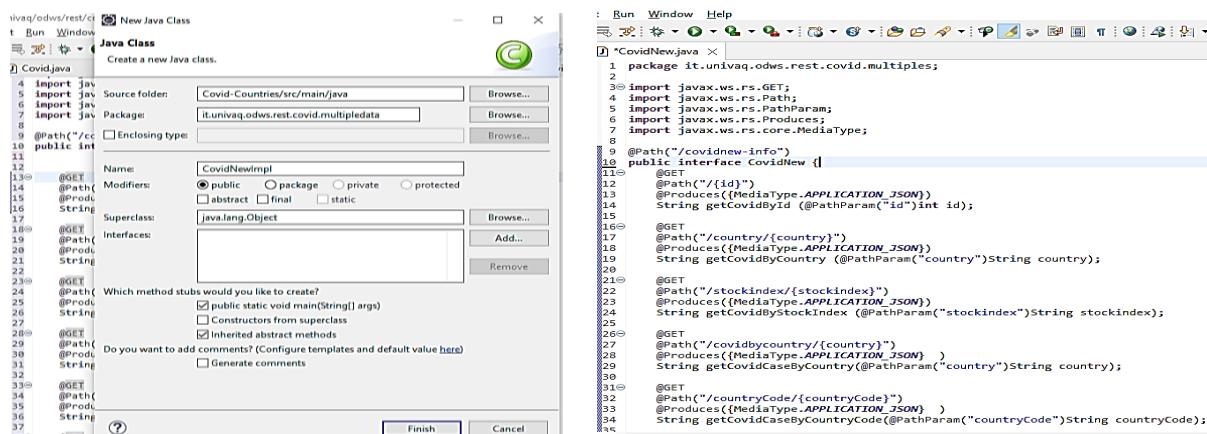
```
27
28
29
30@Override
31 public String getCovidById(int id) {
32     Dataset dataset = loadDataset();
33
34     StringBuilder query = new StringBuilder();
35     query.append("PREFIX dbp: <http://dbpedia.org/ontology/>").append(System.lineSeparator());
36     query.append("PREFIX sch: <https://schema.org/>").append(System.lineSeparator());
37     query.append("SELECT ?country ?stockindex ?sameasstockindex ?gdppercen?year ?indexprice ?inflationrate ?imdb_url ?wikidata_url");
38     query.append("WHERE {").append(System.lineSeparator());
39     query.append(" <http://odws.univaq.it/countrieseconomicdata/" + id + " dbp:country ?country.").append(System.lineSeparator());
40     query.append(" <http://odws.univaq.it/countrieseconomicdata/" + id + " dbp:stockindex ?stockindex.").append(System.lineSeparator());
41     query.append(" <http://odws.univaq.it/countrieseconomicdata/" + id + " dbp:sameasstockindex ?sameasstockindex.").append(System.lineSeparator());
42     query.append(" <http://odws.univaq.it/countrieseconomicdata/" + id + " dbp:gdppercen?gdppercen.").append(System.lineSeparator());
43     query.append(" <http://odws.univaq.it/countrieseconomicdata/" + id + " dbp:year ?year.").append(System.lineSeparator());
44     query.append(" <http://odws.univaq.it/countrieseconomicdata/" + id + " dbp:indexprice ?indexprice.").append(System.lineSeparator());
45     query.append(" <http://odws.univaq.it/countrieseconomicdata/" + id + " dbp:inflationrate ?inflationrate.").append(System.lineSeparator());
46     query.append(")").append(System.lineSeparator());
47
48     QueryExecution queryExecution = QueryExecutionFactory.create(query.toString(), dataset);
49     ResultSet resultSet = queryExecution.execSelect();
50     return convertResultSetToJsonString(resultSet);
51 }
52 }
```

We also queried by country Name

```
Covid-Countries/pom.xml Covid.java CovidImpl.java
2
3@Override
4 public String getCovidByCountry(String country) {
5     // TODO Auto-generated method stub
6     Dataset dataset = loadDataset();
7
8     StringBuilder query = new StringBuilder();
9     query.append("PREFIX dbp: <http://dbpedia.org/ontology/>").append(System.lineSeparator());
10    query.append("PREFIX sch: <https://schema.org/>").append(System.lineSeparator());
11    query.append("SELECT ?country ?year ?gdppercen ?indexprice ?stockindex ?sameasstockindex ?inflationrate ").append(System.lineSeparator());
12    query.append("WHERE {").append(System.lineSeparator());
13    query.append(" ?countrieseconomicdata dbp:country ?country.").append(System.lineSeparator());
14    query.append(" ?countrieseconomicdata dbp:stockindex ?stockindex.").append(System.lineSeparator());
15    query.append(" ?countrieseconomicdata dbp:sameasstockindex ?sameasstockindex.").append(System.lineSeparator());
16    query.append(" ?countrieseconomicdata dbp:year ?year.").append(System.lineSeparator());
17    query.append(" ?countrieseconomicdata dbp:gdppercen ?gdppercen.").append(System.lineSeparator());
18    query.append(" ?countrieseconomicdata dbp:indexprice ?indexprice.").append(System.lineSeparator());
19    query.append(" ?countrieseconomicdata dbp:inflationrate ?inflationrate.").append(System.lineSeparator());
20    query.append(" FILTER((LCASE(?country) = '" + country.toLowerCase() + "'')).append(System.lineSeparator());
21    query.append(")").append(System.lineSeparator());
22
23    QueryExecution queryExecution = QueryExecutionFactory.create(query.toString(), dataset);
24    ResultSet resultSet = queryExecution.execSelect();
25    return convertResultSetToJsonString(resultSet);
26 }
```

We filtered NASDAQ from the dataset

We then created another Package for implementing multiple RDF files.



```

] CovidNew.java | ] CovidNewImpl.java × |
1 package it.univaq.odws.rest.covid.multiples;
2
3 import java.io.ByteArrayOutputStream;
4
5 import org.apache.jena.query.Dataset;
6 import org.apache.jena.query.QueryExecution;
7 import org.apache.jena.query.QueryExecutionFactory;
8 import org.apache.jena.query.ReadWrite;
9 import org.apache.jena.query.ResultSet;
10 import org.apache.jena.query.ResultSetFormatter;
11 import org.apache.jena.riot.RDFDataMgr;
12
13 public class CovidNewImpl implements CovidNew{
14
15     private static final String COVID_DATASET = "Economic-Data.rdf";
16     private static final String COVID_DATASET_NEW = "Covid19-Impact.rdf";
17
18     private Dataset loadDataset() {
19         Dataset dataset = RDFDataMgr.LoadDataset(COVID_DATASET);
20         RDFDataMgr.read(dataset, COVID_DATASET_NEW);
21         dataset.begin(ReadWrite.READ);
22         return dataset;
23     }
24
25     private String convertResultSetToJsonString(ResultSet resultSet) {
26         ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
27         ResultSetFormatter.outputAsJSON(byteArrayOutputStream, resultSet);
28         return byteArrayOutputStream.toString();
29     }
30
31     @Override
32     public String getCovidById(int id) {
33         Dataset dataset = loadDataset();
34     }

```

We filtered by stock index NASDAQ in this data set.

```

// Filtering by Stockindex NASDAQ in the Data |
3 @Override
4 public String getCovidCaseByCountry(String country) {
5
6     Dataset dataset = loadDataset();
7
8     StringBuilder query = new StringBuilder();
9     query.append("PREFIX dbp: <http://dbpedia.org/ontology/>").append(System.lineSeparator());
10    query.append("PREFIX sch: <https://schema.org/>").append(System.lineSeparator());
11    query.append("SELECT ?country ?year ?gdppercen ?indexprice ?stockindex ?sameasstockindex ?inflationrate ").append(System.lineSeparator());
12    query.append("WHERE {}").append(System.lineSeparator());
13    query.append(" ?countrieseconomicdata dbp:country ?country").append(System.lineSeparator());
14    query.append(" ?countrieseconomicdata dbp:year ?year").append(System.lineSeparator());
15    query.append(" ?countrieseconomicdata dbp:gdppercen ?gdppercen").append(System.lineSeparator());
16    query.append(" ?countrieseconomicdata dbp:indexprice ?indexprice").append(System.lineSeparator());
17    query.append(" ?countrieseconomicdata dbp:stockindex ?stockindex").append(System.lineSeparator());
18    query.append(" FILTER(CONTAINS(?stockindex, \"NASDAQ\")) && ?country = \"" + country + "\"").append(System.lineSeparator());
19    query.append("}").append(System.lineSeparator());
20
21    QueryExecution queryExecution = QueryExecutionFactory.create(query.toString(), dataset);
22    ResultSet resultSet = queryExecution.execSelect();
23    return convertResultSetToJsonString(resultSet);
24 }

```

We implemented the code to append country code in the economic data set.

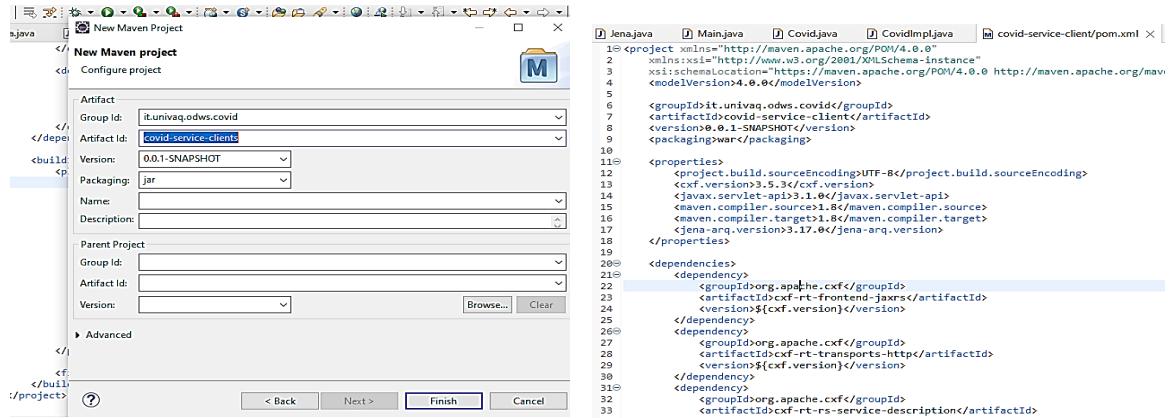
```

] CovidNew.java | ] CovidNewImpl.java × |
125     QueryExecution queryExecution = QueryExecutionFactory.create(query.toString(), dataset);
126     ResultSet resultSet = queryExecution.execSelect();
127     return convertResultSetToJsonString(resultSet);
128 }
129
130 @Override
131 public String getCovidCaseByCountryCode(String countryCode) {
132     Dataset dataset = loadDataset();
133
134     StringBuilder query = new StringBuilder();
135     query.append("PREFIX dbp: <http://dbpedia.org/ontology/>").append(System.lineSeparator());
136     query.append("PREFIX sch: <https://schema.org/>").append(System.lineSeparator());
137     query.append("SELECT ?country ?countryCode ?year ?gdppercen ?indexprice ?stockindex ?sameasstockindex ?inflationrate ").append(System.lineSeparator());
138     query.append("WHERE {}").append(System.lineSeparator());
139     query.append(" ?covid dbp:country ?country").append(System.lineSeparator());
140     query.append(" ?covidnew dbp:countryCode ?countryCode").append(System.lineSeparator());
141     query.append(" ?covid dbp:stockindex ?stockindex").append(System.lineSeparator());
142     query.append(" ?covid dbp:sameasstockindex ?sameasstockindex").append(System.lineSeparator());
143     query.append(" ?covid dbp:year ?year").append(System.lineSeparator());
144     query.append(" ?covid dbp:gdppercen ?gdppercen").append(System.lineSeparator());
145     query.append(" ?covid dbp:indexprice ?indexprice").append(System.lineSeparator());
146     query.append(" ?covid dbp:inflationrate ?inflationrate").append(System.lineSeparator());
147     query.append(" FILTER((LCASE(?countryCode) = \"" + countryCode.toLowerCase() + "\"))").append(System.lineSeparator());
148     query.append("}").append(System.lineSeparator());
149
150     QueryExecution queryExecution = QueryExecutionFactory.create(query.toString(), dataset);
151     ResultSet resultSet = queryExecution.execSelect();
152     return convertResultSetToJsonString(resultSet);
153 }

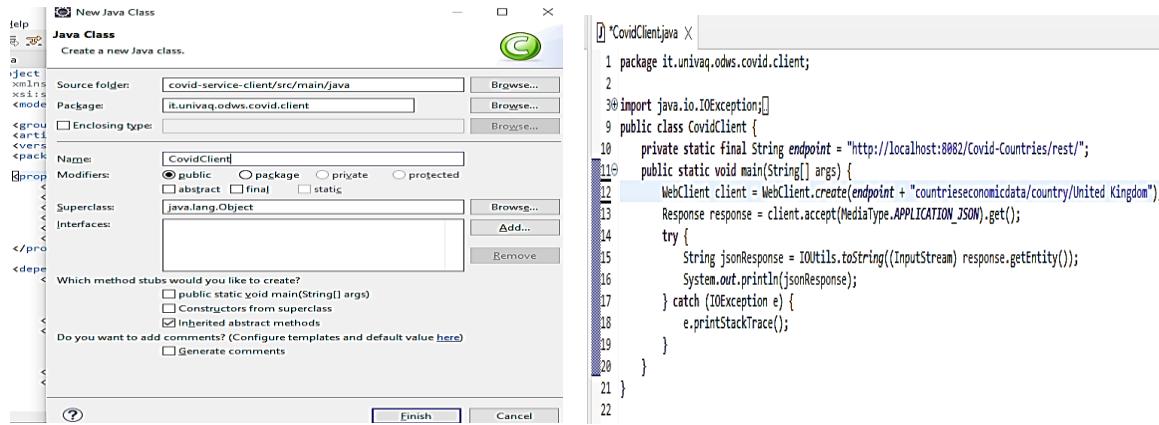
```

3.8 Java Client

We created a java client project, and edited the java client POM file.



We then created the java client class and implemented it.



4. Results

We got the following result from each individual dataset:

Countries by Government Budget Per Capita at index 1

```
{ "head": {  
    "vars": [ "country" , "sameas" , "budgetpercapita" , "year" , "imdb_url" , "wikidata_url" ]  
} ,  
"results": {  
    "bindings": [  
        {  
            "country": { "type": "literal" , "value": "France" } ,  
            "sameas": { "type": "literal" , "value": "https://wikidata.org/wiki/Q142" } ,  
            "budgetpercapita": { "type": "literal" , "datatype": "http://www.w3.org/2001/XMLSchema#int" , "value": "20960" } ,  
            "year": { "type": "literal" , "datatype": "http://www.w3.org/2001/XMLSchema#dateTime" , "value": "2017-01-01T00:00Z" }  
        }  
    ]  
}
```

Covid 19 Impact on Global Economy at index 2350

```
{ "head": {  
    "vars": [ "countryCode" , "country" , "sameascountry" , "date" , "humanDevelopmentIndex" , "tradediversion" , "gdpPerCapita" , "imdb_url" , "wikidata_url" ]  
} ,  
"results": {  
    "bindings": [  
        {  
            "countryCode": { "type": "literal" , "value": "USA" } ,  
            "country": { "type": "literal" , "value": "United States of America" } ,  
            "date": { "type": "literal" , "datatype": "http://www.w3.org/2001/XMLSchema#dateTime" , "value": "2020-10-19T00:00Z" } ,  
            "humanDevelopmentIndex": { "type": "literal" , "datatype": "http://www.w3.org/2001/XMLSchema#double" , "value": "0.924" } ,  
            "tradediversion": { "type": "literal" , "datatype": "http://www.w3.org/2001/XMLSchema#double" , "value": "12.2998991" } ,  
            "gdpPerCapita": { "type": "literal" , "datatype": "http://www.w3.org/2001/XMLSchema#double" , "value": "10.90090556" }  
        }  
    ]  
}
```

Covid 19 case Reported in 2020 at index 0

```
{ "head": {  
    "vars": [ "country" , "sameascountry" , "continent" , "population" , "totalcases" , "totaldeaths" , "criticalcases" , "casespermillionpeople" , "deathspermillionpeople" , "totaltests" ,  
    "testspermillionpeople" , "whoregion" , "imdb_url" , "wikidata_url" ]  
} ,  
"results": {  
    "bindings": [  
        {  
            "country": { "type": "literal" , "value": "United State of America" } ,  
            "continent": { "type": "literal" , "value": "North America" } ,  
            "population": { "type": "literal" , "datatype": "http://www.w3.org/2001/XMLSchema#int" , "value": "331198130" } ,  
            "totalcases": { "type": "literal" , "datatype": "http://www.w3.org/2001/XMLSchema#int" , "value": "5032179" } ,  
            "totaldeaths": { "type": "literal" , "datatype": "http://www.w3.org/2001/XMLSchema#int" , "value": "162804" } ,  
            "criticalcases": { "type": "literal" , "datatype": "http://www.w3.org/2001/XMLSchema#int" , "value": "18296" } ,  
            "casespermillionpeople": { "type": "literal" , "datatype": "http://www.w3.org/2001/XMLSchema#int" , "value": "15194" } ,  
            "deathspermillionpeople": { "type": "literal" , "datatype": "http://www.w3.org/2001/XMLSchema#int" , "value": "492" } ,  
            "whoregion": { "type": "literal" , "value": "Americas" }  
        }  
    ]  
}
```

Economics Data for-9 Countries by country Name (United States of America)

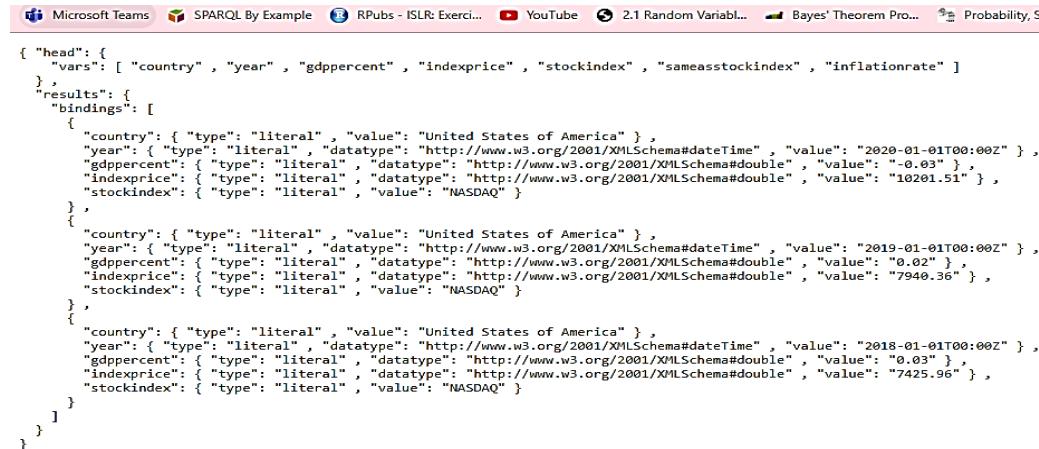
```
{ "head": {  
    "vars": [ "country" , "year" , "gdppercents" , "indexprice" , "stockindex" , "sameassstockindex" , "inflationrate" ]  
} ,  
"results": {  
    "bindings": [  
        {  
            "country": { "type": "literal" , "value": "United States of America" } ,  
            "year": { "type": "literal" , "datatype": "http://www.w3.org/2001/XMLSchema#dateTime" , "value": "2020-01-01T00:00Z" } ,  
            "gdppercents": { "type": "literal" , "datatype": "http://www.w3.org/2001/XMLSchema#double" , "value": "-0.03" } ,  
            "indexprice": { "type": "literal" , "datatype": "http://www.w3.org/2001/XMLSchema#double" , "value": "10201.51" } ,  
            "stockindex": { "type": "literal" , "value": "NASDAQ" } ,  
            "sameassstockindex": { "type": "literal" , "value": "https://www.wikidata.org/wiki/Q575782" } ,  
            "inflationrate": { "type": "literal" , "datatype": "http://www.w3.org/2001/XMLSchema#double" , "value": "0.01" }  
        } ,  
        {  
            "country": { "type": "literal" , "value": "United States of America" } ,  
            "year": { "type": "literal" , "datatype": "http://www.w3.org/2001/XMLSchema#dateTime" , "value": "2019-01-01T00:00Z" } ,  
            "gdppercents": { "type": "literal" , "datatype": "http://www.w3.org/2001/XMLSchema#double" , "value": "0.02" } ,  
            "indexprice": { "type": "literal" , "datatype": "http://www.w3.org/2001/XMLSchema#double" , "value": "7946.36" } ,  
            "stockindex": { "type": "literal" , "value": "NASDAQ" } ,  
            "sameassstockindex": { "type": "literal" , "value": "https://www.wikidata.org/wiki/Q575782" } ,  
            "inflationrate": { "type": "literal" , "datatype": "http://www.w3.org/2001/XMLSchema#double" , "value": "0.02" }  
        } ,  
        {  
            "country": { "type": "literal" , "value": "United States of America" } ,  
            "year": { "type": "literal" , "datatype": "http://www.w3.org/2001/XMLSchema#dateTime" , "value": "2018-01-01T00:00Z" } ,  
            "gdppercents": { "type": "literal" , "datatype": "http://www.w3.org/2001/XMLSchema#double" , "value": "0.03" } ,  
            "indexprice": { "type": "literal" , "datatype": "http://www.w3.org/2001/XMLSchema#double" , "value": "7425.96" } ,  
            "stockindex": { "type": "literal" , "value": "NASDAQ" } ,  
            "sameassstockindex": { "type": "literal" , "value": "https://www.wikidata.org/wiki/Q575782" } ,  
            "inflationrate": { "type": "literal" , "datatype": "http://www.w3.org/2001/XMLSchema#double" , "value": "0.02" }  
        }  
    ]  
}
```

Covid 19 Impact on Global Economy and Economics Data for 9 Countries

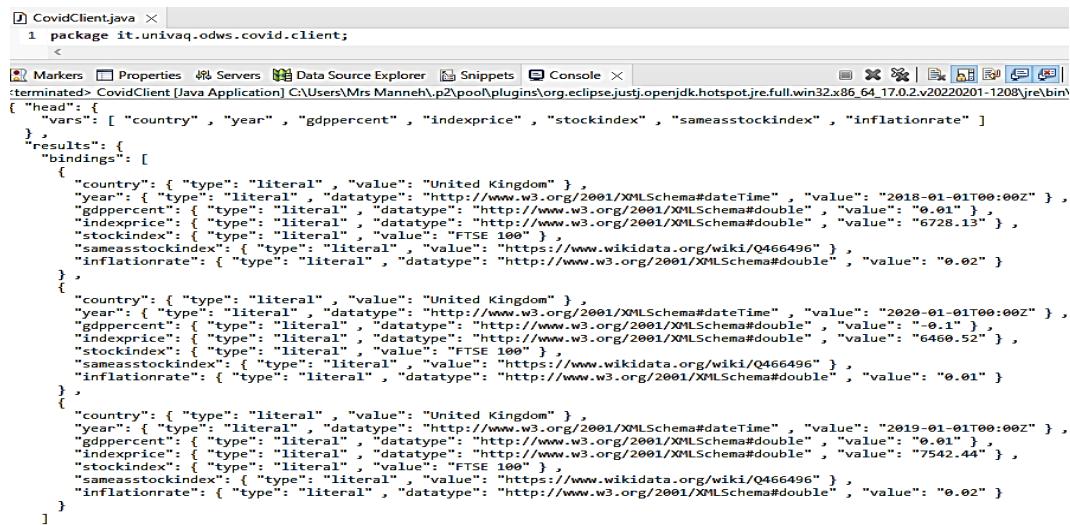
Query result from combining two RDF files: we added country code to the Economics Data.

```
{
  "head": {
    "vars": [ "country", "countryCode", "year", "gdppercents", "indexprice", "stockindex", "sameasstockindex", "inflationrate" ]
  },
  "results": {
    "bindings": [
      {
        "country": { "type": "literal", "value": "Japan" },
        "countryCode": { "type": "literal", "value": "CHN" },
        "year": { "type": "literal", "datatype": "http://www.w3.org/2001/XMLSchema#dateTime", "value": "2019-01-01T00:00Z" },
        "gdppercents": { "type": "literal", "datatype": "http://www.w3.org/2001/XMLSchema#double", "value": "0" },
        "indexprice": { "type": "literal", "datatype": "http://www.w3.org/2001/XMLSchema#double", "value": "21697.23" },
        "stockindex": { "type": "literal", "value": "Nikkei 225" },
        "sameasstockindex": { "type": "literal", "value": "https://www.wikidata.org/wiki/Q507338" },
        "inflationrate": { "type": "literal", "datatype": "http://www.w3.org/2001/XMLSchema#double", "value": "0" }
      },
      {
        "country": { "type": "literal", "value": "Japan" },
        "countryCode": { "type": "literal", "value": "CHN" },
        "year": { "type": "literal", "datatype": "http://www.w3.org/2001/XMLSchema#dateTime", "value": "2019-01-01T00:00Z" },
        "gdppercents": { "type": "literal", "datatype": "http://www.w3.org/2001/XMLSchema#double", "value": "0" },
        "indexprice": { "type": "literal", "datatype": "http://www.w3.org/2001/XMLSchema#double", "value": "21697.23" },
        "stockindex": { "type": "literal", "value": "Nikkei 225" },
        "sameasstockindex": { "type": "literal", "value": "https://www.wikidata.org/wiki/Q507338" },
        "inflationrate": { "type": "literal", "datatype": "http://www.w3.org/2001/XMLSchema#double", "value": "0" }
      },
      {
        "country": { "type": "literal", "value": "Japan" },
        "countryCode": { "type": "literal", "value": "CHN" },
        "year": { "type": "literal", "datatype": "http://www.w3.org/2001/XMLSchema#dateTime", "value": "2019-01-01T00:00Z" },
        "gdppercents": { "type": "literal", "datatype": "http://www.w3.org/2001/XMLSchema#double", "value": "0" },
        "indexprice": { "type": "literal", "datatype": "http://www.w3.org/2001/XMLSchema#double", "value": "21697.23" },
        "stockindex": { "type": "literal", "value": "Nikkei 225" },
        "sameasstockindex": { "type": "literal", "value": "https://www.wikidata.org/wiki/Q507338" },
        "inflationrate": { "type": "literal", "datatype": "http://www.w3.org/2001/XMLSchema#double", "value": "0" }
      },
      {
        "country": { "type": "literal", "value": "Japan" },
        "countryCode": { "type": "literal", "value": "CHN" },
        "year": { "type": "literal", "datatype": "http://www.w3.org/2001/XMLSchema#dateTime", "value": "2019-01-01T00:00Z" },
        "gdppercents": { "type": "literal", "datatype": "http://www.w3.org/2001/XMLSchema#double", "value": "0" },
        "indexprice": { "type": "literal", "datatype": "http://www.w3.org/2001/XMLSchema#double", "value": "21697.23" },
        "stockindex": { "type": "literal", "value": "Nikkei 225" },
        "sameasstockindex": { "type": "literal", "value": "https://www.wikidata.org/wiki/Q507338" },
        "inflationrate": { "type": "literal", "datatype": "http://www.w3.org/2001/XMLSchema#double", "value": "0" }
      }
    ]
  }
}
```

Result from filtering two RDF files



This is the result we got from the java client



5. Conclusion

We successfully implemented the REST web service in maven as well as the java client and we were able to retrieved some relevant information from the dataset.

6. Appendix

Dataset Descriptions, and their licenses.

- Covid19 Impact on Global Economy: This dataset has GDP and covid 19 cases information from many countries. It presents estimated economy-wide and sector-specific impacts of the COVID-19 outbreak. It updates the estimates released on March 28, 2020. Some of its indicators includes, Human Development Index (HDI), GDP per Capita (GDPCAP) and Trade Diversion (TD).
 - License: <https://data.mendeley.com/datasets/b2wvnbnpj9/1>
- List of countries by government budget per capita: This list shows the government budget of each country divided by its total population, not adjusted to purchasing power parity. The Chinese, Brazilian, Indian, and United States government budgets used are the figures reported by the International Monetary Fund.
 - License: <https://creativecommons.org/publicdomain/zero/1.0/>
- Economic Data – 9 Countries: The dataset contains data for 8 countries and one special administrative region (China, France, Germany, Hong Kong, India, Japan, Spain, United Kingdom and United States of America) from 1980 through 2020. It include major macroeconomic factors like inflation, unemployment, GDP, exchange rate (base USD) and per capita income. Apart from that it has the stock prices of the respective country's major stock index which can help in analysing the data set to identify the impact of major macroeconomic variables on the movement of stock index prices.
 - License: <https://creativecommons.org/licenses/by-nc-sa/4.0/>
- COVID-19 Dataset: A new coronavirus designated 2019-covid was first identified in Wuhan, the capital of China's Hubei province. People developed pneumonia without a clear cause and for which existing vaccines or treatments were not effective. The virus has shown evidence of human-to-human transmission. Transmission rate (rate of infection) appeared to escalate in mid-January 2020. As of 30 January 2020, approximately 8,243 cases. This dataset contains the number of Confirmed, Death and Recovered cases.
 - License: <https://github.com/CSSEGISandData/COVID-19>
- COVID-19 Public Health Social Measures: A global database of public health and social measures 2021. Public health and social measures

(PHSMs) are measures or actions by individuals, institutions, communities, local and national governments and international bodies to slow or stop the spread of an infectious disease, such as COVID-19. Since the start of the COVID-19 pandemic, a number of organizations have begun tracking implementation of PHSMs around the world, using different data collection methods, database designs and classification schemes.

- License: <https://www.who.int/about/policies/publishing/copyright>
- License: <https://creativecommons.org/licenses/by-nc-sa/3.0/igo/>

7. Reference

- <https://www.kaggle.com/datasets/shruttisaxena/covid-19-impact-on-global-economy>
- <https://www.kaggle.com/datasets/mathurinache/list-of-countries-by-government-budget-per-capita>
- <https://www.kaggle.com/datasets/pratik453609/economic-data-9-countries-19802020>
- <https://www.kaggle.com/datasets/imdevskp/corona-virus-report>
- <https://www.kaggle.com/datasets/hijest/covid19-public-health-social-measures>

8. Task

S/N	Task	Michael	Lamin	Jai
1	Dataset Gathering	100%	-	-
2	Data cleaning on OpenRefine,	100%	-	-
3	Dataset enrichment, linkage, RDF Extraction	100%	-	-
4	SPARQL Queries	10%	10%	80%
5	Rest Web Service	-	100%	-
6	Java Client	-	100%	-
7	Report	10%	-	90%