Introduction

The entities used for this coursework are Country, City, Politics, Language, Continent, Airport, Economy, and Ethnic Group from the Mondial Relational Database[1].
Task 1 was completed which was focused on writing and executing the exploratory and retrieval queries. Task 2 which includes the report comprising the populated Mondial entity, the timesheet, and comments for the queries along with the discussion on effectiveness and efficiency can be found below.

Entity 1: Country

Mondial Columns: name, code, capital, province, area, population.
Primary key: code (refers to the country code i.e. for India, the code will be 'IND')

Exploratory Query:

```
SELECT DISTINCT ?C WHERE {?C a dbo:Country}
```

TimeSheet for Exploratory:

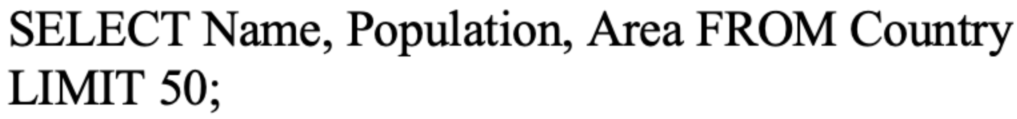
TASK	TIME SPENT	COMMENTS
Analysed the results returned by the exploratory query.	2 hour	- Found results containing unexpected entries such as cities (eg. Calcutta), cricket teams (eg. Canada National Cricket Team)
Studied the predicates to correlate with Mondial columns.	3 hours	- Spent a lot of time to find out all the predicates present in the Mondial table. - Wasn't able to find some of them. - For instance, dbo:countrycode predicate refers to the international calling code i.e. +44 for the UK in contrast with GB in the Mondial Database.
Investigated the reason for unrelated results.	2 hours	- Found out that the rdf:type for these results contains dbo:Country leading to inconsistent results on querying. - Discovered some interesting ambiguity in the data such as overlapping categories where dbo:location and dbo:region essentially refer to the same thing.

Explored alternative query approaches	1 hour	- Experimented with alternative SPARQL query structures to capture data more accurately. - This approach helped in understanding different ways to navigate DBPedia.

Retrieval Query:

```
SELECT ?countryName (SAMPLE(?area) AS ?uniqueArea)  (SAMPLE(STR(?population)) AS
?uniquePopulation)
WHERE {
   ?country a dbo:Country;
      rdfs:label ?countryName.
   ?country dbo:countryCode ?code .
   ?country dbo:area ?area .
   ?country dbo:populationTotal ?population .
FILTER (lang(?countryName) = "en")
}
GROUP BY ?countryName
ORDER BY ASC(?countryName)
LIMIT 50
```

Equivalent Mondial SQLite Query:



```
SELECT Name, Population, Area FROM Country
LIMIT 50;
```

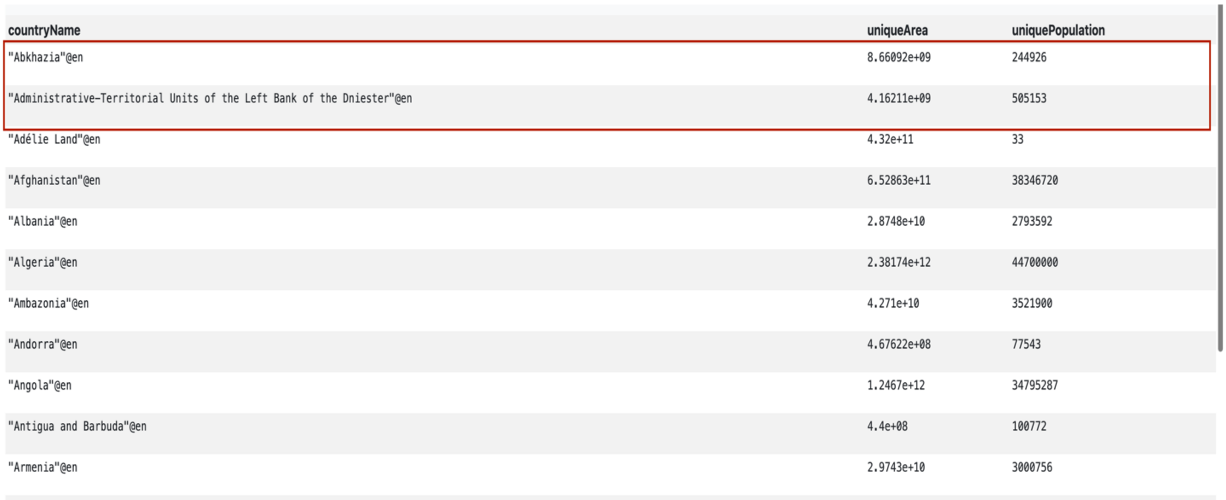
TimeSheet for Retrieval:

TASK	TIME SPENT	COMMENTS
Identified unique predicates for relevant results	2 hours	- Realised that the dbo:country objects that are actually "countries" in the Mondial database contain properties of dbo:currencyCode, dbo:countryCode which is absent in other results. - Identifying these took a lot of time.
Determined the most effective query version	2 hours	- Tried and tested different queries to understand which one leads to the most identical outcome compared to the SQLite query results.

		- DBPedia gave a *503 Service Temporarily Unavailable error* while working on this. - Use SAMPLE to ensure countries where dbo:area contains 2 values, one in xsd:decimal and other in xsd:float format, only one value is returned in the final output.

Comments on Effectiveness:

The Mondial database yields 244 entries for the given query, while the SPARQL retrieval query returns only 202 entries—a shortfall of 17%. The exploratory query for countries generated 14,694 rows, many of which were irrelevant to the country table in Mondial. This highlights the challenges posed by the absence of a schema in DBpedia, which becomes evident when only 202 countries are retrieved. Consequently, any country lacking area or population data was excluded from the results. Making the *?area and ?population* fields OPTIONAL increased the results to 305 entries. However, this count included duplicates and entries not pertinent to countries. Thus, the decision to exclude the OPTIONAL clause was made to prioritize data quality over the sheer number of outputs. Despite this, some undesirable values are present in the output which is shown in Fig. 1, however, this has been limited to make the output as high quality as possible.



countryName	uniqueArea	uniquePopulation
"Abkhazia"@en	8.66092e+09	244926
"Administrative-Territorial Units of the Left Bank of the Dniester"@en	4.16211e+09	505153
"Adélie Land"@en	4.32e+11	33
"Afghanistan"@en	6.52863e+11	38346720
"Albania"@en	2.8748e+10	2793592
"Algeria"@en	2.38174e+12	44700000
"Ambazonia"@en	4.271e+10	3521900
"Andorra"@en	4.67622e+08	77543
"Angola"@en	1.2467e+12	34795287
"Antigua and Barbuda"@en	4.4e+08	100772
"Armenia"@en	2.9743e+10	3000756

Fig 1. Highlighted undesirable output of Retrieval Query

Comments on Efficiency:

Exploring DBpedia's content for countries with SPARQL was a time-intensive process, predominantly due to the intricacies of DBpedia's data. Understanding and analyzing the presence or absence of predicates and exploring the scope for finding suitable replacements for objects or thinking about ways to compute the missing values took significant effort, which could have been easier if there was a schema to refer to as documentation, as well as a schema to constrain the RDF graphs of DBpedia. I spent some time wondering if, for countries where the population is missing (and hence it is not shown in the final output), I could use the property of `dbo:populationDensity` and multiply it with the total area to find the population. However, this approach failed since even cities or football teams-related data containing both these properties were being shown in the final output, which was not ideal. Tools like LodLive provided some assistance but did not substantially expedite the exploration process.

Entity 2: City

Mondial columns: Name, Country, Province, Population, Latitude, Longitude, and Elevation.
Primary Key: (*Name, Country, and Province*)

Exploratory Query:

```
select DISTINCT ?city ?name
WHERE{
?city a dbo:City.
?city rdfs:label ?name.
FILTER (LANG(?name)="en")
}
```

TimeSheet for Exploratory:

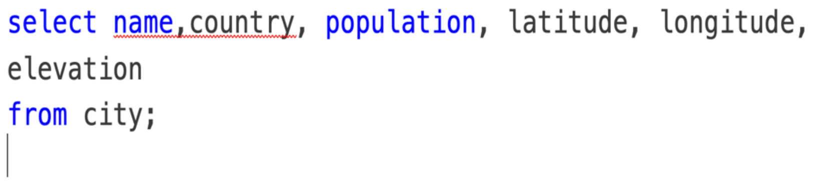
TASK	TIME SPENT	COMMENTS
Studied the results returned by the exploratory query	2 hours	- Noticed that there are almost 3x the number of results than present in the Mondial. - Had to spend a lot of time understanding which entries are present apart from actual cities of the world.
Searched for predicates that give information about the population, province, latitude, longitude, and elevation	1 hour	- Found out that most of the information was present but had different labels. - Identifying how these labels relate to the data I'm searching for took some time.
Compared property values with Mondial data	1.5 hours	- Realised not all values match completely. - For instance, the population of Mumbai in Mondial is 12.44 million whereas in DBpedia it is 12.47 million.

Retrieval Query:

```
SELECT DISTINCT ?Name ?country
```

```
             (STR(?population) AS ?population)
            (xsd:decimal(REPLACE(?coordinates, " .*", "")) AS ?latitude)
            (xsd:decimal(REPLACE(?coordinates, ".* ", "")) AS ?longitude)
            ?elevation
WHERE {
   ?city rdf:type dbo:City ;
        rdfs:label ?Name ;
        dbo:country ?countryName ;
         rdfs:label ?country;
        dbo:populationTotal ?population .
   OPTIONAL { ?city georss:point ?coordinates }
   OPTIONAL { ?city dbo:elevation ?elevation }
   FILTER (lang(?Name) = "en")
   FILTER (lang(?country) = "en")
}
GROUP BY ?city ?country?population ?coordinates ?elevation
```

Equivalent Mondial SQLite Query:



```
select name,country, population, latitude, longitude,
elevation
from city;
```

TimeSheet for Retrieval:

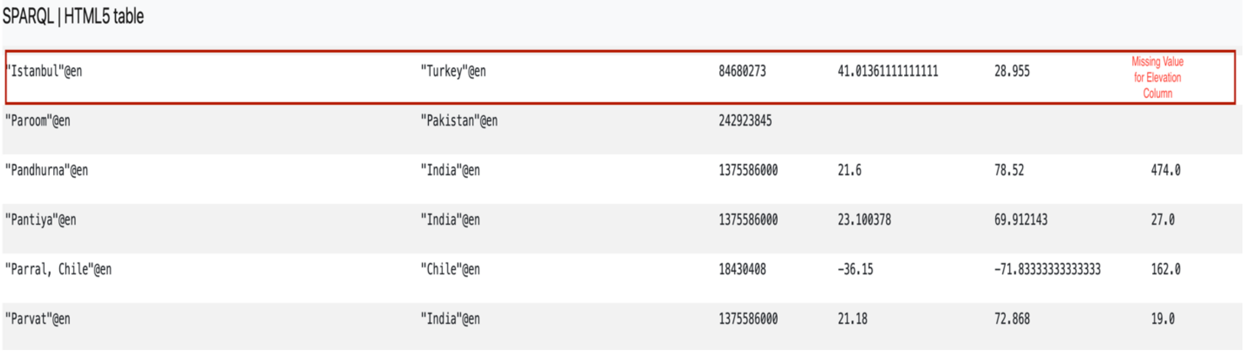
TASK	TIME SPENT	COMMENTS
Considered filtering techniques	1 hour	- Noticed that few cities have names in a local language, however, all have names in English. - Decided to apply the filter of English on City name and Country.
Explored SPARQL syntax for geolocation data.	30 mins	- Figured out some cities had missing values for latitude and/or longitude which led to additional filtering in the query results. - Added the `OPTIONAL` field to georss:point to include records with missing values in the output.
Investigated properties for the Province column	1 hour	- Found a dbo: subdivision property which contained additional information along with the Province information that was necessary.

Comments on Effectiveness:

The output includes around 15,731 entries, in stark contrast with the 3,350 records present in the Mondial. This can be attributed to factors such as the wide variety of data present in the DBpedia database compared to the Mondial database. It is important to consider that, despite applying strict filters on language and using properties to further refine the results, the numbers are still substantial. It can be interpreted that the data present in DBpedia is far more comprehensive, leading to additional results despite eliminating duplicates and redundant occurrences in the query. I included the dbo:subdivision predicate in the query since it gave results similar to Province but later omitted it since it gave additional values along with the Province. For instance, if the Province for Mumbai was “Maharashtra,” it gave “Mumbai City District, Maharashtra, Konkan Division,” which is not accurate.

Comments on Efficiency:

The retrieval process proved highly efficient, capturing 6 out of 7 desired dbo:City details, aligning with Mondial's structure. Nonetheless, the adaptation of DBpedia's values to Mondial's framework was time-consuming. The output was extensive and demanded discerning filtration to enhance result quality. Making georss:point and dbo:elevation optional was essential, as key cities like Istanbul lacked elevation data.



SPARQL | HTML5 table

"Istanbul"@en	"Turkey"@en	84680273	41.01361111111111	28.955	Missing Value for Elevation Column
"Paroom"@en	"Pakistan"@en	242923845			
"Pandhurna"@en	"India"@en	1375586000	21.6	78.52	474.0
"Pantiya"@en	"India"@en	1375586000	23.100378	69.912143	27.0
"Parral, Chile"@en	"Chile"@en	18430408	-36.15	-71.83333333333333	162.0
"Parvat"@en	"India"@en	1375586000	21.18	72.868	19.0

Fig. 2 Proof of Missing value for “Istanbul”

Entity 3: Politics

The Politics entity consists of columns Country, Independence, WasDependent, Dependent, and Government. Country is the primary key. For example, the entry for India would be uniquely identified by the country code 'IND' and include data on its independence and form of government. ?iso31661Code

Exploratory Query:

```
SELECT ?country ?governmentType
WHERE {
   ?country a dbo:Country .
    ?country dbo:governmentType ?governmentType
}
```

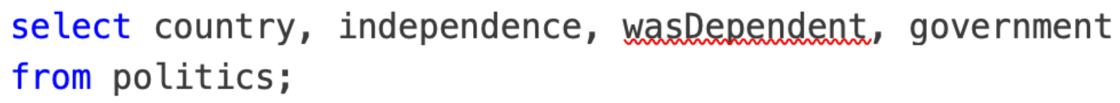
TimeSheet for Exploratory:

TASK	TIME SPENT	COMMENTS
Identify politics-related information/predicates from **dbo:country**.	1 hour	- Was unable to find out information directly for date of independence. - GovernmentType property had multiple values for a single country.
Research equivalency of ISO31661Code and Mondial's Countrycode.	2 hours	- Established date is very similar to independence date which can be used. However, some countries have multiple dates so the correct date must be selected. - Country Code is not present however dbo:iso31661Code seems very similar to it.

Retrieval Query:

```
SELECT ?countryName
     (GROUP_CONCAT(DISTINCT ?governmentType; separator = ", ") AS ?governmentTypes)
     ?Code
     (SAMPLE(?establishedDateSample) AS ?establishedDate)
     ?sovereigntyNote
WHERE {
  ?country a dbo:Country .
  ?country rdfs:label ?countryName .
  ?country dbo:countryCode ?code.
  OPTIONAL { ?country dbo:iso31661Code ?Code . }
  OPTIONAL { ?country dbp:establishedDate ?establishedDateSample . }
  OPTIONAL { ?country dbp:sovereigntyNote ?sovereigntyNote . }
  OPTIONAL {
     ?country dbo:governmentType ?G .
     ?G rdfs:label ?governmentType .
     FILTER (lang(?governmentType) = 'en')
  }
  FILTER (LANG(?countryName) = "en")
}
GROUP BY ?countryName ?Code ?sovereigntyNote
```

Equivalent Mondial SQLite Query:



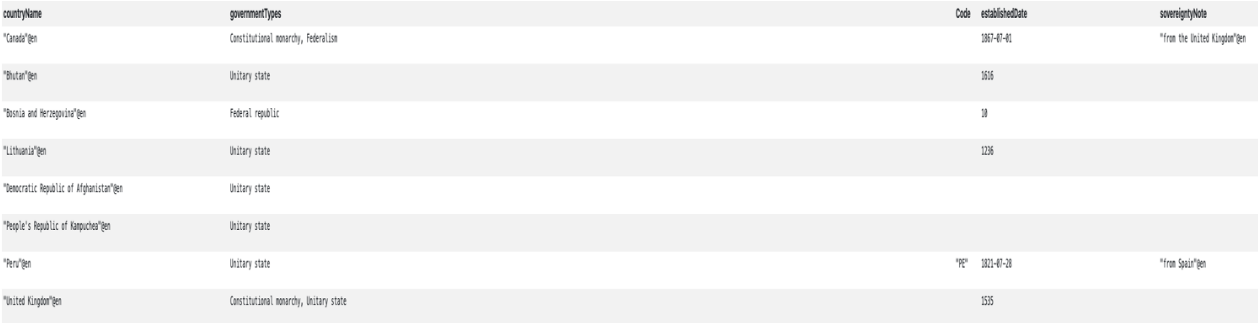
```
select country, independence, wasDependent, government
from politics;
```

TimeSheet for Retrieval:

TASK	TIME SPENT	COMMENTS
Study the syntax of SPARQL for effective retrieval.	1.5 hours	- Went through the documentation to understand how to concatenate various governmentTypes into one for a country. - Studied the sample function to know how to give the correct output for establishedDate (independence date) for the output.
Research if ISO31661Code and Countrycode in Mondial are the same.	1 hour	- dbo:iso31661Code likely refers to the ISO 3166-1 Alpha-2 code set by International Organization for Standardization (ISO). - Couldn’t find what exactly Code in Mondial refers to.

Comments on Effectiveness:

The retrieval query returns 305 records, which is close to the 244 records in the Mondial database. However, not all these records contain all the fields present in the Politics entity. For instance, for the country code, the ISO31661Code has been used, which is missing for a lot of countries, such as the United Kingdom. SovereigntyNote, similar to the wasDependent column in Mondial, contains a sentence instead of just the country's name. I have still kept this column since it preserves the integrity of the results by providing the same information, albeit in a different format. The countryName column also had to be included since Code is missing for a lot of countries making it difficult to identify the record.



countryName	governmentTypes	Code	establishedDate	sovereigntyNote
"Canada"@en	Constitutional monarchy, Federalism		1867-07-01	"from the United Kingdom"@en
"Bhutan"@en	Unitary state		1616	
"Bosnia and Herzegovina"@en	Federal republic		10	
"Lithuania"@en	Unitary state		1236	
"Democratic Republic of Afghanistan"@en	Unitary state			
"People's Republic of Kampuchea"@en	Unitary state			
"Peru"@en	Unitary state	"PE"	1821-07-28	"from Spain"@en
"United Kingdom"@en	Constitutional monarchy, Unitary state		1535	

Fig. 3 Output of the retrieval query

Comments on Efficiency:

Adapting to the property naming discrepancies between databases posed challenges, requiring substantial time to decipher data formats, particularly for **?governmentType**. Clarifying the semantics of results, such as equating 'Democracy' in Mondial with 'Federalism' in SPARQL, demanded additional research[2] . Prior familiarity with the country entity facilitated the task, leading to anticipated outcomes from the query execution.

Entity 4: Language

Mondial Columns: Country, Name, and Percentage.
Primary key: (Name, Country)

Exploratory Query:

```
SELECT ?language ?lang
WHERE {
  ?language rdf:type dbo:Language .
?language rdfs:label ?lang.
FILTER( LANG(?lang)="en")

}
```

TimeSheet for Exploratory:

TASK	TIME SPENT	COMMENTS
Execute the query and find relevant predicates	2 hours	- Could not find data about speakers in percentage like in Mondial. - There are different records for the same language, for instance American English Speakers and Indian English speakers have each been counted separately.

Analyse techniques to transform data into Mondial form	2 hours	- Unable to aggregate and add the English speakers of different countries into one common record. - Spent a lot of time trying to understand how to achieve it in Sparql through resources such as stackoverflow.

Retrieval Query:

Aim: Find the top 10 languages spoken in the world.

```
SELECT ?lang(SAMPLE(?speakers) AS ?sampledSpeakers)
WHERE {
  ?language rdf:type dbo:Language .
?language rdfs:label ?lang.
  OPTIONAL { ?language dbp:speakers ?speakers }
FILTER( LANG(?lang)="en")
  FILTER (datatype(?speakers) = xsd:integer)

}
ORDER BY DESC (?sampledSpeakers)
```

Equivalent Mondial SQLite Query:

```
SELECT l.Name, SUM(c.Population * l.Percentage / 100) AS
TotalSpeakers
FROM Language l
JOIN Country c ON l.Country = c.Code
GROUP BY l.Name
ORDER BY TotalSpeakers DESC
LIMIT 10;
```

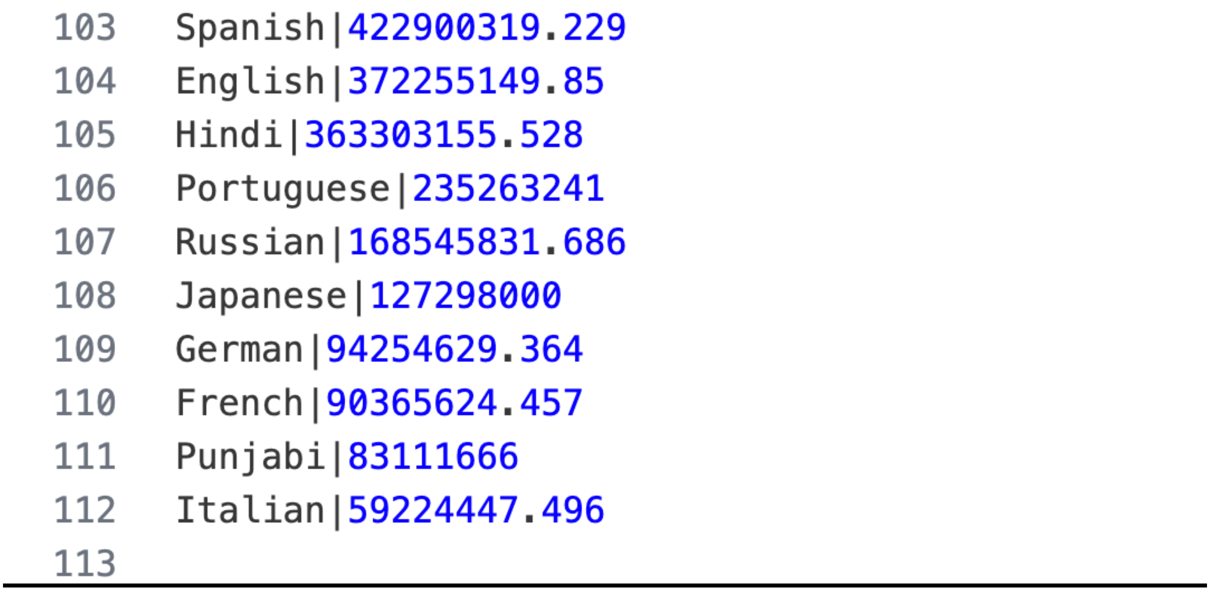
TimeSheet for Retrieval:

TASK	TIME SPENT	COMMENTS
Analyse format of the data	2 hour	- Some data for dbo:speakers property was present in datatype xsd:integer while in some instances there was a string present such as “almost 1

		million L2 speakers of the language” - Filtered the speakers to only include integer value to match Mondial records. - Since some records contained multiple values for the number of speakers in integer as well, I used SAMPLE to select one value since there is no evidence of which value is recent/correct. SAMPLE helps arbitrarily select a value [3].
Check for presence of duplicates	30 minutes	- Found duplicates which were caused by the same values present in multiple languages. - Applied a FILTER on languages to handle this issue.

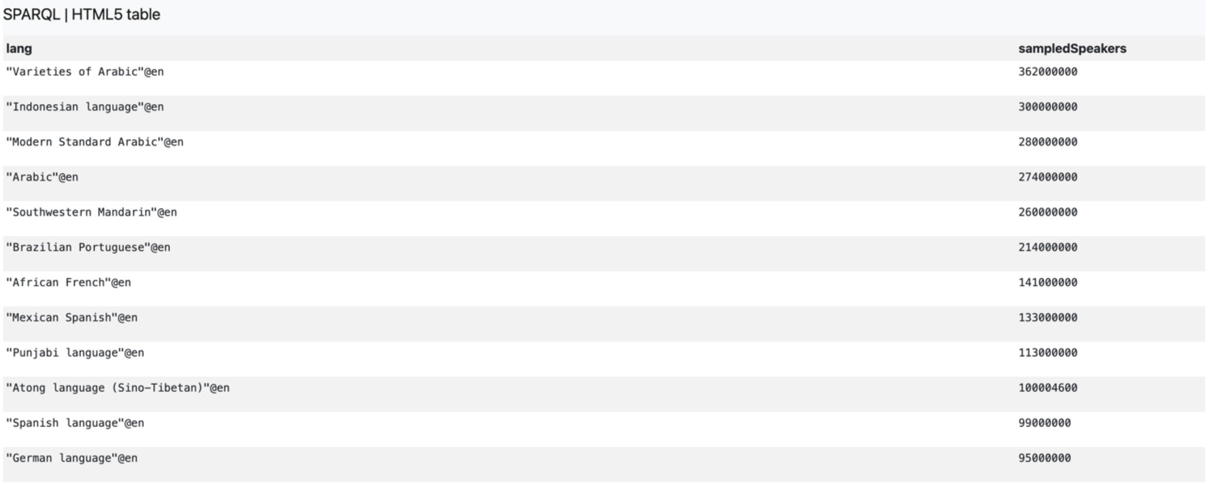
Comments on Effectiveness:

Comparing Figures 4 and 5, which represent outputs from Mondial and DBpedia respectively, it's clear that the task did not mirror the Mondial results precisely. The primary issue was the inability to aggregate the speakers of languages from various regions. For example, while French ranks in the top 10 in Mondial, DBpedia lists it as African French due to the lack of a technique in SPARQL for summation across regions. The presence of various Arabic dialects in DBpedia, which Mondial treats separately, further complicates a direct comparison. The flexible nature of RDF allows for nuanced data representation, a trade-off that should be recognized when considering data integrity.



```
Spanish|422900319.229
English|372255149.85
Hindi|363303155.528
Portuguese|235263241
Russian|168545831.686
Japanese|127298000
German|94254629.364
French|90365624.457
Punjabi|83111666
Italian|59224447.496
```

Fig 4. Output of Top 10 languages spoken using Mondial database



SPARQL | HTML5 table

lang	sampledSpeakers
"Varieties of Arabic"@en	362000000
"Indonesian language"@en	300000000
"Modern Standard Arabic"@en	280000000
"Arabic"@en	274000000
"Southwestern Mandarin"@en	260000000
"Brazilian Portuguese"@en	214000000
"African French"@en	141000000
"Mexican Spanish"@en	133000000
"Punjabi language"@en	113000000
"Atong language (Sino-Tibetan)"@en	100004600
"Spanish language"@en	99000000
"German language"@en	95000000

Fig 5. Output of Top 10 languages spoken using DBpedia

Comments on Efficiency:

This task was more efficient compared to previous entities since only 2 columns (speakers and percentage) had to be searched during the exploratory phase. Hence, the absence of a schema was not immediately felt. However, processing data through the query and choosing a suitable datatype for output required some thought which could've been simpler if a schema language restricted the usage of datatypes for the dbo:speakers property.

Entity 5: Continent

Mondial Columns: Name and Area.
Primary Key: Name

Exploratory Query:

```
SELECT DISTINCT ?Continent
WHERE {
?Continent a umbel-rc:Continent.
}
```

TimeSheet for Exploratory:

TASK	TIME SPENT	COMMENTS
Identify relevant predicates for the **umbel-rc:Continent** entity.	2 hour	- Found a property, dbo:areaTotal, present in 5 out of 7 records. - Noticed that the area was in square meters, as opposed to square kilometers in Mondial.

Retrieval Query:

```
SELECT DISTINCT ?Continent, STR((?Area/ 1000000)) AS ?AREA
WHERE {?C a umbel-rc:Continent.
?C rdfs:label ?Continent.
FILTER(lang(?Continent)="en")
OPTIONAL{?C dbo:areaTotal ?Area.}
}
```

Equivalent Mondial SQLite Query:

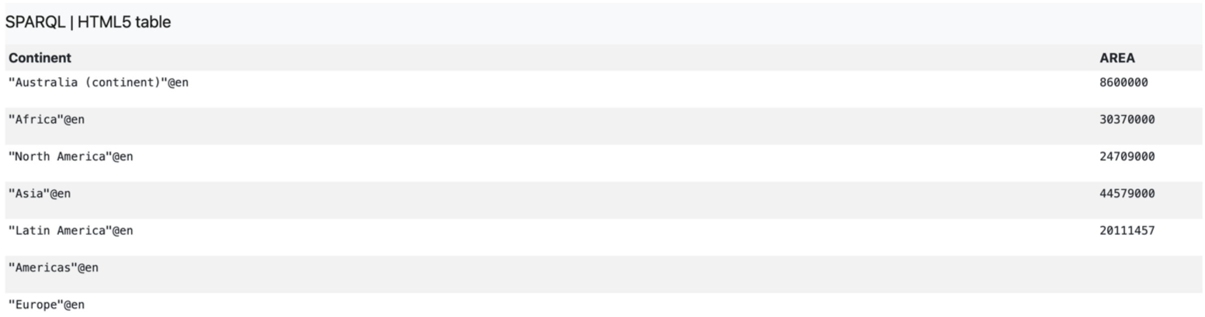
```
SELECT *
FROM CONTINENT;
```

TimeSheet for Retrieval:

TASK	TIME SPENT	COMMENTS
Refine query to eliminate duplicates and redundancies, ensuring optimal results.	1 hour	- Identified that redundancy was caused by the presence of multiple languages. - Used FILTER to retain only the records in English. - Converted the area from square meters to square kilometers.

Comments on Effectiveness:

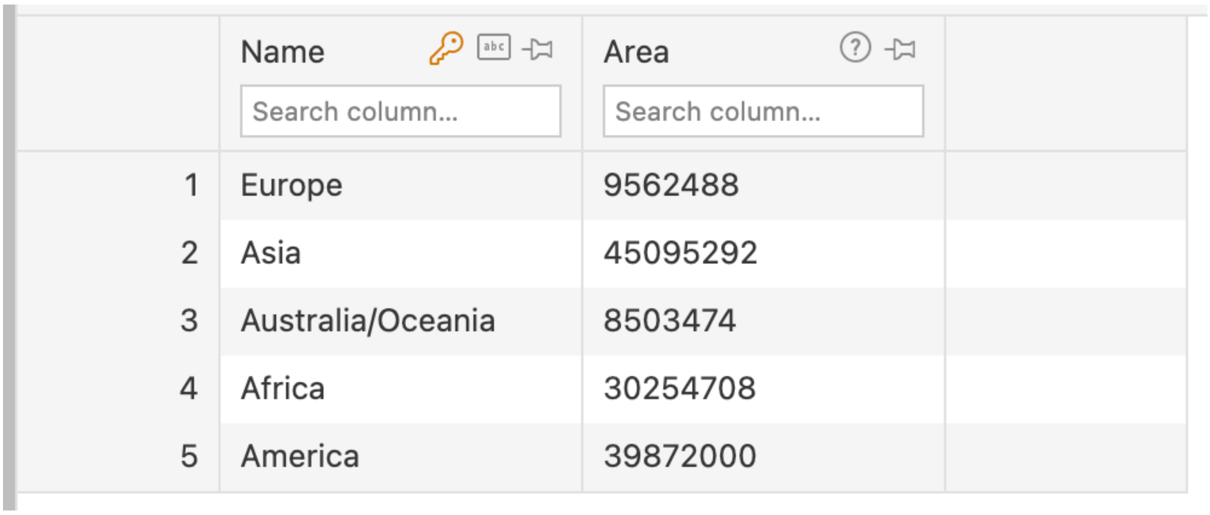
The execution of this task was considerably effective due to the availability of corresponding data in the Mondial database. The initial challenge was identifying the accurate continent property within the **umbel-rc** namespace. The exploratory phase revealed almost matching results between the Mondial and SPARQL queries, with discrepancies in the area for the continents of the Americas and Europe. The area for the Americas could potentially be deduced by summing North and Latin America. However, there is no explanation for why it is missing for *Europe.*



SPARQL | HTML5 table

Continent	AREA
"Australia (continent)"@en	8600000
"Africa"@en	30370000
"North America"@en	24709000
"Asia"@en	44579000
"Latin America"@en	20111457
"Americas"@en	
"Europe"@en	

Fig. 6 Results from SPARQL Query



	Name	Area
	Search column...	Search column...
1	Europe	9562488
2	Asia	45095292
3	Australia/Oceania	8503474
4	Africa	30254708
5	America	39872000

Fig. 7 Results from Mondial

Comments on Efficiency:

This task roughly took 3 hours to complete. There was some initial hurdle for identifying the appropriate namespace but the namespace prefix document [4] proved to be useful for this task. The semi-structured nature of this format did not hinder the fetching of the required data for this query.

Entity 6: Airport

The Airport entity is another non-geographical entity in mondial which holds columns for IATACode, Name, Country, City, Province, Island, Latitude, Longitude, Elevation, and gmtOffset. The IATACode serves as the primary key, uniquely identifying each airport. For example, the London Heathrow Airport, would have the IATACode `LHR`.

Exploratory Query:

```
SELECT ?airport
WHERE {
?airport a dbo:Airport .
}
LIMIT 50
```

TimeSheet for Exploratory:

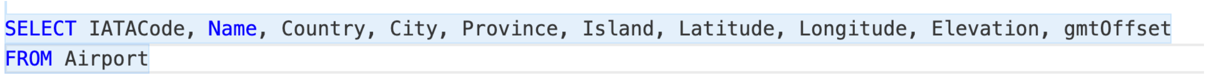
TASK	TIME SPENT	COMMENTS
Analyze the presence of required predicates	2.5 hours	- Spent a significant amount of time trying to understand where all the 10 relevant columns are present in the records. - Tried to find out if the relevant data is present in one of the hyperlinks to the other properties. For instance, if data is missing in dbo:aiport, can I find it using some other property within dbo:airport such as dbo:country or dbo:cityServed?
Structuring the query to maximize data retrieval	1.5 hours	- Decided which fields to keep optional and devised a technique to get the `Province` column using the dbo:subdivision property. - Couldn’t find any property that relates to `island`. - This took a significant time to search.

Retrieval Query:

```
SELECT ?IATACode  ?airportName ?countryName ?city
            (GROUP_CONCAT(DISTINCT STR(?subdivision); separator = ", ") AS ?province)
(xsd:decimal(REPLACE(?coordinates, " .*", "")) AS ?latitude)
(xsd:decimal(REPLACE(?coordinates, ".* ", "")) AS ?longitude) ?elevation ?gmtOffset
WHERE {
?airport a dbo:Airport .
?airport rdfs:label ?airportName . FILTER(lang(?airportName) = 'en')
?airport dbo:city ?ct.
 ?ct rdfs:label ?city  . FILTER(lang(?city) = 'en')
?airport dbo:iataLocationIdentifier ?IATACode .
 OPTIONAL{    ?ct dbo:country ?country .
     ?country rdfs:label ?countryName .
     ?country dbo:utcOffset ?gmtOffset.
     FILTER (lang(?countryName) = "en")}
OPTIONAL { ?airport georss:point ?coordinates }
OPTIONAL { ?airport dbo:elevation ?elevation }
OPTIONAL {
       ?ct dbo:subdivision ?subdivisionURL.
       ?subdivisionURL rdfs:label ?subdivision
```

```
        FILTER (lang(?subdivision) = "en")
    }
}
LIMIT 50
```

Equivalent Mondial SQLite Query:



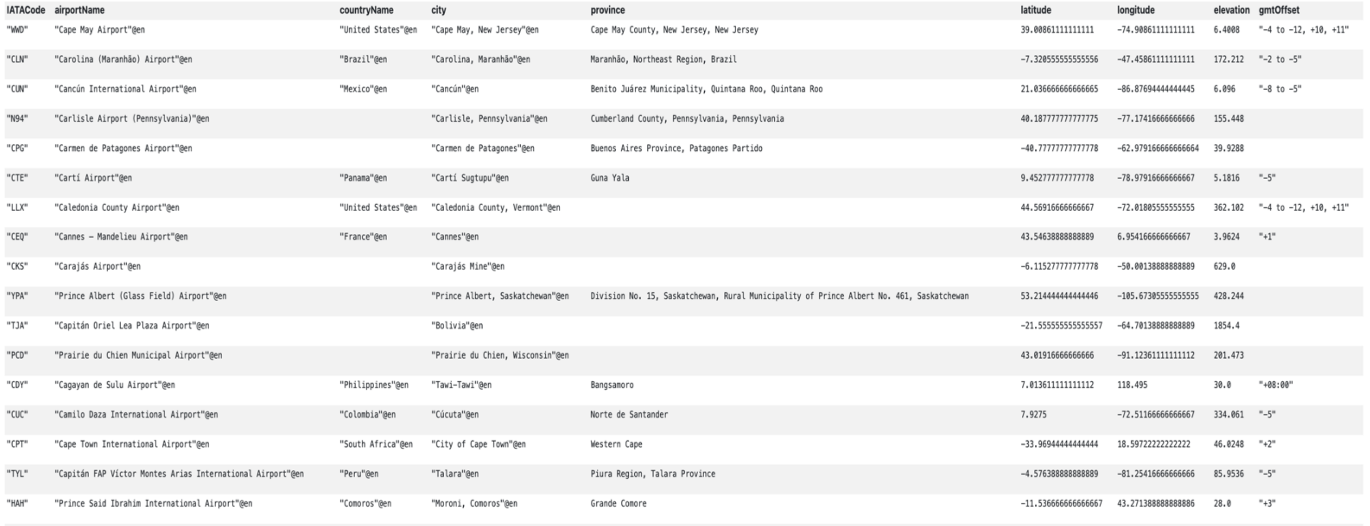
```
SELECT IATACode, Name, Country, City, Province, Island, Latitude, Longitude, Elevation, gmtOffset
FROM Airport
```

TimeSheet for Retrieval:

TASK	TIME SPENT	COMMENTS
Decision-making on OPTIONAL fields	1.5 hours	- Used a trial and error approach to understand which version of the retrieval yields results most similar to the Mondial database by experimenting with the `OPTIONAL` fields in the query. - Decided to keep `dbo: country` optional since 1000s of airports do not contain this value leading to missing and inconsistent results.
Investigate on the gmtOffset field	1 hour	- Found a property dbo: utcOffset and studied its relevance to the gmtOffset field present in the Mondial table. GMT is the actual timezone whereas UTC is the time standard [5], since both are same in terms of the offset value but only differ in notation, it is appropriate to use it in the gmtOffset column.

Comments on Effectiveness:

The airport table returns 6,730 records, a few of which is shown in figure 8, which is significantly more than the 1,320 records contained in the Mondial table. Part of the reason for this discrepancy is that DBpedia's results include not only commercial airports but also private, military airports, aviation schools, etc. This leads to an overpopulation of entries, which is difficult to control since there is no flag or indication of this aspect directly mentioned on the respective airport pages. The gmtOffset values can be slightly different from those in the Mondial table; however, this difference adds more completeness to the records, which improves the quality of the output when benchmarked against Mondial. Overall, the SPARQL query does a respectable job of completing the task of retrieving airports and related information.



IATACode	airportName	countryName	city	province	latitude	longitude	elevation	gmtOffset
"WWD"	"Cape May Airport"@en	"United States"@en	"Cape May, New Jersey"@en	Cape May County, New Jersey, New Jersey	39.00861111111111	-74.90861111111111	6.4008	"-4 to -12, +10, +11"
"CLN"	"Carolina (Maranhão) Airport"@en	"Brazil"@en	"Carolina, Maranhão"@en	Maranhão, Northeast Region, Brazil	-7.320555555555556	-47.45861111111111	172.212	"-2 to -5"
"CUN"	"Cancún International Airport"@en	"Mexico"@en	"Cancún"@en	Benito Juárez Municipality, Quintana Roo, Quintana Roo	21.036666666666665	-86.87694444444445	6.096	"-8 to -5"
"N94"	"Carlisle Airport (Pennsylvania)"@en		"Carlisle, Pennsylvania"@en	Cumberland County, Pennsylvania, Pennsylvania	40.187777777777775	-77.17416666666666	155.448	
"CPG"	"Carmen de Patagones Airport"@en		"Carmen de Patagones"@en	Buenos Aires Province, Patagones Partido	-40.77777777777778	-62.979166666666664	39.9288	
"CTE"	"Cartí Airport"@en	"Panama"@en	"Cartí Sugtupu"@en	Guna Yala	9.452777777777778	-78.97916666666667	5.1816	"-5"
"LLX"	"Caledonia County Airport"@en	"United States"@en	"Caledonia County, Vermont"@en		44.56916666666667	-72.01805555555555	362.102	"-4 to -12, +10, +11"
"CEQ"	"Cannes – Mandelieu Airport"@en	"France"@en	"Cannes"@en		43.54638888888889	6.954166666666667	3.9624	"+1"
"CKS"	"Carajás Airport"@en		"Carajás Mine"@en		-6.115277777777778	-50.00138888888889	629.0	
"YPA"	"Prince Albert (Glass Field) Airport"@en		"Prince Albert, Saskatchewan"@en	Division No. 15, Saskatchewan, Rural Municipality of Prince Albert No. 461, Saskatchewan	53.214444444444446	-105.67305555555555	428.244	
"TJA"	"Capitán Oriel Lea Plaza Airport"@en		"Bolivia"@en		-21.555555555555557	-64.70138888888889	1854.4	
"PCD"	"Prairie du Chien Municipal Airport"@en		"Prairie du Chien, Wisconsin"@en		43.01916666666666	-91.12361111111112	201.473	
"CDY"	"Cagayan de Sulu Airport"@en	"Philippines"@en	"Tawi-Tawi"@en	Bangsamoro	7.013611111111112	118.495	30.0	"+08:00"
"CUC"	"Camilo Daza International Airport"@en	"Colombia"@en	"Cúcuta"@en	Norte de Santander	7.9275	-72.51166666666667	334.061	"-5"
"CPT"	"Cape Town International Airport"@en	"South Africa"@en	"City of Cape Town"@en	Western Cape	-33.96944444444444	18.59722222222222	46.0248	"+2"
"TYL"	"Capitán FAP Víctor Montes Arias International Airport"@en	"Peru"@en	"Talara"@en	Piura Region, Talara Province	-4.576388888888889	-81.25416666666666	85.9536	"-5"
"HAH"	"Prince Said Ibrahim International Airport"@en	"Comoros"@en	"Moroni, Comoros"@en	Grande Comore	-11.536666666666667	43.271388888888886	28.0	"+3"

Fig. 8 Output of the retrieval query

Comments on Efficiency:

The task proved challenging due to the ambiguous nature of airport location properties. Distinguishing which property among dpr:location, dpo:cityServed, and dbo:city accurately reflects Mondial's data required extensive analysis. Identifying which one of these three actually gives the desirable output was very time-consuming. The choice to use dbo:city was based on its provision of atomic values closely aligned with Mondial's dataset. Furthermore, determining whether the subdivision provides acceptable values to mirror the Province table was quite challenging. The absence of a schema defining constraints for this in the RDF graph was felt significantly for this task. A lot of manual analysis was required to ensure a good quality output.

Entity 7: Economy

The Economy entity in Mondial includes Country, GDP, Agriculture, Service, Industry, Inflation, Unemployment. The primary key is country. This table provides important non-geographical context about the countries, hence it has been selected.

Exploratory Query:

```
SELECT DISTINCT ?C
WHERE {
  ?C a dbo:PopulatedPlace .
}
```

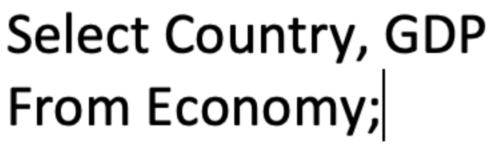
TimeSheet for Exploratory:

TASK	TIME SPENT	COMMENTS
Understanding the Economy entity columns.	30 minutes	- Understood how GDP, cashflow and population are all interconnected for this computation. - Studied enough background to actually calculate GDP from other variables if it is not available readily.
Analyzing predicates for economic data	3 hours	- A lot of time was dedicated towards finding where to start to find information about GDP, unemployment, inflation etc. - Found dbo:PopulatedPlace to be good starting point when compared to other entities. - Spent more time to find specific details with the property of dbo:gdp but could not find anything valuable.

Retrieval Query:

```
SELECT DISTINCT ?country (STR(?gdpNominal) AS ?gdp)
WHERE {
  ?C a dbo:PopulatedPlace .
 ?C rdfs:label ?country.
?C dbo:countryCode ?code.
  ?C dbp:gdpNominal ?gdpNominal .
FILTER ( LANG(?country)=”en”)
}
GROUP BY ?C ?gdpNominal
```

Equivalent Mondial SQLite Query:



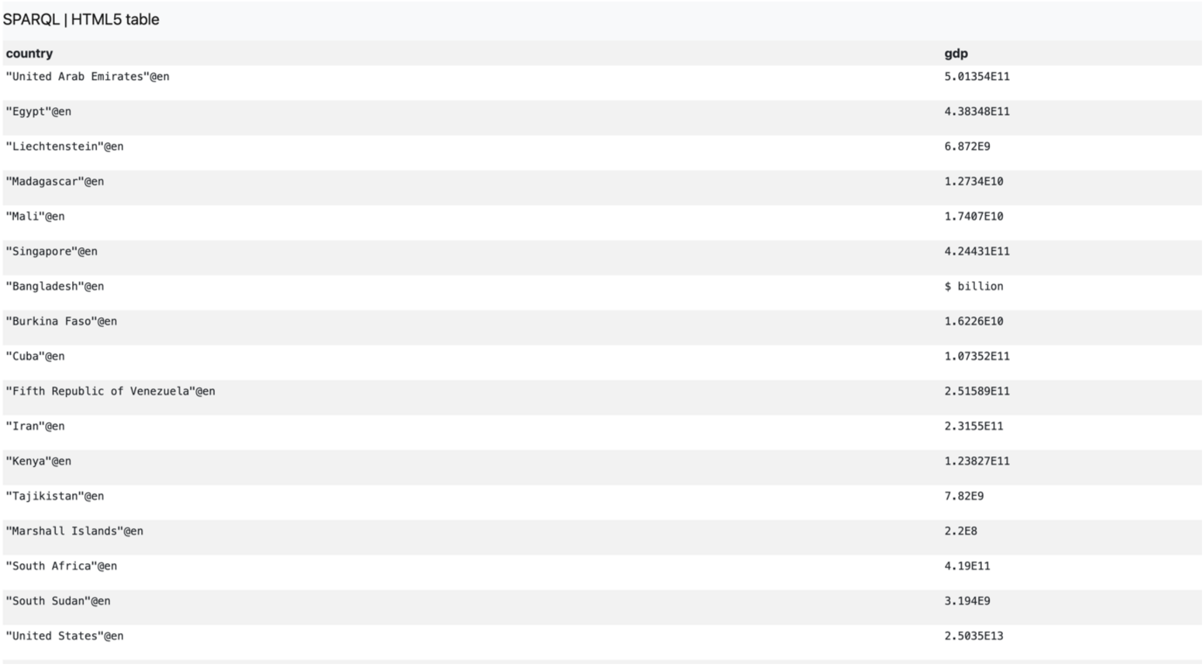
```
Select Country, GDP
From Economy;
```

TimeSheet for Retrieval:

TASK	TIME SPENT	COMMENTS
Study of **dbo:gdpNominal** property	1 hour	- Since there was no direct property present for gdp, I decided to figure out if gdpNominal can be used. On researching, I learnt gdpNominal is closely related to gbp but it is just inflation-adjusted to the current market value [6]. - Decided to include this in the GDP column since it serves the same purpose that GDP does in such records.
Find Agriculture, Service, Industry, Inflation, Unemployment values for the entity	3.5 hours	- Went through more than 12 namespaces present at [7] in the owl:sameAs property for unemployment to find data, some of which returned a `502 bad gateway` error. Did not manage to find unemployment data. - Despite numerous attempts at finding data for inflation, found no such direct property that could be cross-referenced from the GDP values. - Efforts were not fruitful.

Comments on Effectiveness:

This task was effective in terms of the number of countries returned, which is 322 for DBpedia and 244 for Mondial, respectively. Nonetheless, it is even more ineffective when discussing the number of columns filled, which is 1 out of 6. This may be because the RDF graph organizes data a lot differently than Mondial. For instance, while nominal GDP is associated with **dbo:Country**, unemployment correlates with **dbo:PersonFunction** [7]. This leads to an inconsistency that would require extracting data from various sources and transforming and storing them into a new graph for the economy altogether. Although the possibility exists that unexplored namespaces may contain pertinent economic data, over 8 hours of diligent search yielded no further progress, illustrating the difficulty of data retrieval without a guiding schema or documentation.



SPARQL | HTML5 table

country	gdp
"United Arab Emirates"@en	5.01354E11
"Egypt"@en	4.38348E11
"Liechtenstein"@en	6.872E9
"Madagascar"@en	1.2734E10
"Mali"@en	1.7407E10
"Singapore"@en	4.24431E11
"Bangladesh"@en	$ billion
"Burkina Faso"@en	1.6226E10
"Cuba"@en	1.07352E11
"Fifth Republic of Venezuela"@en	2.51589E11
"Iran"@en	2.3155E11
"Kenya"@en	1.23827E11
"Tajikistan"@en	7.82E9
"Marshall Islands"@en	2.2E8
"South Africa"@en	4.19E11
"South Sudan"@en	3.194E9
"United States"@en	2.5035E13

Fig. 9 Output of the retrieval query

Comments on Efficiency:

This was one of the entities selected that did not yield desirable results, despite rigorously dedicating significant time to it. The lack of a definitive structure impeded the development of a comprehensive query to encapsulate all necessary economic dimensions. It may be beneficial to devise a bespoke ontology specifically tailored to expressing economic data, which would streamline both querying and data representation. Although, that is beyond the scope of this coursework. The GDP nominal data was converted using STR to exclude any extraneous links or metadata.

Entity 8: Ethnic Group

The EthnicGroup table contains columns for Country, Name, and Percentage. The primary key, a combination of Name and Country, uniquely identifies an ethnic group within a country, such as 'Bengali' in India, indicated by 'Bengali' and 'IND'.

Exploratory Query:

```
SELECT DISTINCT ?e
WHERE {
  ?e a dbo:EthnicGroup.
}
```

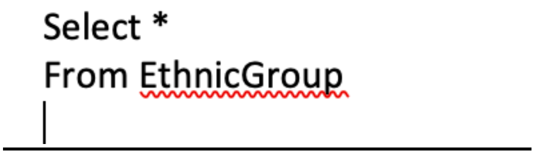
TimeSheet for Exploratory:

TASK	TIME SPENT	COMMENTS
Analyze predicate information that matches the Mondial database.	1.5 hours	- Found a lot more ethnicities than expected, partially due to the more detailed bifurcation for every country/region. - Unusual entries such as 'Caffeine Free coca cola' was found associated with dbo:EthnicGroup
Search for percentage information for each group.	1 hour	- Could not find this information. - One way of doing this could be to find the country and use a population of the country and the population of the ethnic group to find the percentage. - However, this is not feasible since not all ethnic groups belong to a single country, so the computation can give incorrect results.

Retrieval Query:

```
SELECT DISTINCT ?name STR(?p) AS ?POPULATION
WHERE {
  ?e a dbo:EthnicGroup.
?e rdfs:label ?name.
?e dbo:totalPopulation ?p.
FILTER (LANG(?name)="en")
}
```

Equivalent Mondial SQLite Query:



```
Select *
From EthnicGroup
```

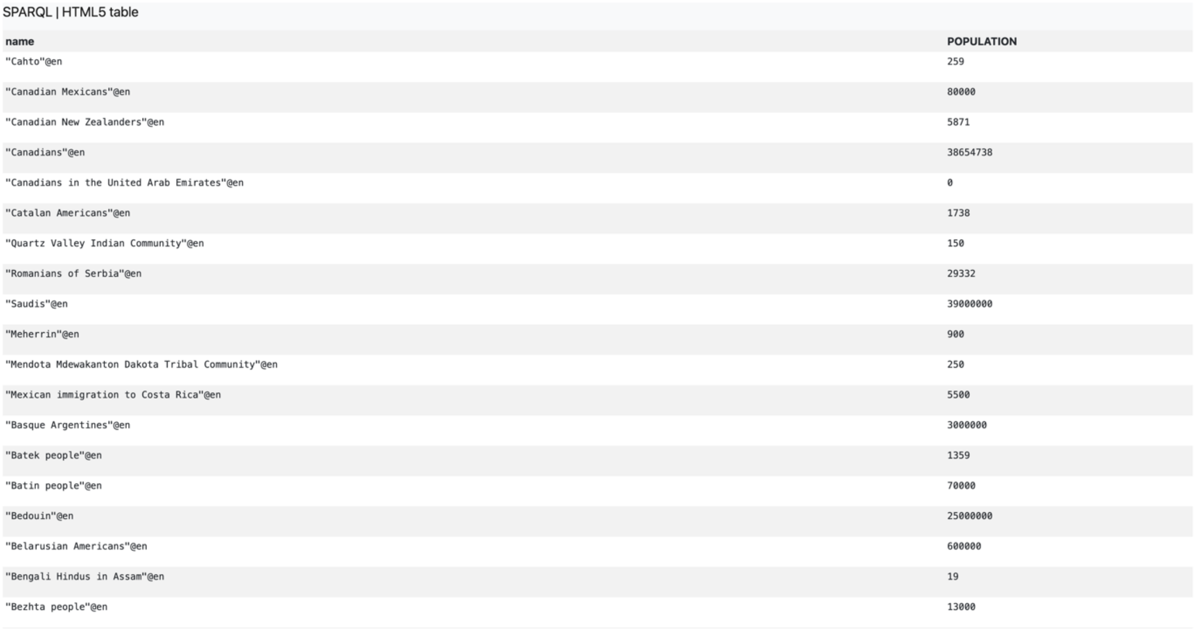
TimeSheet for Retrieval:

TASK	TIME SPENT	COMMENTS
Compare the results of the retrieval with the Mondial table	1 hour	- Identified a lot of additional diasporas and ethnicities compared to Mondial.

		- Thought of ways to aggregate some of the results – but couldn't implement it. - For instance, 'Romanians' has 14 different records for different regions, which I wanted to combine but it didn't work out.
Attempt to compute percentage from population data.	1.5 hours	- The population data was creating redundancy for the results since a lot of columns had a missing value which led to a value of `0%` being stored in the population column, which is incorrect.

Comments on Effectiveness:

The retrieval query yields 3,943 records, which is significantly more than the 628 records present in Mondial. Although this results in the output overpopulating the Mondial table, it also indicates that the data in DBpedia is more detailed and specific, capturing the nuances of ethnicity. It covers ethnic groups extensively but fails to provide information about the country and the percentage of the population. This is because both pieces of information are very difficult to cross-reference and/or compute. Most ethnic groups do not have a dbo:country property and most contain only information about the population, which is included in Figure 10 below for illustration purposes.



SPARQL | HTML5 table

name	POPULATION
"Cahto"@en	259
"Canadian Mexicans"@en	80000
"Canadian New Zealanders"@en	5871
"Canadians"@en	38654738
"Canadians in the United Arab Emirates"@en	0
"Catalan Americans"@en	1738
"Quartz Valley Indian Community"@en	150
"Romanians of Serbia"@en	29332
"Saudis"@en	39000000
"Meherrin"@en	900
"Mendota Mdewakanton Dakota Tribal Community"@en	250
"Mexican immigration to Costa Rica"@en	5500
"Basque Argentines"@en	3000000
"Batek people"@en	1359
"Batin people"@en	70000
"Bedouin"@en	25000000
"Belarusian Americans"@en	600000
"Bengali Hindus in Assam"@en	19
"Bezhta people"@en	13000

Fig 10. Output of the retrieval query

Comments on Efficiency:

The query provided a data-rich exploration of ethnic groups. Despite missing two columns, the output inclusively matched all ethnic group entries against the Mondial benchmark. A schema mandating the inclusion of a dbo:country property in the RDF could have allowed for manual computation of the missing columns.

References

1. https://www.dbis.informatik.uni-goettingen.de/Mondial/mondial-abh.pdf
2. https://ecpr.eu/Events/Event/PanelDetails/10569
3. https://www.w3.org/TR/sparql11-query/#defn_aggSample
4. https://dbpedia.org/sparql?nsdecl
5. https://24timezones.com/gmt-vs-utc
6. https://www.investopedia.com/ask/answers/030515/when-do-economists-use-real-gdp-instead-gdp.asp#:~:text=Key%20Takeaways,any%20distorting%20effects%20from%20inflation.
7. https://dbpedia.org/page/Unemployment