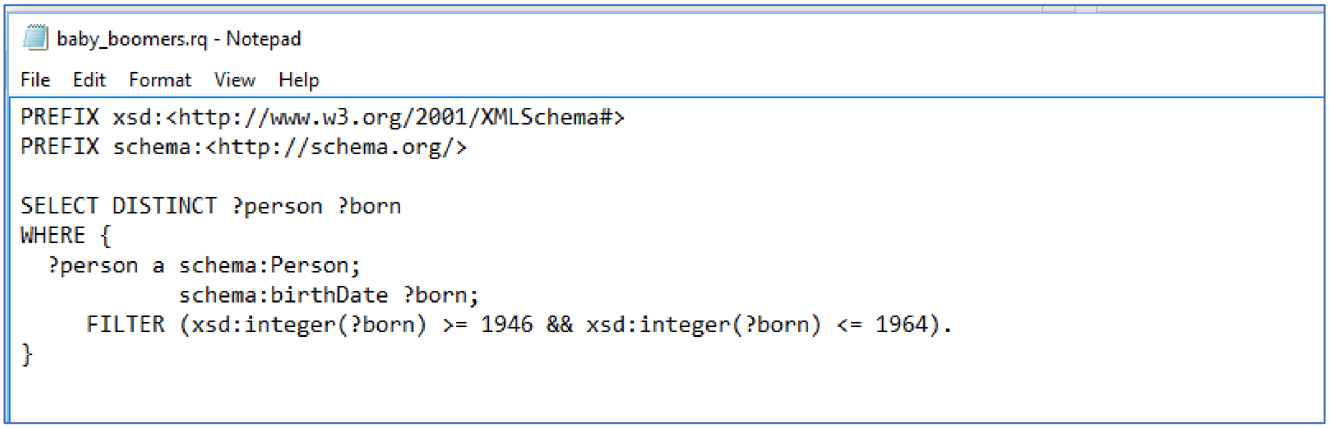
EXERCISE 4:

How many people in the dataset are "baby boomers"? HINT: "Baby boomers" is a term used to describe the generation born after WWII, usually between the years 1946 and 1964.

One way to solve this exercise is to write a query like the following:



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

SELECT DISTINCT ?person ?born
WHERE {
  ?person a schema:Person;
           schema:birthDate ?born;
     FILTER (xsd:integer(?born) >= 1946 && xsd:integer(?born) <= 1964).
}
```

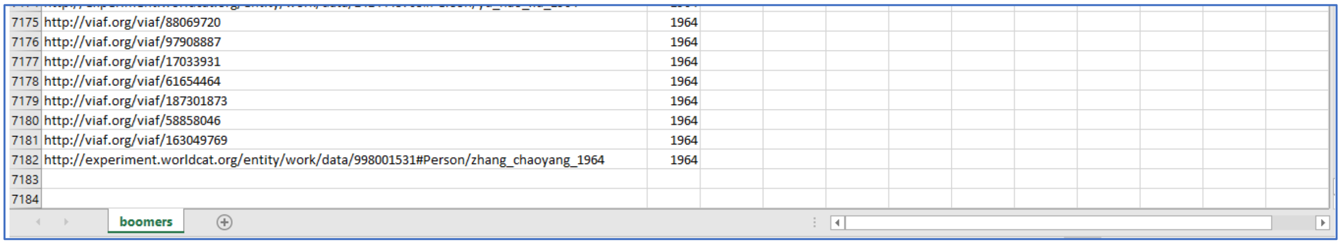
This query first asks for all individuals of the class *"Person"*, then asks for the value of the property *"birthDate"* for each individual and stores it in the variable *"?born"*. Then, the FILTER command is used to select only those results for individuals whose *birthDate* had a value between 1946 and 1964.

An important thing to note- while we know that the values for birthDates in this dataset are literals, *we might not know their datatype*. So, to ensure that the values are integers which can be compared to one another, we need to convert the values stored in *?born* to integers using the *xsd:integer* function. In this case, *"xsd:"* is the abbreviation for **XML Schema**, which we made sure to declare as a *namespace* before the query using PREFIX. Once we are sure the values are integers, we can simply filter the results asking for only those birthDates greater-than-or-equal to 1946 and less-than-or-equal. The syntax for this is the same as that used in virtually every programming language (*"<="* and *">="* for the mathematical comparisons; *"&&"* to combine the two into one conditional statement).

Then, run it from the command line and save it as a CSV file.

```
tdbquery --loc libsci --results CSV --time --query baby_boomers.rq > boomers.csv
```

Finally, open the file and scroll down to see how many rows are in it (don't forget to subtract one for the header).

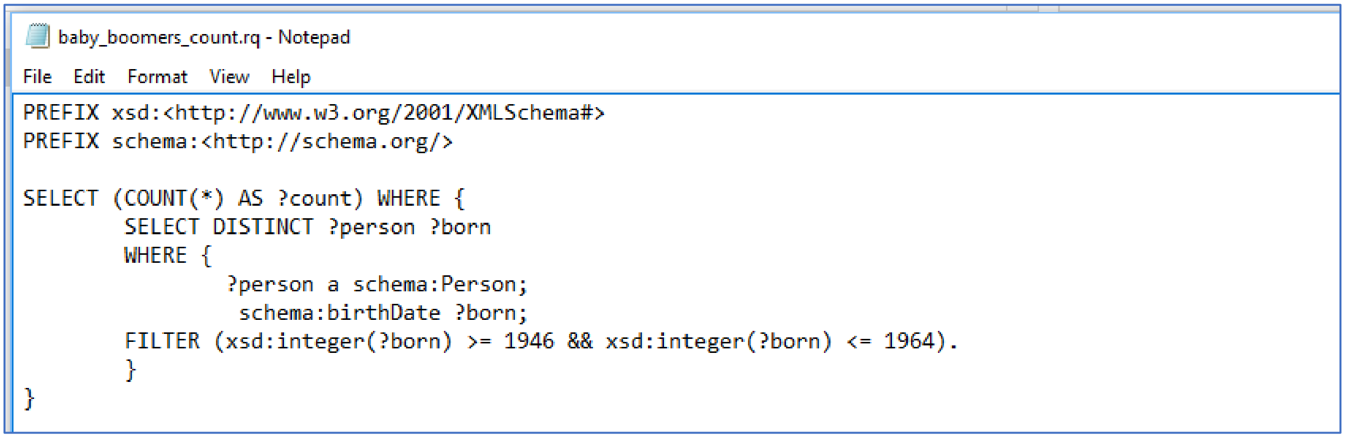


7175	http://viaf.org/viaf/88069720	1964
7176	http://viaf.org/viaf/97908887	1964
7177	http://viaf.org/viaf/17033931	1964
7178	http://viaf.org/viaf/61654464	1964
7179	http://viaf.org/viaf/187301873	1964
7180	http://viaf.org/viaf/58858046	1964
7181	http://viaf.org/viaf/163049769	1964
7182	http://experiment.worldcat.org/entity/work/data/998001531#Person/zhang_chaoyang_1964	1964
7183		
7184		

Of course, we could also just let the output display on the terminal, but this has a major drawback - several thousand un-numbered lines will be displayed in the result set. Just try counting them all manually!

A BETTER WAY TO DO IT:

However, if we don't really want a table with all the baby boomers in it to be produced, we just want to know how many people meet the criteria in our dataset, we can use the following trick:



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

SELECT (COUNT(*) AS ?count) WHERE {
        SELECT DISTINCT ?person ?born
        WHERE {
                ?person a schema:Person;
                 schema:birthDate ?born;
        FILTER (xsd:integer(?born) >= 1946 && xsd:integer(?born) <= 1964).
        }
}
```

All we have done here is take the previous query and "wrap" it in another SELECT statement. The *outer* SELECT statement asks for the COUNT of all ("*") the results from the *inner* SELECT statement to be saved in a *"?count"* variable and for this *?count* variable to be returned as the result of the query.

We can run this query from the command line and not bother with the extra step of saving the results to a file, because all this query will output to the terminal is the number we want!

ANSWER: 7181

QUERY: baby_boomers.rq *OR* baby_boomers_count.rq