Example Queries for SPARQL

1. All Actors from Game of Thrones

URL: https://w.wiki/65ke

```
SELECT ?actor ?actorLabel
WHERE {
  wd:Q23572 wdt:P161 ?actor. # The triple we have is Game of Thrones - Cast member - Actor.
                             # But note that it doesnt go the other direction!
                             # (i.e actors dont have a "cast member in" predicate)

 SERVICE wikibase:label { bd:serviceParam wikibase:language "en".}
  # Entires are identified by their Q numbers,properties by P numbers.
  # The label service is built in and we can use it like this.
  # Labels are the human readable component to identifiers.
  # To see them in the output; for any ?variable, also SELECT its ?variableLabel.
  # Try changing the language to "de" and SELECT ?actorDescription as well!
}
```

2. Nationality distribution of GoT Actors

URL: https://w.wiki/65ma

```
SELECT (COUNT(?actor) As ?numActors) # We would like to see the distribution of where GoT actors are from
                                     # So we need to get COUNT the actors and GROUP them BY the countries
        ?countryLabel # Display the country names
WHERE {
  wd:Q23572 wdt:P161 ?actor. # Get all GoT actors
  ?actor wdt:P27 ?country.   # Get the country of citizenship of actors
  SERVICE wikibase:label { bd:serviceParam wikibase:language "en".}
}
GROUP BY (?countryLabel) # We GROUP the actors BY their countries. Works exactly the same as in SQL.
ORDER BY DESC(?numActors) # We can sort the output for a better view.
```

(Bonus) Get the actor/character pairs from GoT

I'd say this a bit advanced though, but here for completeness since we talked about it.
URL: https://w.wiki/65$Q

```
SELECT ?actorLabel ?characterLabel
WHERE {
  # The character role information is not itself stored as a triple,
  # but as a so-called "qualifier" to the triple GoT-cast member-actor.
  # Retrieving this information works as follows:
  wd:Q23572 p:P161 ?statement. # The prefix p: points not to the object, but to a statement node.
                              # This node then is the subject of other triples. Like an address pointer.
  ?statement ps:P161 ?actor. # The prefix ps: within the statement node retrieves the object. In our case the actor.
  ?statement pq:P453 ?character. # The prefix pq: within the statement node retrieves the qualifier information. In our case the character.


 SERVICE wikibase:label { bd:serviceParam wikibase:language "en".}
}
```

3. Actors whose birthdays are today

URL: https://w.wiki/65yc

```
SELECT ?actorLabel (YEAR(?date) as ?birthYear)
WHERE {
  BIND(MONTH(NOW()) AS ?nowMonth) # BINDing can be thought of as storing some value in a variable
  BIND(DAY(NOW()) AS ?nowDay)

  ?actor wdt:P106 wd:Q33999. # Get all actors
  ?actor wdt:P569 ?date. # Get the date of birth for all actors
  FILTER (MONTH(?date) = ?nowMonth && DAY(?date) = ?nowDay) # We FILTER the output we get by removing elements
                                                            # that don't fit a given conditional
                                                            # in this case birthdays that aren't today
  SERVICE wikibase:label {
    bd:serviceParam wikibase:language "en" .
  }
}
LIMIT 30 #LIMITing allows you to bound the number of elements you get as your output
```

(Note BB: also works without BIND of course)

4. Number of actors per 1 million inhabitant of countries in the EU

URL: https://w.wiki/65$3

```
SELECT ?countryLabel ?population # Get the country names and their populations
  (ROUND(?actors/?population*1000000) AS ?actorsPerMillionInhabitants) # Get the number of actors per 1 million
WHERE
{
    # Nested queries are executed from "inside out, meaning first the section below will be executed.
    # You can only put subqueries like this within the WHERE block.
    # Think about whether you can achieve the same result without nesting and what the disadvantage might be.
  { SELECT DISTINCT ?country (count(*) as ?actors)
    WHERE {
      wd:Q458 wdt:P150 ?country . # Get the countries in the EU
      ?actor wdt:P106 wd:Q33999. # Get all actors
      ?actor wdt:P27 ?country . # Associate actors with the country they're from,
    } GROUP BY ?country  # GROUPed BY the country
  }
  ?country wdt:P1082 ?population # Get the populations of the countries.

  SERVICE wikibase:label { bd:serviceParam wikibase:language "en" }
}
ORDER BY DESC(?actorsPerMillionInhabitants)
```

5. Most successful actors from every country in the world

I think the “messiness” of this query (and its results, it returns for example the same person once as from Russia and once as from the Soviet Union) would make a good discussion. How to measure “successful” etc...

URL: https://w.wiki/66Gz

```
SELECT DISTINCT ?countryLabel ?ActorLabel ?AwardsReceived
WHERE {
      # Here we can not simply SELECT ?actor, because we are grouping by countries
      # Therefore SAMPLE is used to pick one of the actors from that country with the MAX number of awards
      # (note that it could easily be the case that 2 actors have the same country and num. awards)
  {SELECT ?country (SAMPLE(?actor) as ?Actor) (MAX(?AwardCount) as ?AwardsReceived)
  WHERE {
   { SELECT ?actor (COUNT(?award) as ?AwardCount) # count the awards of each actor
    WHERE{
        ?actor wdt:P106 wd:Q33999. # get all actors
        ?actor wdt:P166 ?award. # get all awards associated with actors
        FILTER NOT EXISTS{?actor wdt:P570 ?date } # only count living people
      } GROUP BY ?actor
    }
    ?actor wdt:P27 ?country. # get the countries actors are from
  } GROUP BY ?country }
    SERVICE wikibase:label { bd:serviceParam wikibase:language "en" }

}
LIMIT 10
```

Note on CSV export: Its super easy to do, at the top right corner of the result box, just click download and select “CSV file” from the drop down. First you need to run the query of course.