Trabajo extraclase - consultas SPARQL a DBpedia (individual)

In [3]:
```
pip install SPARQLWrapper
```

```
Collecting SPARQLWrapper
  Downloading SPARQLWrapper-2.0.0-py3-none-any.whl.metadata (2.0 kB)
Collecting rdflib>=6.1.1 (from SPARQLWrapper)
  Downloading rdflib-7.5.0-py3-none-any.whl.metadata (12 kB)
Requirement already satisfied: pyparsing<4,>=2.1.0 in /usr/local/lib/python3.12/dist
-packages (from rdflib>=6.1.1->SPARQLWrapper) (3.2.5)
Downloading SPARQLWrapper-2.0.0-py3-none-any.whl (28 kB)
Downloading rdflib-7.5.0-py3-none-any.whl (587 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0.0/587.2 kB ? eta -:--:--
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 587.2/587.2 kB 34.4 MB/s eta 0:00:00
Installing collected packages: rdflib, SPARQLWrapper
Successfully installed SPARQLWrapper-2.0.0 rdflib-7.5.0
```

In [4]:
```
import sys
from SPARQLWrapper import SPARQLWrapper, JSON, RDF, XML
from pandas import json_normalize
import pandas as pd
```

definicion de endpoint

In [5]:
```
endPoint = "http://dbpedia.org/sparql"
sparql = SPARQLWrapper(endPoint)
```

1.- Consultar cómo se usan los comandos ASK, DESCRIBE y CONSTRUCT y proponer un ejemplo de consulta con cada opción.

Consulta de ASK

El comando ASK se utiliza para verificar si un patrón de triple (sujeto, predicado, objeto) existe en el conjunto de datos. • Resultado: No devuelve una tabla de datos, sino un valor booleano: true (si el patrón existe) o false (si no existe).

Estructura básica

ASK {

?sujeto ?predicado ?objeto .

}

In [16]:
```
#conuslta uno que regresa false
query_ASK_1 = """
 PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>

ASK {
  dbr:Jefferson_Pérez a dbo:Astronaut .
}
"""
```

In [17]:
```
#conuslta dos que regresa true
query_ASK_2 = """
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>

ASK {
  dbr:Jefferson_Pérez a dbo:Athlete .
}
"""
```

In [19]:
```
def get_ask_result(query_ASK_1):
    sparql.setReturnFormat(JSON)
    sparql.setQuery(query_ASK_1)
    results = sparql.query().convert()
    return results["boolean"]

    # Ejecución
es_astronauta = get_ask_result(query)

# Mostrar el resultado de forma elegante
print(f"¿Jefferson Pérez es astronauta?: {es_astronauta}")
```

```
¿Jefferson Pérez es astronauta?: False
```

In [21]:
```
def get_ask_result(query_ASK_2):
    sparql.setReturnFormat(JSON)
    sparql.setQuery(query_ASK_2)
    results = sparql.query().convert()
    return results["boolean"]

    # Ejecución
es_astronauta = get_ask_result(query_ASK_2)

# Mostrar el resultado de forma elegante
print(f"¿Jefferson Pérez es astronauta?: {es_astronauta}")
```

```
¿Jefferson Pérez es astronauta?: True
```

Consulta de DESCRIBE

¿Qué devuelve? A diferencia de SELECT (que devuelve tablas) o ASK (que devuelve un sí/no), DESCRIBE devuelve todas las relaciones donde el recurso que buscas es el sujeto (y a veces también donde es el objeto). con DESCRIBE le dices al servidor: "Oye, no sé exactamente qué datos tienes, pero dame todo lo que sepas sobre este recurso".

La ventaja: No necesitas conocer los nombres de las propiedades (como dbo:birthPlace o dbo:abstract) de antemano. El servidor decide qué información es relevante.

Estructura básica

DESCRIBE URI_del_recurso

In [43]:
```
query_describe = """
PREFIX dbr: <http://dbpedia.org/resource/>
DESCRIBE dbr:Jefferson_Pérez
"""
```

In [51]:
```
def get_simple_describe(query):
    sparql.setQuery(query)
    sparql.setReturnFormat(RDF)

    graph = sparql.query().convert()

    data = []
    for s, p, o in graph:
        data.append({
            "Sujeto": s.split("/")[-1],
            "Propiedad": p.split("/")[-1].split("#")[-1],
            "Valor": str(o)
        })

    return pd.DataFrame(data)

df = get_simple_describe(query_describe)
df.head(10)
```

```
/usr/local/lib/python3.12/dist-packages/SPARQLWrapper/Wrapper.py:794: RuntimeWarnin
g: Sending Accept header '*/*' because unexpected returned format 'rdf' in a 'DESCRI
BE' SPARQL query form
  warnings.warn(
```

Out[51]:

	Sujeto	Propiedad	
0	Athletics_at_the_2008_Summer_Olympics	wikiPageWikiLink	http://dbpedia.org/resource/
1	Jefferson_Pérez	wikiPageWikiLink	http://dbpedia.org/resou
2	Jefferson_Pérez	wikiPageWikiLink	http://dbpedia.org
3	Jefferson_Pérez	wikiPageWikiLink	http://dbpedia.org/resource/Ath
4	1992_Ibero-American_Championships_in_Athletics	wikiPageWikiLink	http://dbpedia.org/resource/
5	Jefferson_Pérez	wikiPageWikiLink	http://dbpedia.org/resource/Sain
6	Jefferson_Pérez	wikiPageWikiLink	http://dbpedia.org/resource/Ath
7	1991_IAAF_World_Indoor_Championships_–_Men's_5...	wikiPageWikiLink	http://dbpedia.org/resource/
8	Jefferson_Pérez	wikiPageWikiLink	http://dbpedia.org/resour
9	Jefferson_Pérez	subject	http://dbpedia.org/resource/Categ

Comsulta CONSTRUCT

CONSTRUCT es para crear o transformar. Es el comando más potente cuando quieres tomar datos de un sitio (como DBpedia) y convertirlos a tu propio formato o vocabulario.

El comando CONSTRUCT devuelve un grafo RDF (tripletas) basado en una plantilla que tú defines.

La Plantilla: Tú decides qué forma tendrán las nuevas tripletas.

El Patrón: Tú decides qué datos buscar en la base de datos original.

Estructura básica

CONSTRUCT {

AQUÍ defines la "plantilla" (cómo quieres que se vean los datos)

?sujeto mi_propiedad_personalizada ?objeto .

} WHERE {

AQUÍ buscas los datos reales en DBpedia

?sujeto propiedad_de_dbpedia ?objeto .

}

In [61]:
```
query_construct = """
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX kriss: <https://kriss.org/>

CONSTRUCT {
  dbr:Jefferson_Pérez a kriss:Person ;
                      kriss:name ?nombre ;
                      kriss:homeLocation ?lugar .
}
WHERE {
  dbr:Jefferson_Pérez rdfs:label ?nombre ;
                      dbo:birthPlace ?lugar .
  FILTER (lang(?nombre) = 'es')
}
"""
```

In [62]:
```
def get_simple_construct(query):
    sparql.setQuery(query)
    sparql.setReturnFormat(RDF)

    graph = sparql.query().convert()

    data = []
    for s, p, o in graph:
        data.append({
            "Sujeto": s.split("/")[-1],
            "Propiedad": p.split("/")[-1].split("#")[-1],
            "Valor": str(o)
        })

    return pd.DataFrame(data)

df = get_simple_construct(query_construct)
df
```

Out[62]:

	Sujeto	Propiedad	Valor
0	Jefferson_Pérez	homeLocation	http://dbpedia.org/resource/Cuenca,_Ecuador
1	Jefferson_Pérez	name	Jefferson Pérez
2	Jefferson_Pérez	type	https://kriss.org/Person
3	Jefferson_Pérez	homeLocation	http://dbpedia.org/resource/Azuay_Province

2.- Diseñar y ejecutar consultas SPARQL para responder las siguientes consultas:

2.1.- Presidentes de países de Suramérica que hayan nacido entre los años 1950 y 1980. Por cada político, indicar sus nombres, edad y ciudad de nacimiento.

In [82]:
```
query_presidentes_sudamericanos = """
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT DISTINCT ?nombre ?edad ?ciudadNacimiento
WHERE {
  ?p a dbo:President ;
     dbo:birthDate ?fechaNacimiento ;
     dbo:birthPlace ?ciudad .

  ?ciudad dbo:country ?pais .

  VALUES ?pais {
    dbr:Argentina
    dbr:Bolivia
    dbr:Brazil
    dbr:Chile
    dbr:Colombia
    dbr:Ecuador
    dbr:Paraguay
    dbr:Peru
    dbr:Uruguay
    dbr:Venezuela
  }

  ?p rdfs:label ?nombre .
  FILTER (LANG(?nombre) = "es")

  FILTER (
    YEAR(?fechaNacimiento) >= 1950 &&
    YEAR(?fechaNacimiento) <= 1980
  )

  BIND(YEAR(NOW()) - YEAR(?fechaNacimiento) AS ?edad)

  OPTIONAL {
    ?ciudad rdfs:label ?ciudadLabel .
    FILTER (LANG(?ciudadLabel) = "es")
  }

  BIND(COALESCE(?ciudadLabel, STR(?ciudad)) AS ?ciudadNacimiento)
}
ORDER BY ?nombre
LIMIT 20
"""
```

In [70]:
```
def get_results(query):
    sparql.setReturnFormat(JSON)
    sparql.setQuery(query)
    results = sparql.query().convert()
    return(results)
```

In [83]:
```
resultados = (get_results(query_presidentes_sudamericanos))

df = json_normalize(resultados["results"]["bindings"])

df.head(10)
```

Out[83]:

	nombre.type	nombre.xml:lang	nombre.value	edad.type	ed
0	literal	es	Freddy Bernal	literal	http://www.w3.org/2001/XMLSch
1	literal	es	Graciela Camaño	literal	http://www.w3.org/2001/XMLSch
2	literal	es	Nicanor Duarte Frutos	literal	http://www.w3.org/2001/XMLSch

2.2.-Presidentes del Ecuador que hayan nacido en ciudades que son capitales de provincia. Por cada presidente, presentar sus nombres, ciudad de nacimiento e instituciones donde estudió. Recomendación: explorar entre los sub-conceptos de dbc:Presidents_by_country

In [71]:
```
query_presidentes_ecuador = """
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbc: <http://dbpedia.org/resource/Category:>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT DISTINCT ?nombre ?ciudadNacimiento ?institucion
WHERE {
  # Presidentes del Ecuador
  ?presidente dct:subject dbc:Presidents_of_Ecuador ;
              dbo:birthPlace ?ciudad ;
              rdfs:label ?nombre .

  FILTER (LANG(?nombre) = "es")

  # Ciudades capitales de provincia
```

```
  VALUES ?ciudad {
    dbr:Quito
    dbr:Guayaquil
    <http://dbpedia.org/resource/Cuenca,_Ecuador>
    dbr:Ambato
    dbr:Riobamba
    <http://dbpedia.org/resource/Loja,_Ecuador>
    dbr:Machala
    dbr:Portoviejo
    <http://dbpedia.org/resource/Ibarra,_Ecuador>
    <http://dbpedia.org/resource/Esmeraldas,_Ecuador>
  }

  # Nombre de la ciudad
  OPTIONAL {
    ?ciudad rdfs:label ?ciudadLabel .
    FILTER (LANG(?ciudadLabel) = "es")
  }
  BIND(COALESCE(?ciudadLabel, STR(?ciudad)) AS ?ciudadNacimiento)

  # Instituciones donde estudió
  OPTIONAL {
    ?presidente dbo:almaMater ?inst .
    ?inst rdfs:label ?institucion .
    FILTER (LANG(?institucion) = "es")
  }
}
ORDER BY ?nombre
LIMIT 50
"""
```

In [80]:

```
resultados = (get_results(query_presidentes_ecuador))

df = json_normalize(resultados["results"]["bindings"])

df.head(10)
```

Out[80]:

	nombre.type	nombre.xml:lang	nombre.value	ciudadNacimiento.type	ciudadNacimiento
0	literal	es	Abdalá Bucaram	literal	
1	literal	es	Abelardo Montalvo	literal	
2	literal	es	Alberto Guerrero Martínez	literal	
3	literal	es	Alfredo Baquerizo Moreno	literal	
4	literal	es	Antonio Borrero Cortázar	literal	
5	literal	es	Antonio Borrero Cortázar	literal	
6	literal	es	Antonio Flores Jijón	literal	
7	literal	es	Antonio Pons	literal	
8	literal	es	Aurelio Mosquera Narváez	literal	
9	literal	es	Camilo Ponce Enríquez	literal	

2.3.- Políticos (no solo presidentes) ecuatorianos que hayan desempeñado algún cargo nacional o provincial en el país y que hayan realizado sus estudios superiores fuera del país. Por cada político, presentar sus nombres, cargo desempeñado e instituciones donde realizó sus estudios.

In [77]:
```
query_ecuador_politicos = """
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

```
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX dbc: <http://dbpedia.org/resource/Category:>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbp: <http://dbpedia.org/property/>

SELECT DISTINCT ?nombre ?cargo ?institucion ?paisInstitucion
WHERE {
  # Buscar políticos ecuatorianos
  {
    ?persona dct:subject dbc:Ecuadorian_politicians .
  } UNION {
    ?persona dct:subject ?categoria .
    ?categoria skos:broader dbc:Ecuadorian_politicians .
  } UNION {
    ?persona dct:subject dbc:Presidents_of_Ecuador .
  }

  ?persona foaf:name ?nombre ;
           dbo:almaMater ?universidad .

  # La universidad debe estar fuera de Ecuador
  ?universidad dbo:country ?paisEstudio .
  FILTER(?paisEstudio != dbr:Ecuador)

  # Nombre de la universidad
  ?universidad rdfs:label ?uniLabel .
  FILTER(LANG(?uniLabel) = "es" || LANG(?uniLabel) = "en")
  BIND(?uniLabel AS ?institucion)

  # País de la universidad
  ?paisEstudio rdfs:label ?paisLabel .
  FILTER(LANG(?paisLabel) = "es" || LANG(?paisLabel) = "en")
  BIND(?paisLabel AS ?paisInstitucion)

  # Cargo desempeñado
  OPTIONAL {
    {
      ?persona dbo:office ?cargoURI .
      BIND(STR(?cargoURI) AS ?cargo)
    } UNION {
      ?persona dbp:office ?cargoTxt .
      BIND(STR(?cargoTxt) AS ?cargo)
    }
  }
}
ORDER BY ?nombre
LIMIT 50

"""
```

In [79]:

```
resultados = (get_results(query_ecuador_politicos))

df = json_normalize(resultados["results"]["bindings"])

df.head(10)
```

Out[79]:

	nombre.type	nombre.xml:lang	nombre.value	cargo.type	cargo.value	institucion.type
0	literal	en	Johana Pesántez	literal	Minister of Justice, Human Rights and Religiou...	literal
1	literal	en	Johana Pesántez	literal	Minister of Justice, Human Rights and Religiou...	literal
2	literal	en	Johana Pesántez	literal	Minister of Justice, Human Rights and Religiou...	literal
3	literal	en	Johana Pesántez	literal	Minister of Justice, Human Rights and Religiou...	literal
4	literal	en	Johana Pesántez	literal	Minister of Justice, Human Rights and Religiou...	literal
5	literal	en	Johana Pesántez	literal	Minister of Justice, Human Rights and Religiou...	literal
6	literal	en	José Joaquín de Olmedo	NaN	NaN	literal
7	literal	en	José Joaquín de Olmedo	NaN	NaN	literal
8	literal	en	José Joaquín de Olmedo	NaN	NaN	literal
9	literal	en	José Joaquín de Olmedo	NaN	NaN	literal

2.4.-Artistas ecuatorianos que hayan ganado algún tipo de premio y que hayan nacido en ciudades con menos de 600 mil habitantes. Por cada artista indicar su nombre, el premio alcanzado, su sexo, el nombre de la ciudad donde nació y la cantidad de habitantes de la ciudad.

In [89]:
```
query_ecuador_artistas = """
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX dbc: <http://dbpedia.org/resource/Category:>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>

SELECT DISTINCT ?nombre ?premio ?sexo ?ciudadNacimiento ?poblacion
WHERE {
  # Artistas ecuatorianos
  {
    ?persona dct:subject dbc:Ecuadorian_artists .
  } UNION {
    ?persona dct:subject ?categoria .
    ?categoria skos:broader* dbc:Ecuadorian_artists .
  }

  # Nombre y lugar de nacimiento
  ?persona foaf:name ?nombre ;
           dbo:birthPlace ?ciudadNac .

  # Nombre de la ciudad
  ?ciudadNac rdfs:label ?ciudadLabel .
  FILTER(LANG(?ciudadLabel) = "es" || LANG(?ciudadLabel) = "en")
  BIND(?ciudadLabel AS ?ciudadNacimiento)

  # Población
  OPTIONAL { ?ciudadNac dbo:populationTotal ?poblacion }
  OPTIONAL { ?ciudadNac dbp:populationTotal ?poblacion }

  FILTER(BOUND(?poblacion) && ?poblacion < 600000)

  # Premio - se usaron varias formas de obtener el premio
  OPTIONAL {
    ?persona dbo:award ?premioURI .
    ?premioURI rdfs:label ?premioLabel .
    FILTER(LANG(?premioLabel) = "es" || LANG(?premioLabel) = "en")
    BIND(?premioLabel AS ?premio)
  }
```

```
  OPTIONAL {
    ?persona dbp:awards ?premio .
  }

  # Sexo
  OPTIONAL { ?persona foaf:gender ?sexo }
  OPTIONAL {
    ?persona dbo:gender ?sexoURI .
    ?sexoURI rdfs:label ?sexoLabel .
    FILTER(LANG(?sexoLabel) = "es" || LANG(?sexoLabel) = "en")
    BIND(?sexoLabel AS ?sexo)
  }
}
ORDER BY ?premio
LIMIT 50

"""
```

In [90]:

```
resultados = (get_results(query_ecuador_artistas))

df = json_normalize(resultados["results"]["bindings"])

df.head(10)
```

Out[90]:

	nombre.type	nombre.xml:lang	nombre.value	premio.type	premio.xml:lang	premio.val
0	literal	en	Eduardo Kingman	literal	en	Prem Euger Espe
1	literal	en	Eduardo Kingman	literal	en	Prem Euger Espe
2	literal	en	Eduardo Kingman	literal	es	Prem Nacio Euger Espe
3	literal	en	Eduardo Kingman	literal	es	Prem Nacio Euger Espe
4	literal	en	María Estefanía Dávalos y Maldonado	NaN	NaN	N
5	literal	en	María Estefanía Dávalos y Maldonado	NaN	NaN	N
6	literal	en	Pablo Caviedes	NaN	NaN	N
7	literal	en	Pablo Caviedes	NaN	NaN	N
8	literal	en	Judith Gutiérrez	NaN	NaN	N
9	literal	en	Judith Gutiérrez	NaN	NaN	N

3.-Explicacion de aportes

Para la creacion de este trabajo se uso la IA de chat gpt para entender la posible estructura de dbpedia y como podrian relacionarse para luego comprobar los datos en wikipwedia y dbpedia, esto para explorar posibles vinculos mas efectivos. tambien me ayudo a ordenar las consultas con espacios y comentarios.

Tambien se uso Gemini para consulta de los primeros puntos y de las nuevas funciones, fue util para entender como se debia usar y luego construir los ejemplos.

Mi aporte

mi aporte fue ir seleccionando las propiedades correctas deacuerdo a la dbpedia ayudandome en parte de la wikipwdia y construi casi por completo las consultas guiandome de eplicaciones de la IA para encontrar las relaciones semanticas.

realice la comprobacion de todos los datos que regresaba el endpoint en https://dbpedia.org/sparql, asi pude ir construendo las consultas.

define las estructuras de los patrones de las busquedas identificando las propiedas y restricciones adecuadas.

con respecto al codigo de python que se uso lo tome del ejemplo dado en clase.

In [92]:
```
!jupyter nbconvert --to html "/content/drive/MyDrive/septimo_ciclo/interoperabilida
```

```
[NbConvertApp] WARNING | pattern '/content/drive/MyDrive/septimo_ciclo/interoperabil
idad/segundo/diciembre.ipynb' matched no files
This application is used to convert notebook files (*.ipynb)
        to various other formats.

        WARNING: THE COMMANDLINE INTERFACE MAY CHANGE IN FUTURE RELEASES.

Options
=======
The options below are convenience aliases to configurable class-options,
as listed in the "Equivalent to" description-line of the aliases.
To see all configurable class-options for some <cmd>, use:
    <cmd> --help-all

--debug
    set log level to logging.DEBUG (maximize logging output)
    Equivalent to: [--Application.log_level=10]
--show-config
    Show the application's configuration (human-readable format)
    Equivalent to: [--Application.show_config=True]
--show-config-json
    Show the application's configuration (json format)
    Equivalent to: [--Application.show_config_json=True]
--generate-config
    generate default config file
    Equivalent to: [--JupyterApp.generate_config=True]
-y
    Answer yes to any questions instead of prompting.
    Equivalent to: [--JupyterApp.answer_yes=True]
--execute
    Execute the notebook prior to export.
    Equivalent to: [--ExecutePreprocessor.enabled=True]
--allow-errors
    Continue notebook execution even if one of the cells throws an error and include
the error message in the cell output (the default behaviour is to abort conversion).
This flag is only relevant if '--execute' was specified, too.
    Equivalent to: [--ExecutePreprocessor.allow_errors=True]
--stdin
    read a single notebook file from stdin. Write the resulting notebook with defaul
t basename 'notebook.*'
    Equivalent to: [--NbConvertApp.from_stdin=True]
--stdout
    Write notebook output to stdout instead of files.
    Equivalent to: [--NbConvertApp.writer_class=StdoutWriter]
--inplace
    Run nbconvert in place, overwriting the existing notebook (only
            relevant when converting to notebook format)
    Equivalent to: [--NbConvertApp.use_output_suffix=False --NbConvertApp.export_for
mat=notebook --FilesWriter.build_directory=]
--clear-output
    Clear output of current file and save in place,
            overwriting the existing notebook.
    Equivalent to: [--NbConvertApp.use_output_suffix=False --NbConvertApp.export_for
mat=notebook --FilesWriter.build_directory= --ClearOutputPreprocessor.enabled=True]
--coalesce-streams
    Coalesce consecutive stdout and stderr outputs into one stream (within each cel
```

```
l).
    Equivalent to: [--NbConvertApp.use_output_suffix=False --NbConvertApp.export_for
mat=notebook --FilesWriter.build_directory= --CoalesceStreamsPreprocessor.enabled=Tr
ue]
--no-prompt
    Exclude input and output prompts from converted document.
    Equivalent to: [--TemplateExporter.exclude_input_prompt=True --TemplateExporter.
exclude_output_prompt=True]
--no-input
    Exclude input cells and output prompts from converted document.
            This mode is ideal for generating code-free reports.
    Equivalent to: [--TemplateExporter.exclude_output_prompt=True --TemplateExporte
r.exclude_input=True --TemplateExporter.exclude_input_prompt=True]
--allow-chromium-download
    Whether to allow downloading chromium if no suitable version is found on the sys
tem.
    Equivalent to: [--WebPDFExporter.allow_chromium_download=True]
--disable-chromium-sandbox
    Disable chromium security sandbox when converting to PDF..
    Equivalent to: [--WebPDFExporter.disable_sandbox=True]
--show-input
    Shows code input. This flag is only useful for dejavu users.
    Equivalent to: [--TemplateExporter.exclude_input=False]
--embed-images
    Embed the images as base64 dataurls in the output. This flag is only useful for
the HTML/WebPDF/Slides exports.
    Equivalent to: [--HTMLExporter.embed_images=True]
--sanitize-html
    Whether the HTML in Markdown cells and cell outputs should be sanitized..
    Equivalent to: [--HTMLExporter.sanitize_html=True]
--log-level=<Enum>
    Set the log level by value or name.
    Choices: any of [0, 10, 20, 30, 40, 50, 'DEBUG', 'INFO', 'WARN', 'ERROR', 'CRITI
CAL']
    Default: 30
    Equivalent to: [--Application.log_level]
--config=<Unicode>
    Full path of a config file.
    Default: ''
    Equivalent to: [--JupyterApp.config_file]
--to=<Unicode>
    The export format to be used, either one of the built-in formats
            ['asciidoc', 'custom', 'html', 'latex', 'markdown', 'notebook', 'pdf',
'python', 'qtpdf', 'qtpng', 'rst', 'script', 'slides', 'webpdf']
            or a dotted object name that represents the import path for an
            ``Exporter`` class
    Default: ''
    Equivalent to: [--NbConvertApp.export_format]
--template=<Unicode>
    Name of the template to use
    Default: ''
    Equivalent to: [--TemplateExporter.template_name]
--template-file=<Unicode>
    Name of the template file to use
    Default: None
    Equivalent to: [--TemplateExporter.template_file]
```

```
--theme=<Unicode>
    Template specific theme(e.g. the name of a JupyterLab CSS theme distributed
    as prebuilt extension for the lab template)
    Default: 'light'
    Equivalent to: [--HTMLExporter.theme]
--sanitize_html=<Bool>
    Whether the HTML in Markdown cells and cell outputs should be sanitized.This
    should be set to True by nbviewer or similar tools.
    Default: False
    Equivalent to: [--HTMLExporter.sanitize_html]
--writer=<DottedObjectName>
    Writer class used to write the
                                        results of the conversion
    Default: 'FilesWriter'
    Equivalent to: [--NbConvertApp.writer_class]
--post=<DottedOrNone>
    PostProcessor class used to write the
                                        results of the conversion
    Default: ''
    Equivalent to: [--NbConvertApp.postprocessor_class]
--output=<Unicode>
    Overwrite base name use for output files.
                Supports pattern replacements '{notebook_name}'.
    Default: '{notebook_name}'
    Equivalent to: [--NbConvertApp.output_base]
--output-dir=<Unicode>
    Directory to write output(s) to. Defaults
                                    to output to the directory of each notebook. To re
cover
                                    previous default behaviour (outputting to the curr
ent
                                    working directory) use . as the flag value.
    Default: ''
    Equivalent to: [--FilesWriter.build_directory]
--reveal-prefix=<Unicode>
    The URL prefix for reveal.js (version 3.x).
            This defaults to the reveal CDN, but can be any url pointing to a copy
            of reveal.js.
            For speaker notes to work, this must be a relative path to a local
            copy of reveal.js: e.g., "reveal.js".
            If a relative path is given, it must be a subdirectory of the
            current directory (from which the server is run).
            See the usage documentation
            (https://nbconvert.readthedocs.io/en/latest/usage.html#reveal-js-html-sl
ideshow)
            for more details.
    Default: ''
    Equivalent to: [--SlidesExporter.reveal_url_prefix]
--nbformat=<Enum>
    The nbformat version to write.
            Use this to downgrade notebooks.
    Choices: any of [1, 2, 3, 4]
    Default: 4
    Equivalent to: [--NotebookExporter.nbformat_version]

Examples
```

```
--------

    The simplest way to use nbconvert is

            > jupyter nbconvert mynotebook.ipynb --to html

            Options include ['asciidoc', 'custom', 'html', 'latex', 'markdown', 'not
ebook', 'pdf', 'python', 'qtpdf', 'qtpng', 'rst', 'script', 'slides', 'webpdf'].

            > jupyter nbconvert --to latex mynotebook.ipynb

            Both HTML and LaTeX support multiple output templates. LaTeX includes
            'base', 'article' and 'report'.  HTML includes 'basic', 'lab' and
            'classic'. You can specify the flavor of the format used.

            > jupyter nbconvert --to html --template lab mynotebook.ipynb

            You can also pipe the output to stdout, rather than a file

            > jupyter nbconvert mynotebook.ipynb --stdout

            PDF is generated via latex

            > jupyter nbconvert mynotebook.ipynb --to pdf

            You can get (and serve) a Reveal.js-powered slideshow

            > jupyter nbconvert myslides.ipynb --to slides --post serve

            Multiple notebooks can be given at the command line in a couple of
            different ways:

            > jupyter nbconvert notebook*.ipynb
            > jupyter nbconvert notebook1.ipynb notebook2.ipynb

            or you can specify the notebooks list in a config file, containing::

                c.NbConvertApp.notebooks = ["my_notebook.ipynb"]

            > jupyter nbconvert --config mycfg.py

To see all available configurables, use `--help-all`.
```