EJERCICIOS A PAPEL SW2

XML MAL FORMADOS https://github.com/mikimb99/Proyecto-SWII

EJEMPLO 1 RANDOM

```
<data>
  <person>
    <name>John Doe</name>
    <age>30
    <email>john.doe@example.com</email>
  </person>
  <product
    <name>Smartphone</brand>
    <price>999.99</price
  </product>
  <product>
    <name>Tablet</name>
    <brand>Apple</brand>
    <price>799.99<price>
  </product>
</data>
```

CORRECCIÓN

```
<data>
  <person>
    <name>John Doe</name>
    <age>30 </age>
    <email>john.doe@example.com</email>
  </person>
  <product name=”Smartphone”>
<brand> X</brand>
    <price>999.99</price>
  </product>
  <product>
    <name>Tablet</name>
    <brand>Apple</brand>
    <price>799.99<price>
  </product>
</data>
```

EJEMPLO 3 DIAP

MAL

<breakfast_menu>
<food>
<name>French Toast</name>
<name>French Toast</name>
<price currency="$">4.50</price>
<description>Thick slices made from our homemade sourdough
bread</DESCRIPTION>
<calories>600</calories>
</food>
<food>
<price currency>6.95</price>
<description>Two eggs, bacon or sausage, toast, and our ever-popular hash browns
<calories>950</calories>
</food>
<food></food>
</breakfastMenu>
<drinks_menu>
<drink>
<name>Coffe<name/>
<price>1.5$</price>
</drink></drinks_menu>

CORREGIDO

<menu>
<breakfast_menu>
<food>
<name>French Toast</name>
<name>French Toast</name>
<price currency="$">4.50</price>
<description>Thick slices made from our homemade sourdough bread</description>
<calories>600</calories>
</food>
<food>
<price currency="$">6.95</price>
<description>Two eggs, bacon or sausage, toast, and our ever-popular hash
browns</description>
<calories>950</calories>
</food>
<food></food>

```
</breakfast_menu>
<drinks_menu>
<drink>
<name>Coffe</name>
<price>1.5$</price>
</drink>
</drinks_menu>
</menu>
```

DTD A XML

EJ 1 DIAP

DTD

```
<!DOCTYPE TVSCHEDULE [

<!ELEMENT TVSCHEDULE (CHANNEL+)>
<!ELEMENT CHANNEL (BANNER,DAY+)>
<!ELEMENT BANNER (#PCDATA)>
<!ELEMENT DAY (DATE,(HOLIDAY|PROGRAMSLOT+)+)>
<!ELEMENT HOLIDAY (#PCDATA)>
<!ELEMENT DATE (#PCDATA)>
<!ELEMENT PROGRAMSLOT (TIME,TITLE,DESCRIPTION?)>
<!ELEMENT TIME (#PCDATA)>
<!ELEMENT TITLE (#PCDATA)>
<!ELEMENT DESCRIPTION (#PCDATA)>

<!ATTLIST TVSCHEDULE NAME CDATA #REQUIRED>
<!ATTLIST CHANNEL CHAN CDATA #REQUIRED>
<!ATTLIST PROGRAMSLOT VTR CDATA #IMPLIED>
<!ATTLIST TITLE RATING CDATA #IMPLIED>
<!ATTLIST TITLE LANGUAGE CDATA #IMPLIED>
]>
```

XML

```
<TVSCHEDULE NAME=” de”>
        <CHANNEL CHAN=”sdf ”>
                <BANNER>BAN</BANNER>
                <DAY>
                        <DATE>28</DATE>
```

```
                    <HOLIDAY>HOLI</HOLIDAY>
                    <PROGRAMSLOT VTR=”NO HACE FALTA”>
                        <TIME></TIME>
                        <TITLE></TITLE>
                    </PROGRAMSLOT>
                </DAY>

        </CHANNEL>
</TVSCHEDULE>
```

XML A DTD

XML

```
<articles>
<article id="x34675">
<name>Apache Spark Architecture</name>
<month>december</month>
<author name="kay vennisla"/>
<reviews lang=""/>
<feedback > high rating</feedback>
<reviews lang="de">The best content with diagrams</reviews>
</article>
</articles>
```

DTD

```
<!DOCTYPE articles[
<!ELEMENT articles(article +)>
<!ELEMENT article(name,month, author, (reviews,feedback?)* )>
<!ATTLIST article id CDATA #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT month(#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ATTLIST author name CDATA #REQUIRED>
<!ELEMENT reviews (#PCDATA)>
<!ATTLIST reviews lang CDATA #REQUIRED>
<!ELEMENT feedback(#PCDATA)>
]>
```

XML A XSD

XML

```
<items>
<item>
<name>Item 1</name>
<photo>http://example.com/photo1.png</photo>
<tags>Tag1, Tag2</tags>
<diameter>32</diameter>
<weight>540</weight>
<price>60</price>
<size>Big</size>
</item>
<item>
<name>Item 2</name>
<photo>http://example.com/photo2.jpg</photo>
<tags>Tag1</tags>
<diameter>23</diameter>
<weight>340</weight>
<price>50.1</price>
</item>
</items>
```

Tenga en cuenta que los únicos valores válidos de la etiqueta size, si aparece, son: • Big • Small

XSD

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name=”items”>
        <xs:complexType>
                <xs:sequence>
                        <xs:element name=”item” maxOccurs=”unbounded”>
                                <xs:complexType>
                                        <xs:sequence>
                                                <xs:element name=”name”
type=”xs:string”/>
                                                <xs:element name=”name”
type=”xs:string”/>
                                                <xs:element name=”photo”
                                                type=”xs:string”/>
```

```
                                        <xs:element name=”tags”
                                        type=”xs:string”/>
                                        <xs:element name=”diameter”
                                        type=”xs:decimal”/>
                                        <xs:element name=”weigth”
                                        type=”xs:decimal”/>
                                        <xs:element name=”price”
                                        type=”xs:decimal”/>
                                        <xs:element name=”size”
minOccurs=”0”>
                                        <xs:simpleType>
                                                <xs:restriction base=”xs:string”>
                                                        <xs:enumeration
                                        value=”Big”>
                                                        <xs:enumeration
                                        value=”Small”>
                                                <xs:restriction>
                                        <xs:simpleType>
                                        </xs:element>
                                                </xs:sequence>
                                                </xs:complexType>
                                               </xs:element>
                                              </xs:sequence>
                                            </xs:complexType>
                                          </xs:element>
                                        </xs:schema>
```

XSD A XML

XSD

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="Orderdetails">
<xs:complexType>
<xs:sequence>
<xs:element maxOccurs="unbounded" name="Customer">
<xs:complexType>
```

```
<xs:sequence>
<xs:element minOccurs="0" name="fname">
<xs:complexType>
<xs:simpleContent>
<xs:extension base="xs:string">
<xs:attribute name="from" type="xs:string" use="required" />
</xs:extension>
</xs:simpleContent>
</xs:complexType>
</xs:element>
<xs:element minOccurs="0" name="cname" type="xs:string" />
 <xs:element name="destination">
        <xs:complexType>
              <xs:simpleContent>
                    <xs:extension base="xs:string">
                          <xs:attribute name="Country" type="xs:string"
                    use="required" />
                          <xs:attribute name="Delivdate" type="xs:string"
                          use="required" />
                    </xs:extension>
              </xs:simpleContent>
        </xs:complexType>
</xs:element>
<xs:element minOccurs="0" name="email" type="xs:string" />
<xs:element minOccurs="0" name="eid" type="xs:string" />
        </xs:sequence>
        <xs:attribute name="id" type="xs:unsignedByte" use="required" />
        </xs:complexType>
</xs:element>
        </xs:sequence>
        </xs:complexType>
</xs:element>
</xs:schema>
```

XML

```
<Orderdetails>
        <Customer id=”13”>
              <fname from=”from”> f</fname>
              <cname> cname </cname>
              <destination Country=”country” Delivdate= “deliv” > destino
</destination>
              <email></email>
              <eid></eid>
```

</Customer>
<Customer>
<cname> cname </cname>
<destination Country=”country” Delivdate= “deliv”/>
<email></email>
</Customer>
</Orderdetails>

XPATH

EJEMPLOS

<?xml version="1.0" encoding="UTF-8"?>
<class>
<name>L.1.1.1</name>
<student id="035">
<name>Mark</name>
<year>1999</year>
</student>
<student id="007">
<name>Ana</name>
<year>1998</year>
</student>
</class>

• ¿La información de toda la clase?

/class

• ¿Todos los estudiantes?

//student ó /class/student

• ¿Todos los nombres de los estudiantes?

/class/student/name ó //student/name

Muestrame las ids de los estudiantes que tengan un id

//student[@id]/@id ó /class/student[@id]/@id

• ¿Todas las ids de los estudiantes?

//student/@id ó /class/student/@id

• ¿El segundo estudiante?

/class/student[2]

• ¿Estudiante con id 035?

//student[@id='035']

• ¿Estudiantes del año 1999 o posteriores?

//student[year>='1999']

• ¿Todos los elementos student que tengan al menos un atributo?

//student[@*]

¿Cuántos estudiantes hay?

count(/class/student) o count(//student)

• ¿Nombre de los estudiantes que han nacido en 1999?

//student[year='1999']/name ó //student[year='1999']/name/text()

• ¿Año de nacimiento de Ana?

//student[name='Ana']/year/text()

• ¿Último estudiante?

//student[last()]

Los id de lista que tengan @id >10

//student[@id >10]/@id

El id de los que se llamen Mark y sean del 1999

/class/student[name='Mark' and year=1999]/@id

PARA EL USO DE CONTAINS (en caso de que contenga la palabra Ana y haya un Ana y un Anacardo por ejemplo) y quiera el año o cualquier variable

//student[name[contains(text(),"Ana")]]/year

LISTA Y DESPUÉS VALOR A OBTENER

EJERCICIO 1

```
<?xml version="1.0" encoding="utf-8"?>
<Wikimedia>
<projects>
<project name="Wikipedia" launch="2001-01-05">
<editions>
<edition language="English">en.wikipedia.org</edition>
<edition language="German">de.wikipedia.org</edition>
<edition language="French">fr.wikipedia.org</edition>
<edition language="Polish">pl.wikipedia.org</edition>
<edition language="Spanish">es.wikipedia.org</edition></editions>
</project>
<project name="Wiktionary" launch="2002-12-12">
<editions>
<edition language="English">en.wiktionary.org</edition>
<edition language="French">fr.wiktionary.org</edition>
<edition language="Vietnamese">vi.wiktionary.org</edition>
<edition language="Turkish">tr.wiktionary.org</edition>
<edition language="Spanish">es.wiktionary.org</edition>
</editions>
</project>
</projects>
```

Haz las consultas XPath que cumplan lo siguiente y el resultado sea el que se muestre después:

• Los nombres de todos los proyectos:

Wikipedia
Wiktionary

Consulta→ //project/data(@name) Non-standard output

//project/@name

• Solo las URL de todos los proyectos en español

es.wikipedia.org
es.wiktionary.org

Consulta→ //project/editions/data(edition[@language='Spanish']) non-standard output ???

//edition[@language="Spanish"]

Todas las ediciones de todos los proyectos

<edition language="English">en.wikipedia.org</edition>
<edition language="German">de.wikipedia.org</edition>
<edition language="French">fr.wikipedia.org</edition>
<edition language="Polish">pl.wikipedia.org</edition>
<edition language="Spanish">es.wikipedia.org</edition>
<edition language="English">en.wiktionary.org</edition>
<edition language="French">fr.wiktionary.org</edition>
<edition language="Vietnamese">vi.wiktionary.org</edition>
<edition language="Turkish">tr.wiktionary.org</edition>
<edition language="Spanish">es.wiktionary.org</edition>

Consulta→//edition ó //project/editions/edition ó //project/editions/*

• Sólo las URL de todas las Wikipedias

en.wikipedia.org
de.wikipedia.org
fr.wikipedia.org
pl.wikipedia.org
es.wikipedia.org

Consulta→//project[@name='Wikipedia']

//project[@name="Wikipedia"]/editions/edition más especifico

• La cuarta edición del Wiktionary

<edition language="Turkish">tr.wiktionary.org</edition>

Consulta→//project[@name="Wiktionary"]/editions/edition[4]

XQUERY

```
> XQUERY doc("ejXquery.xml")/bookstore/book[price>30]/title
<title lang="en">XQuery Kick Start</title>
<title lang="en">Learning XML</title>
```

```
> XQUERY for $x in doc("ejXquery.xml")/bookstore/book where $x/price>30 return $x/title
<title lang="en">XQuery Kick Start</title>
<title lang="en">Learning XML</title>
```

```
> XQUERY for $x in doc("ejXquery.xml")/bookstore/book return if ($x/@category="CHILDREN") then <child>{data($x/title)}</child> else <adult>{d
ata($x/title)}</adult>
<adult>Everyday Italian</adult>
<child>Harry Potter</child>
<adult>XQuery Kick Start</adult>
<adult>Learning XML</adult>
```

```
> XQUERY for $x at $i in doc("ejXquery.xml")/bookstore/book/title return <book>{$i}. {data($x)}</book>
<book>1. Everyday Italian</book>
<book>2. Harry Potter</book>
<book>3. XQuery Kick Start</book>
<book>4. Learning XML</book>
```

```
> XQUERY for $act in doc("hamlet.xml")//ACT let $speakers := distinct-values($act//SPEAKER) return <div> <h1>{ string($act/TITLE) }</h1> <ul>
 {for $speaker in $speakers return <li>{ $speaker }</li> } </ul> </div>
<div><h1>ACT I</h1><ul><li>BERNARDO</li><li>FRANCISCO</li><li>HORATIO</li><li>MARCELLUS</li><li>KING CLAUDIUS</li><li>CORNELIUS</li><li>VOLTI
MAND</li><li>LAERTES</li><li>LORD POLONIUS</li><li>HAMLET</li><li>QUEEN GERTRUDE</li><li>All</li><li>OPHELIA</li><li>Ghost</li></ul></div>
<div><h1>ACT II</h1><ul><li>LORD POLONIUS</li><li>REYNALDO</li><li>OPHELIA</li><li>KING CLAUDIUS</li><li>QUEEN GERTRUDE</li><li>ROSENCRANTZ</
li><li>GUILDENSTERN</li><li>VOLTIMAND</li><li>HAMLET</li><li>First Player</li></ul></div>
<div><h1>ACT III</h1><ul><li>KING CLAUDIUS</li><li>ROSENCRANTZ</li><li>GUILDENSTERN</li><li>QUEEN GERTRUDE</li><li>LORD POLONIUS</li><li>OPHE
LIA</li><li>HAMLET</li><li>First Player</li><li>HORATIO</li><li>Prologue</li><li>Player King</li><li>Player Queen</li><li>LUCIANUS</li><li>Al
l</li><li>Ghost</li></ul></div>
<div><h1>ACT IV</h1><ul><li>KING CLAUDIUS</li><li>QUEEN GERTRUDE</li><li>HAMLET</li><li>ROSENCRANTZ</li><li>GUILDENSTERN</li><li>PRINCE FORTI
NBRAS</li><li>Captain</li><li>Gentleman</li><li>HORATIO</li><li>OPHELIA</li><li>LAERTES</li><li>Danes</li><li>Servant</li><li>First Sailor</l
i><li>Messenger</li></ul></div>
<div><h1>ACT V</h1><ul><li>First Clown</li><li>Second Clown</li><li>HAMLET</li><li>HORATIO</li><li>LAERTES</li><li>First Priest</li><li>QUEEN
 GERTRUDE</li><li>KING CLAUDIUS</li><li>All</li><li>OSRIC</li><li>Lord</li><li>PRINCE FORTINBRAS</li><li>First Ambassador</li></ul></div>
```

```
> XQUERY for $x in doc("ej1.xml")/bib/book order by xs:decimal($x/price) descending return $x/title
<title>The Economics of Technology and Content for Digital
TV</title>
<title>TCP/IP Illustrated</title>
<title>Advanced Programming in the Unix environment</title>
<title>Data on the Web</title>
<title>Principles of Databases</title>
Query "sweb2" executed in 20.72 ms.
```

```
> XQUERY count(doc("ej1.xml")/bib/book[author="Abiteboul"])
2
Query "sweb2" executed in 39.94 ms.
>
```

```
<authors>{
  for $x in distinct-values(doc("xquery_ejercicio1.xml")/bib/book/author)
  return <author>
    <name>{$x}</name>
    <count>{count(doc("xquery_ejercicio1.xml")/bib/book[author=$x])}</count>
  </author>
}</authors>
```

JSON

{ "nombre": "Juan", "telefonos": [{"movil": 612345678},
{"fijo": [912345678]}, false]
edad: 27,
mayorDeEdad: "true",
"direccion": { "calle": "Avenida Ciudad de Barcelona 23",
"ciudad": Madrid, null },
}

JSON CORREGIDO

{
"nombre": "Juan",
"telefonos": [{
"movil": 612345678
},
{
"fijo": [912345678]
}
],
"edad": 27,
"mayorDeEdad": true,
"direccion": {
"calle": "Avenida Ciudad de Barcelona 23",
"ciudad": "Madrid"
}
}

JSON A JSONSCHEMA

JSON

```
{
 "squadName": "Super hero squad",
```

```
 "homeTown": "Metro City",
 "formed": 2016,
 "secretBase": "Super tower",
 "active": true,
 "members": [
   {
     "name": "Molecule Man",
     "age": 29,
     "secretIdentity": "Dan Jukes",
     "powers": [
       "Radiation resistance",
       "Turning tiny",
       "Radiation blast"
     ]
   },
   {
     "name": "Madame Uppercut",
     "age": 39,
     "secretIdentity": "Jane Wilson",
     "powers": [
       "Million tonne punch",
       "Damage resistance"
     ]
   }
 ]
}
```

JSONSCHEMA

```
{
        "$schema": "https://json-schema.org/draft/2020-12/schema",
        "$id": "https://example.com/product.schema.json",
        “title”: “squad”,
        “description”: “Descripcion de la squad,
        “type”: “object”,
        “properties”: {
                “squadName”: {
                        “description”: “”, “type”:”string”}
                “hometown”: {
                        “description”: “”, “type”:”string”}
```

“formed”: {
“description”: “”, “type”:”integer”}
“secretBase”: {
“description”: “”, “type”:”string”}
“active”: {
“description”: “”, “type”:”boolean”}
“members”: {
“description”: “”, “type”:”array”,
“items”:{ “name”: {“type”:”string”}, “age”:
{“type”:”string”}}, required:[“name”, “age”...]
},

```
<articles>
 <article id="x34675">
 <name>Apache Spark Architecture</name>
 <month>december</month>
 <author name="kay vennisla"/>
 <reviews lang=""/>
 <feedback > high rating</feedback>
 <reviews lang="de">The best content with diagrams</reviews>
 </article>
</articles>
```

<! DOCTYPE articles[
<!ELEMENT articles (article+)>
<!ELEMENT article (name, month, author, (reviews, feedback?)*)>
<!ELEMENT name #PCDATA>
<!ELEMENT month #PCDATA>

<!ELEMENT author #PCDATA>

<!ELEMENT reviews #PCDATA>

<!ELEMENT feedback #PCDATA>

<!ATTLIST article id CDATA #REQUIRED>
<!ATTLIST author name CDATA #REQUIRED>
<!ATTLIST lang CDATA #REQUIRED>

]>

—----------

XML

```
<Orderdetails>
	<Customer id=”13”>
		<fname from=”from”> f</fname>
		<cname> cname </cname>
		<destination Country=”country” Delivdate= “deliv” > destino
</destination>
		<email></email>
		<eid></eid>
	</Customer>
	<Customer>
		<cname> cname </cname>
			<destination Country=”country” Delivdate= “deliv”/>
			<email></email>
	</Customer>
</Orderdetails>
```

XSD del ejercicio2.xml

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name= “Orderdetails”>
	<xs:complexType>
	<xs:sequence>
		<xs:element name=”Customer” maxOccurs=”unbounded”>
		<xs:complexType>
		<xs:sequence>
			<xs:element name=”fname” minOccurs=”0”>
			<xs:simpleContent>
				<xs:extension base=”xs:string”>
					<xs:attribute name=”from” type:”xs:string”
use=”required”/ >
				</xs:extension>
			</xs:simpleContent>
		</xs:complexType>
		</xs:sequence>
		<xs:element name=”cname” minOccurs=”1”/>
		<xs:element name=”destination minOccurs=”1”>
		<xs:complexType>
```

```
                    <xs:simpleContent>
                        <xs:extension base=”xs:string”>
                            <xs:attribute name=”Country” type:”xs:string”
use=”required”/ >
                            <xs:attribute name=”Delivdate”
type:”xs:string” use=”required”/ >
                        </xs:extension>
                    </xs:simpleContent>
                </xs:complexType>
                </xs:element>
                <xs:element name=”email” minOccurs=”1” “type:xs:string” />
                <xs:element name=”eid” minOccurs=”0” “type:xs:string” />
                </xs:sequence>
                    <xs:attribute name=”id” type:”xs:unsignedByte”
use:”required”/>

                </xs:element>
        </xs:sequence>
        </xs:complexType>
</xs:element>
</xs:schema>
```

MONGODB

1. En sample_training.zips ¿Cuántas colecciones tienen menos de 1000 personas en el campo pop? (sol. 8065)

db.zips.find({pop:{$lt:1000}}).count()

2. En sample_training.trips ¿Cuál es la diferencia entre la gente que nació en 1998 y la que nació después de 1998? (sol. 6)

```
Math.abs(db.find({"birth year": {$lt:
1998}})).count()-(db.find({"birth year": {$gt:
1998}})).count()
```

3. En sample_training.routes ¿Cuántas rutas tienen al menos una parada? (sol. 11)

```
db.find({"stops": {"$gt": 0 }}).count()
```

4. En sample_training.inspections ¿Cuántos negocios tienen un resultado de inspección "Out of Business" y pertenecen al sector "Home Improvement Contractor - 100"? (sol. 4)

```
db.find({
$and: [
   { "result":{$eq:  "Out of Business"}},
   { "sector": {$eq: "Home Improvement Contractor - 100"}}
]
}).count();
```

ó

db.inspections.find({result:"Out of Business", sector:"Home Improvement Contractor - 100"}).count()

5. En sample_training.inspections ¿Cuántos documentos hay con fecha de inspección "Feb 20 2015" o "Feb 21 2015" y cuyo sector no sea "Cigarette Retail Dealer - 127"? (sol. 204)

```
db.find({ "$and": [
   { "$or" : [ {date: "Feb 20 2015"} , {date: "Feb 21
2015"}]},
   { sector: {"$ne": "Cigarette Retail Dealer - 127"}}
]}).count()
```

Ó

db.inspections.find({$or:[{date:"Feb 20 2015"},{date:"Feb 21 2015"}],sector:{$ne:"Cigarette Retail Dealer - 127"}}).count()

PARA COMPARAR 2 VARIABLES CON EXPR o CUANDO USES DOS

db.companies.find({$expr:{$gt:["$number_of_employees","$founded_year"]}}).count()

3. En sample_airbnb.listingsAndReviews, ¿cuál es el nombre del alojamiento en el que pueden estar más de 6 personas alojadas y tiene exactamente 50 reviews? (sol. Sunset Beach Lodge Retreat)

```
db.find({ "$expr":
{"$and":
   [ { "$gt": ["$accommodates", 6]},
     {"$eq": ["$number_of_reviews",
50]}]}},{projection: {_id:0,name:1}}).toArray()
```

Ó

db.listingsAndReviews.find({$and:[{accommodates:{$gt:6}},{number_of_reviews:{$eq:50}}]},{name:1,_id:0})

En sample_airbnb.listingsAndReviews, ¿cuántos documentos tienen el "property_type" "House" e incluyen "Changing table" como una de las "amenities"? (sol. 11)

({property_type:"House",amenities:"Changing table"}).count()

6. En sample_training.companies, haga una query que devuelve únicamente el nombre de las empresas que tengan exactamente 8 "funding_rounds"

TAMAÑO DE ARRAY CON SIZE

db.companies.find({funding_rounds:{$size:8}},{name:1, _id:0})

7. En sample_training.trips, ¿cuántos viajes empiezan en estaciones que están al oeste de la longitud -74? (sol. 1928)

PARA COMPARAR UNO DE LOS CAMPOS DE UN ARRAY

db.trips.find({'start station location.coordinates.0':{$lt:-74}}).count()

3. En sample_training.trips, ¿en qué año nació el ciclista más joven? (sol. 1999)

```
db.find({"birth year":{$ne: ""}},{ projection: {_id:0,"birth year":1}}).sort({"birth year": -1}).limit(1).toArray()
```

SIMULACRO

MONGODB

En la colección listingAndReviews indique el/los nombre(s) del alojamiento con más reviews.

```
find({}, { projection:{ _id:0, name:1}}).sort({"number_of_reviews": 1}).limit(1).toArray()
```

En la colección listingAndReviews indique el/los nombre(s) del alojamiento con más amenities.

```
db.aggregate([
 {
   $project: {
     name: 1,
     numAmenities: { $size: "$amenities" }
   }
 },
 {
   $sort: { numAmenities: -1 }
 },
 {
   $limit: 1
 }
]).toArray()
```

En la colección listingAndReviews indique el número de alojamientos que tienen 2, 3, 4 o 5 beds

```
aggregate([
 {
   $group: {
     _id: "$property_type",
     count: { $sum: 1 }
   }
 },
 {
```

```
    $project: {
        _id: 0,
        property_type: "$_id",
        count: 1
    }
  }
]).toArray()
```

En la colección listingAndReviews indique el número de alojamientos que tienen 2, 3, 4 o 5 beds

```
db.find({"$or":[
        {"beds":2},
        { "beds":3},
        { "beds":4},
        { "beds":5}]}).count()
```

BÚSQUEDA DE ARRAY CON ELEMENTOS CONCRETOS

Book.find({ author: { $all: ['author1', 'author2'], $size: 2 } })

esta consulta buscará documentos en la colección "Book" donde el campo "author" contenga tanto "author1" como "author2" y tenga un tamaño de 2 elementos.

CONSULTAS Listing And Reviews:

https://www.w3resource.com/mongodb-exercises/mongodb-listingsandreviews-collection-index.php

https://gist.github.com/iamramann/4838049b3faeb37e45482fed0a11aa61

https://www.mongodb.com/docs/manual/reference/operator/aggregation/group/

Repo práctica:

https://github.com/mikimb99/Proyecto-SWII.git

Repo Alvaro:

https://bitbucket.org/alvarosp_ceu/sw2_2023_repository/src/master/simulacro/api/routes/

mongosh

use database

db.collection.find() acuérdate de pegarla así en el archivo!

PREGUNTAS

- **Diferencias XSD para restriction o extension:**
  - **Extension→ cuando tengo algún atributo aparte del element**
  - **Restriction→ solo quiero esos valores para el elemento**