

- **Mostrar los títulos de los libros con la etiqueta "titulo".**

```
for $tit in /bookstore/book
```

```
return <titulo>{data($tit/title)}</titulo>
```

The screenshot shows the DAW-20-2 interface. The query editor contains the following XQuery:

```
1 for $tit in /bookstore/book
2 return <titulo>{data($tit/title)}</titulo>
```

The results pane displays a table with the following data:

book title	author	book title	author	book title	author	book title	author
Everyday Italian	Giada De Laurentiis	Harry Potter	J K. Rowling	XQuery Kick Start	James Linn	Learning XML	Erik T. Ray
year	price	year	price	author	author	year	price
2005	30.00	2005	29.99	Per Both...	Kurt Cagle	2003	49.99
						2003	39.95

The result pane shows the XML output:

```
<titulo>Everyday Italian</titulo>
<titulo>Harry Potter</titulo>
<titulo>XQuery Kick Start</titulo>
<titulo>Learning XML</titulo>
```

Compiling: - pre-evaluate root to document-node(): root() -> db:open-pre("doc2", 0)

Optimized Query: for \$tit\_0 in db:open-pre("doc2", 0)/\*/bookstore/\*:book return element titulo { (data(\$tit\_0/\*:title)) }

- **Mostrar los libros cuyo precio sea menor o igual a 30. Primero incluyendo la condición en la cláusula "where" y luego en la ruta del Xpath.**

```
Where; for $lib in /bookstore/book
```

```
let $p := ($lib/price)
```

```
where $p <= 30
```

```
return $lib
```

The screenshot shows the DAW-20-2 interface. The query editor contains the following XQuery:

```
1 for $lib in /bookstore/book
2 let $p := ($lib/price)
3 where $p <= 30
4 return $lib
```

The results pane displays a table with the following data:

book title	author	book title	author	book title	author	book title	author
Everyday Italian	Giada De Laurentiis	Harry Potter	J K. Rowling	XQuery Kick Start	James Linn	Learning XML	Erik T. Ray
year	price	year	price	author	author	year	price
2005	30.00	2005	29.99	Per Both...	Kurt Cagle	2003	49.99
						2003	39.95

The result pane shows the XML output:

```
<book category="COOKING">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year>
  <price>30.00</price>
</book>
<book category="CHILDREN">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
```

Compiling: - pre-evaluate root to document-node(): root() -> db:open-pre("doc2", 0)

- rewrite <= operator to range comparison: (\$p\_1 <= 30) -> (\$p\_1 <= 30.0)

- inline \$p\_1

- rewrite where clause(s)

- simplify glwfor

Optimized Query: db:open-pre("doc2", 0)/\*/bookstore/\*:book[(\*:price <= 30.0)]

Query: for \$lib in /bookstore/book let \$p := (\$lib/price) where \$p <= 30 return \$lib

Result:

Xpath:

```
for $lib in /bookstore/book
```

```
return $lib[price<=30]
```

mentos/DAW-20-2

\*.xml, \*.xq\*

Find contents...

xml

- Bases
  - doc1.xml (624 b)
  - doc2.xml (859 b)
  - ejemplo\_usandoxsd.xsd
  - ejemplo\_xsd.xsd (623 b)
  - mensaje (1623 b)
  - taller.xml (2305 b)

OK

4 : 12

```

1 for $lib in /bookstore/book
2 let $p := ($lib/price)
3 where $p <= 30
4 return $lib

```

doc2.xml

bookstore		book		book		book		book	
title	auth..	title	auth..	title	auth..	author	author	author	author
Everyday Italian	Giada De Lau..	Harry Potter	J K. Row..	XQuery Kick Start	Ja.. Mc..	James Linn	Vai.. Nag..	Learning XML	Erik T. Ray
year	price	year	price	author	author	year	price	year	price
2005	30.00	2005	29.99	Per Both..	Kurt Cagle	2003	49.99	2003	39.95

Result

```

<book category="COOKING">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year>
  <price>30.00</price>
</book>
<book category="CHILDREN">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>

```

Compiling:

- pre-evaluate root to document-node(): root() -> db:open-pre("doc2", 0)
- rewrite <= operator to range comparison: (\$p\_1 <= 30) -> (\$p\_1 <= 30.0)
- inline \$p\_1
- rewrite where clause(s)
- simplify gfiwor

Optimized Query:

```
db:open-pre("doc2", 0)/*:bookstore/*:book[(*:price <= 30.0)]
```

Query:

```
for $lib in /bookstore/book let $p := ($lib/price) where $p <= 30 return $lib
```

Result:

Total Time: 4.63 ms

- Mostrar sólo el título de los libros cuyo precio sea menor o igual a 30.

```

for $lib in /bookstore/book
where $lib/price<=30
return ($lib/title)

```

mentos/DAW-20-2

\*.xml, \*.xq\*

Find contents...

xml

- Bases
  - doc1.xml (624 b)
  - doc2.xml (859 b)
  - ejemplo\_usandoxsd.xsd
  - ejemplo\_xsd.xsd (623 b)
  - mensaje (1623 b)
  - taller.xml (2305 b)

OK

1 : 1

```

1 for $lib in /bookstore/book
2 where $lib/price<=30
3 return ($lib/title)

```

doc2.xml

bookstore		book		book		book		book	
title	auth..	title	auth..	title	auth..	author	author	author	author
Everyday Italian	Giada De Lau..	Harry Potter	J K. Row..	XQuery Kick Start	Ja.. Mc..	James Linn	Vai.. Nag..	Learning XML	Erik T. Ray
year	price	year	price	author	author	year	price	year	price
2005	30.00	2005	29.99	Per Both..	Kurt Cagle	2003	49.99	2003	39.95

Result

```

<title lang="en">Everyday Italian</title>
<title lang="en">Harry Potter</title>

```

Compiling:

- pre-evaluate root to document-node(): root() -> db:open-pre("doc2", 0)
- rewrite <= operator to range comparison: (\$lib\_0/\*:price <= 30) -> (\$lib\_0/\*:price <= 30.0)
- rewrite where clause(s)

Optimized Query:

```
for $lib_0 in db:open-pre("doc2", 0)/*:bookstore/*:book[(*:price <= 30.0)] return $lib_0/*:title
```

Info

Total Time: 4.2 ms

- Mostrar sólo el título sin atributos de los libros cuyo precio sea menor o igual a 30.

```

for $lib in /bookstore/book
where $lib/price<=30
return data($lib/title)

```

doc2.xml bookstore

book title	auth..	book title	auth..	book title	auth..	author	author	book title	author
Everyday Italian	Giada De Lau..	Harry Potter	J K. Row..	XQuery Kick Start	Ja..	James Linn	Vai..	Learning XML	Erik T. Ray
year	price	year	price	author	author	year	price	year	price
2005	30.00	2005	29.99	Per Both..	Kurt Cagle	2003	49.99	2003	39.95

Result: Everyday Italian, Harry Potter

Compiling: - pre-evaluate root to document-node(): root() -> db:open-pre("doc2", 0)  
- rewrite <= operator to range comparison: (\$lib 0/\*:price <= 30) -> (\$lib 0/\*:price <= 30)

- **Mostrar el título y el autor de los libros del año 2005, y etiquetar cada uno de ellos con "lib2005".**

```
for $lib in /bookstore/book
where $lib/year=2005
return <lib2005>
{data($lib/title)}, {data($lib/author)}
</lib2005>
```

doc2.xml bookstore

book title	auth..	book title	auth..	book title	auth..	author	author	book title	author
Everyday Italian	Giada De Lau..	Harry Potter	J K. Row..	XQuery Kick Start	Ja..	James Linn	Vai..	Learning XML	Erik T. Ray
year	price	year	price	author	author	year	price	year	price
2005	30.00	2005	29.99	Per Both..	Kurt Cagle	2003	49.99	2003	39.95

Result: <lib2005>Everyday Italian, Giada De Laurentiis</lib2005>  
<lib2005>Harry Potter, J K. Rowling</lib2005>

Compiling: - pre-evaluate root to document-node(): root() -> db:open-pre("doc2", 0)  
- rewrite = operator to range comparison: (\$lib 0/\*:year = 2005) -> (\$lib 0/\*:year = 2005)

- **Mostrar los años de publicación, primero con "for" y luego con "let" para comprobar la diferencia entre ellos. Etiquetar la salida con "publicacion".**

For:

```
for $lib in /bookstore/book
return <publicacion>
{data($lib/year)}
</publicacion>
```

mentos/DAW-20-2

doc2.xml

```

1 for $lib in /bookstore/book
2 return <publicacion>
3   {data($lib/year)}
4 </publicacion>

```

Result

```

<publicacion>2005</publicacion>
<publicacion>2005</publicacion>
<publicacion>2003</publicacion>
<publicacion>2003</publicacion>

```

doc2.xml

book title	auth..	book title	auth..	book title	auth..	author	author	book title	author
Everyday Italian	Giada De Lau..	Harry Potter	J K. Row..	XQuery Kick Start	Ja.. Mc..	James Linn	Vai.. Nag..	Learning XML	Erik T. Ray
year	price	year	price	author	author	year	price	year	price
2005	30.00	2005	29.99	Per Both..	Kurt Cagle	2003	49.99	2003	39.95

Info

Compiling:

- pre-evaluate root to document-node(): root() -> db:open-pre("doc2", 0)

Optimized Query:

for \$lib\_0 in db:open-pre("doc2", 0)/\*:bookstore/\*:book return element publicacion { {

let:

```

let $lib := /bookstore/book
return <publicacion>
  {data($lib/year)}
</publicacion>

```

mentos/DAW-20-2

doc2.xml

```

1 let $lib := /bookstore/book
2 return <publicacion>
3   {data($lib/year)}
4 </publicacion>

```

Result

```

<publicacion>2005 2005 2003 2003</publicacion>

```

doc2.xml

book title	auth..	book title	auth..	book title	auth..	author	author	book title	author
Everyday Italian	Giada De Lau..	Harry Potter	J K. Row..	XQuery Kick Start	Ja.. Mc..	James Linn	Vai.. Nag..	Learning XML	Erik T. Ray
year	price	year	price	author	author	year	price	year	price
2005	30.00	2005	29.99	Per Both..	Kurt Cagle	2003	49.99	2003	39.95

Info

Compiling:

- pre-evaluate root to document-node(): root() -> db:open-pre("doc2", 0)
- inline \$lib\_0
- simplify gñwor

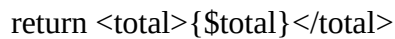
- **Mostrar los libros ordenados primero por "category" y luego por "title" en una sola consulta.**

```

let $cat := (
  for $c in /bookstore/book
  let $ca := data($c/@category)
  order by $c/@category
  return ($ca, data($c/title))
), $tit := (
  for $t in /bookstore/book
  order by $t/title
  return data($t/title)
)

```

&lt;/libro&gt;



```

<total>

{

  let $total := count(/bookstore/book)

  return $total

}

</total>

</libros>

```

The screenshot shows the mentos/DAW-20-2 interface. On the left, a file explorer shows a project named 'Bases' containing several XML and XSD files. The main editor displays an XQuery script for 'doc2.xml'. The script defines a function to count books and returns a list of book titles. Below the editor, a table displays the results of the query, showing book titles, authors, and years. The bottom pane shows the 'Result' of the query, which is a list of book titles. The 'Info' pane on the right provides details about the query execution, including the total time and the optimized query.

book title	author	year
Everyday Italian	Giada De Laurentiis	2005
XQuery Kick Start	Per Bothner	2003
Learning XML	Erik T. Ray	2003
Harry Potter	J. K. Rowling	2005
James Linn	James Linn	2003

- **Mostrar el precio mínimo y máximo de los libros.**

```

let $min := min(/bookstore/book/price), $max := max(/bookstore/book/price)

return

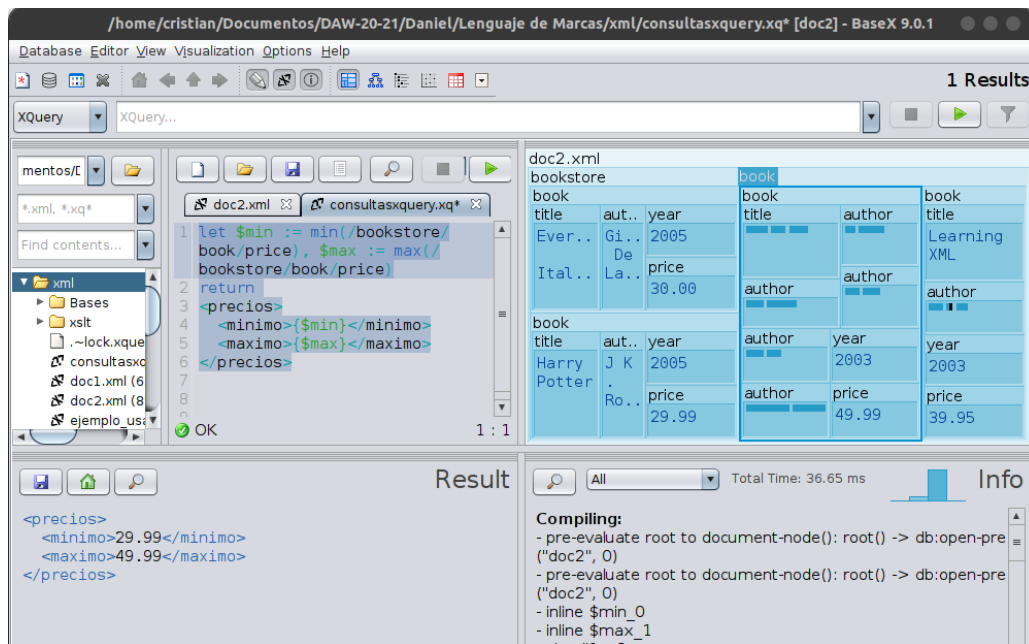
<precios>

  <minimo>{$min}</minimo>

  <maximo>{$max}</maximo>

</precios>

```



- **Mostrar el título del libro, su precio y su precio con el IVA incluido, cada uno con su propia etiqueta. Ordénalos por precio con IVA.**

for \$lib in /bookstore/book

let \$p := (\$lib/price \* 1.04)

return

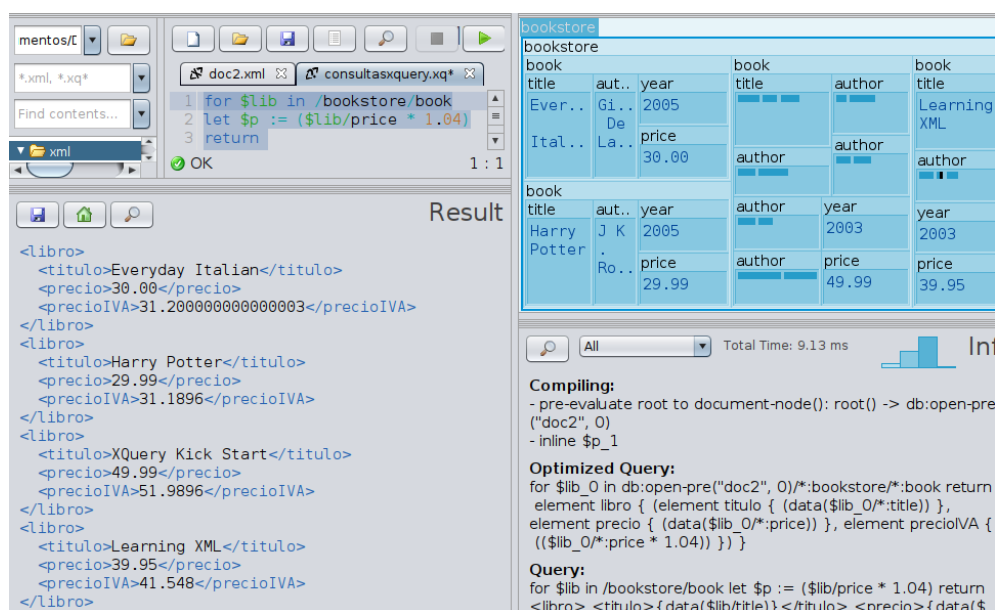
<libro>

<titulo>{data(\$lib/title)}</titulo>

<precio>{data(\$lib/price)}</precio>

<precioIVA>{\$p}</precioIVA>

</libro>



- **Mostrar la suma total de los precios de los libros con la etiqueta "total".**

```
let $total := sum(/bookstore/book/price)
```

```
return
```

```
<total>{$total}</total>
```

The screenshot shows a software interface with a file explorer on the left, a central editor, and a results pane on the right.

**File Explorer:** Shows a directory structure with folders like 'Bases', 'xslt', and files like 'doc2.xml', 'ejemplo\_usa', 'ejemplo\_xsc', and 'mensaje (1)'.

**Editor:** Contains the following XSLT code:

```
1 let $total := sum(/bookstore/
2 book/price)
3 return
4 <total>{$total}</total>
5
6
```

**Results Pane:** Displays a table of books from 'doc2.xml'.

book title	author	year	price
Ever..	Gi.. De	2005	30.00
Ital..	La..		
Harry Potter	J K Ro..	2005	29.99
		2003	49.99
			39.95

Below the table, the result is shown as: `<total>149.93</total>`

**Info Panel:** Shows 'Total Time: 3.46 ms' and 'Compiling:' information.

- **Mostrar cada uno de los precios de los libros, y al final una nueva etiqueta con la suma de los precios.**

```
<libro>
```

```
<precio>
```

```
{
```

```
for $lib in /bookstore/book
```

```
return
```

```
<p>{data($lib/price)}</p>
```

```
}
```

```
</precio>
```

```
<total>
```

```
{let $total := sum(/bookstore/book/price)
```

```
return $total
```

```
}
```

```
</total>
```

```
</libro>
```



mentos/DAW-20-21/Dan

\*.xml, \*.xq\*

Find contents...

xml

- Bases
- xslt
  - lock.xquery.odt# (82 b)
  - consultasxquery.xq (217 b)
  - doc1.xml (624 b)
  - doc2.xml (859 b)
  - ejemplo\_usandoxsd.xml (2)
  - ejemplo\_xsd.xsd (623 b)
  - mensaje (1623 b)
  - taller.xml (2305 b)
  - taller\_dtd.pdf (137 kB)
  - ticket\_dtd.pdf (152 kB)
  - ticket.pdf (111 kB)
  - ticket.xml (3248 b)
  - ticket\_doc.odt (128 kB)
  - xml\_valido.pdf (246 kB)

OK

1 : 1

Editor

```

1 <libro>
2 <precio>
3 {
4   for $lib in /bookstore/book
5   return
6     <p>{data($lib/price)}</p>
7 }
8 </precio>
9 <total>
10 {let $total := sum(/bookstore/book/price)
11  return $total
12 }
13 </total>
14 </libro>
15
16

```

doc2.xml

bookstore		book		book		book	
title	author	title	author	title	author	title	author
Everyday Italian	Giada De Laurentiis	Harry Potter	J K. Rowling	XQuery Kick Start	James McGovern	Learning XML	Kurt Cagle
year	price	year	price	author	author	year	price
2005	30.00	2005	29.99	James Linn	Vaidy Nagarajan	2003	49.99

Total Time: 4.66 ms

**Compiling:**

- pre-evaluate root to document-node(): root() -> c
- pre-evaluate root to document-node(): root() -> c
- inline \$total\_1
- simplify gfiwor

**Optimized Query:**

element libro { (element precio { (for \$lib\_0 in db:open-pre("doc2", 0) return element p { (data(\$lib\_0/price)) }}), element total {let \$total := sum(/bookstore/book/price) return \$total}) }

**Query:**

<libro> <precio> { for \$lib in /bookstore/ book return {let \$total := sum(/bookstore/book/price) return \$total} }

**Result:**

- Hit(s): 1 item
- Updated: 0 items

Result

```

<libro>
  <precio>
    <p>30.00</p>
    <p>29.99</p>
    <p>49.99</p>
    <p>39.95</p>
  </precio>
  <total>149.93</total>
</libro>

```

- **Mostrar el título y el número de autores que tiene cada título en etiquetas diferentes.**

for \$lib in /bookstore/book

let \$a := count(\$lib/author)

return

<libro>

<titulo>{data(\$lib/title)}</titulo>

<autores>{data(\$a)}</autores>

</libro>

mentos/DAW-20-21/Dan

\*.xml, \*.xq\*

Find contents...

xml

- Bases
- xslt
  - lock.xquery.odt# (82 b)
  - consultasxquery.xq (217 b)
  - doc1.xml (624 b)
  - doc2.xml (859 b)
  - ejemplo\_usandoxsd.xml (2)
  - ejemplo\_xsd.xsd (623 b)
  - mensaje (1623 b)
  - taller.xml (2305 b)
  - taller\_dtd.pdf (137 kB)
  - ticket\_dtd.pdf (152 kB)
  - ticket.pdf (111 kB)
  - ticket.xml (3248 b)
  - ticket\_doc.odt (128 kB)
  - xml\_valido.pdf (246 kB)

OK

9 : 1

Editor

```

1 for $lib in /bookstore/book
2 let $a := count($lib/author)
3 return
4   <libro>
5     <titulo>{data($lib/title)}</titulo>
6     <autores>{data($a)}</autores>
7   </libro>
8
9

```

doc2.xml

bookstore		book		book		book	
title	author	title	author	title	author	title	author
Everyday Italian	Giada De Laurentiis	Harry Potter	J K. Rowling	XQuery Kick Start	James McGovern	Learning XML	Kurt Cagle
year	price	year	price	author	author	year	price
2005	30.00	2005	29.99	James Linn	Vaidy Nagarajan	2003	49.99

Total Time: 1.93 ms

**Compiling:**

- pre-evaluate root to document-node(): root() -> db:open-pre("doc2", 0)
- rewrite fn:data([items]) to variable: data(\$a\_1) -> \$a\_1
- inline \$a\_1

**Optimized Query:**

for \$lib\_0 in db:open-pre("doc2", 0)/\*:bookstore/\*:book return element libro { (element titulo { (data(\$lib\_0/title)) }, element autores { (count(\$lib\_0/\*:author)) }) }

**Query:**

for \$lib in /bookstore/book let \$a := count(\$lib/author) return <libro> <titulo>{data(\$lib/title)} <autores>{data(\$a)}</autores> </libro>

**Result:**

- Hit(s): 4 items
- Updated: 0 items

Result

```

<libro>
  <titulo>Everyday Italian</titulo>
  <autores>1</autores>
</libro>
<libro>
  <titulo>Harry Potter</titulo>
  <autores>1</autores>
</libro>
<libro>
  <titulo>XQuery Kick Start</titulo>
  <autores>5</autores>
</libro>
<libro>
  <titulo>Learning XML</titulo>
  <autores>1</autores>
</libro>

```

- **Mostrar en la misma etiqueta el título y entre paréntesis el número de autores que tiene ese título.**

```
for $lib in /bookstore/book
```

```
let $a := count($lib/author)
```

```
return
```

```
<titulo>{data($lib/title)}, ({ $a})</titulo>
```

The screenshot shows the DAW-20-21 XML editor interface. The left pane displays a file explorer with a folder named 'xml' containing various files. The main editor pane shows an XQuery query with line numbers 1 through 8. The query is as follows:

```
1 for $lib in /bookstore/book
2 let $a := count($lib/author)
3 return
4 <titulo>{data($lib/title)}, ({ $a})</titulo>
```

The right pane shows the XML document 'doc2.xml' with a tree view of the 'bookstore' element. The 'book' element is expanded, showing a table with columns 'title' and 'author'. The data is as follows:

title	author
Everyday Italian	Giada De Laurentiis

Below the tree view, a table shows the 'year' and 'price' for the book 'Everyday Italian':

year	price
2005	30.00

The bottom pane shows the 'Result' of the query, which is an XML fragment:

```
<titulo>Everyday Italian, (1)</titulo>
<titulo>Harry Potter, (1)</titulo>
<titulo>XQuery Kick Start, (5)</titulo>
<titulo>Learning XML, (1)</titulo>
```

On the far right, there is a 'Compiling' section with the following information:

**Compiling:**

- pre-evaluate root to
- inline \$a\_1

**Optimized Query:**

```
for $lib_0 in db:open
, count($lib_0/*:auth
```

**Query:**

```
for $lib in /bookstore/book
let $a := count($lib/author)
return
<titulo>{data($lib/title)}, ({ $a})</titulo>
```

- **Mostrar los libros escritos en años que terminen en "3".**

```
for $lib in /bookstore/book
```

```
where ends-with($lib/year, "3")
```

```
return
```

```
<titulo>{data($lib/title)}</titulo>
```

The screenshot shows the XQuery IDE with a query in the Editor pane and its results in the Result pane.

**Editor:**

```

1 for $lib in /bookstore/book
2 where ends-with($lib/year, "3")
3 return
4 <titulo>{data($lib/title)}</titulo>

```

**Result:**

bookstore		book		book		book	
title	author	title	author	title	author	title	author
Everyday Italian	Giada De Laur..	Harry Potter	J K. Rowl..	XQuery Kick Start			
year	price	year	price	author			
2005	30.00	2005	29.99	James Linn			

**Compiling:**

- pre-evaluate root to document-node(): root() -> db:doc
- rewrite where clause(s)

- **Mostrar los libros cuya categoría empiece por "C".**

```

for $lib in /bookstore/book
where starts-with($lib/@category, "C")
return
<titulo>{data($lib/title)}</titulo>

```

The screenshot shows the XQuery IDE with a query in the Editor pane and its results in the Result pane.

**Editor:**

```

1 for $lib in /bookstore/book
2 where starts-with($lib/@category, "C")
3 return
4 <titulo>{data($lib/title)}</titulo>

```

**Result:**

bookstore		book		book	
title	author	title	author	title	author
Everyday Italian	Giada De Laur..	Harry Potter	J K. Rowl..		
year	price	year	price		
2005	30.00	2005	29.99		

**Compiling:**

- pre-evaluate root to document-node(): root() -> db:doc
- rewrite where clause(s)

**Optimized Query:**

```

for $lib_0 in db:open-pre("doc2", 0)/*/:bookstore
return { (data($lib_0/*:title)) }

```

- **Mostrar los libros que tengan una "X" mayúscula o minúscula en el título ordenados de manera descendente.**

```

for $lib in /bookstore/book
where contains($lib/title, "X") or contains($lib/title, "x")
return
<titulo>{data($lib/title)}</titulo>

```

The screenshot shows the XQuery IDE with a query in the Editor and its result in the Result pane.

**Editor:**

```

1 for $lib in /bookstore/book
2 where contains($lib/title, "X") or contains($lib/title, "x")
3 return
4 <titulo>{data($lib/title)}</titulo>
5
6
7
8

```

**Result:**

book title	author	book title	author	book title	author	author
Everyday Italian	Giada De Laurentiis	Harry Potter	J. K. Rowling	XQuery Kick Start	James McGovern	Per Bothner
						author Kurt Cagle
year	price	year	price	author	author	year
2005	30.00	2005	29.99	James Linn	Vaidy Nagar	2003
						price
						49.9

**Compiling:**

- pre-evaluate root to document-node(): root() -> db:open-pre("doc2", 0)
- rewrite where clause(s)

**Optimized Query:**

```

for $lib_0 in db:open-pre("doc2", 0)/*:bookstore/*:book[ (contains(*:title, "X") or
return element titulo { (data($lib_0/*:title)) }

```

- **Mostrar el título y el número de caracteres que tiene cada título, cada uno con su propia etiqueta.**

for \$lib in /bookstore/book

let \$scar := string-length(\$lib/title)

return

<libro>

<titulo>{data(\$lib/title)}</titulo>

<caracteres>{\$scar}</caracteres>

</libro>

The screenshot shows the XQuery IDE with a query in the Editor and its result in the Result pane.

**Editor:**

```

1 for $lib in /bookstore/book
2 let $scar := string-length($lib/title)
3 return
4 <libro>
5   <titulo>{data($lib/title)}</titulo>
6   <caracteres>{$scar}</caracteres>
7 </libro>
8
9
10
11

```

**Result:**

book title	author	book title
Everyday Italian	Giada De Laurentiis	Harry Potter
year	price	year
2005	30.00	2005

**Compiling:**

- pre-evaluate root to document-n
- inline \$scar\_1

**Optimized Query:**

```

for $lib_0 in db:open-pre("doc2", 0)/*:bookstore/*:book[ (contains(*:title), C
lib_0/*:title)) } , element caractere

```

**Query:**

```

for $lib in /bookstore/book let $scar
titulo> <caracteres>{ $scar}</car

```

**Result:**

- Hit(s): 4 Items
- Updated: 0 Items
- Printed: 324 b
- Read Locking: doc2

- **Mostrar todos los años en los que se ha publicado un libro eliminando los repetidos. Etiquétalos con "año".**

```
for $lib in distinct-values(/bookstore/book/year)
```

```
return <year>{$lib}</year>
```

The screenshot shows a software interface with three main panels:

- Left Panel (File Explorer):** Displays a directory structure with folders like 'Bases' and 'xslt', and various XML/XSLT files.
- Editor Panel:** Contains an XSLT script:
 

```
1 for $lib in distinct-values(/bookstore/book/year)
2 return <year>{$lib}</year>
```
- Right Panel:**
  - doc2.xml:** A preview of the source XML document showing a 'bookstore' with 'book' elements containing 'title', 'author', and 'year'.
  - Result:** Displays the output of the XSLT transformation:
 

```
<year>2005</year>
<year>2003</year>
```
  - Compiling:** A status bar at the bottom right showing compilation details.

- **Mostrar todos los autores eliminando los que se repiten y ordenados por el número de caracteres que tiene cada autor.**

```
for $lib in distinct-values(/bookstore/book/author)
```

```
let $scar := string-length($lib)
```

```
order by $scar
```

```
return
```

```
<autor>{$scar}, {$lib}</autor>
```

**Editor**

```

1 for $lib in distinct-values(/bookstore/book/author)
2 let $car := string-length($lib)
3 order by $car
4 return
5 <autor>{$car}, {$lib}</autor>

```

**Result**

```

<autor>10, Kurt Cagle</autor>
<autor>10, James Linn</autor>
<autor>11, Per Bothner</autor>
<autor>11, Erik T. Ray</autor>
<autor>12, J K. Rowling</autor>
<autor>14, James McGovern</autor>
<autor>19, Giada De Laurentiis</autor>
<autor>22, Vaidyanathan Nagarajan</autor>

```

**doc2.xml**

book title	author	book title	author
Everyday Italian	Giada De Laur..	Harry Potter	J K. Rowl..
year	price	year	price
2005	30.00	2005	29.99

**Compiling:**  
- pre-evaluate root to document-node(): root()  
- pre-evaluate fn:distinct-values(items[, collation, 0])\*... -> ("Giada De Laurentiis", ...)

**Optimized Query:**  
for \$lib\_0 in ("Giada De Laurentiis", ...) let \$car return element autor { (\$car\_1, ", ", \$lib\_0) }

**Query:**  
for \$lib in distinct-values(/bookstore/book/author) return <autor>{\$car}, {\$lib}</autor>

**Result:**  
- Hit(s): 8 Items

- **Mostrar los títulos en una tabla de HTML.**

```

<table>
  <tr><th>Títulos</th></tr>

  {
    for $lib in (/bookstore/book/author)
    return
    <tr><td>{data($lib)}</td></tr>
  }

</table>

```

**Editor**

```

1 <table>
2

```

**Result**

```

<tr>
<th>Títulos</th>
</tr>
<tr>
<td>Giada De Laurentiis</td>
</tr>
<tr>
<td>J K. Rowling</td>
</tr>
<tr>
<td>James McGovern</td>
</tr>
<tr>
<td>Per Bothner</td>
</tr>
<tr>
<td>Kurt Cagle</td>
</tr>
<tr>
<td>James Linn</td>
</tr>
<tr>
<td>Vaidyanathan Nagarajan</td>
</tr>
<tr>
<td>Erik T. Ray</td>
</tr>
</table>

```

**doc2.xml**

book title	author	book title	author
Everyday Italian	Giada De Laur..	Harry Potter	J K. Rowl..
year	price	year	price
2005	30.00	2005	29.99

**Compiling:**  
- pre-evaluate root to document-node(): root()

**Optimized Query:**  
element table { (element tr { (element td { element bookstore/\*:book/\*:author return \$lib }) </td></tr> } ) </table>

**Query:**  
<table> <tr><th>Títulos</th></tr> <tr><td>{data(\$lib)}</td></tr> } </table>

**Result:**  
- Hit(s): 1 Item  
- Updated: 0 Items  
- Printed: 393 b  
- Read Locking: doc2  
- Write Locking: (none)