

01

```
for $o in fn:doc('data.xml')//owner
where $o/address/country = "Česká republika"
order by $o/name
return $o
```

```
for $o in //owner ...
```

```
for $o in fn:doc('data.xml')//owner[address/country = "Česká republika"]
order by $o/name
return $o
```

```
for $o in fn:doc('data.xml')//owner
let $c := $o/address/country
where $c = "Česká republika"
order by $o/name
return $o
```

```
for $p in fn:doc('data.xml')//property
return
  <property>
    { $p/@idProperty }
    <owner>
      {
        fn:doc('data.xml')//owner[@idOwner = $p/@owner]/name/text()
      }
    </owner>
    { $p/address }
  </property>

... return ... <property idProperty="{ $p/@idProperty }"> ...
```

```

for $e in fn:doc('data.xml')//agency/employee
let $a := $e/parent::agency
return
  element employee {
    element name {
      text { fn:concat($e/firstName, " ", $e/lastName) }
    },
    text { " works at " },
    element agency {
      attribute id { $a/@idAgency },
      text { $a/name }
    }
  }

... return
  element employee {
    element name {
      fn:concat($e/firstName, " ", $e/lastName)
    },
    " works at ",
    element agency {
      attribute id { fn:data($a/@idAgency) },
      $a/name/text()
    }
  }

... return ... element { "employee" } { ... } ...

for $a in fn:doc('data.xml')//agency
for $e in $a/employee
return ...

for $a in fn:doc('data.xml')//agency, $e in $a/employee
return ...

for $a in fn:doc('data.xml')//agency
return
  for $e in $a/employee
  return ...

```

```

for $f in fn:doc('data.xml')//flat
let $pName := //property[@idProperty = $f/@property]/name
order by $pName descending, $f/name
return
  (
    element property { data($f/@property) },
    $f/name,
    <comfort>{ data($f/@comfort) }</comfort>
  )

... order by $pName descending, $f/name ascending ...

... order by $pName/text() ...

... order by //property[@idProperty = $f/propertyRef]/name descending, $f/name ...

```

```
let $flats := fn:doc('data.xml')//flat
let $avg := fn:avg($flats/rate)
for $f in $flats
let $c := $f/@comfort
where ($f/rate < $avg) and ($c = "B" or $c = "C")
return fn:data($f/@idFlat)

let $flats := ..., $avg := ... ...

... where ($f/rate < $avg) and ($c = ("B", "C")) ...
```

```

<table>
  <tr>
    <th>Id</th><th>Name</th><th>Features</th>
  </tr>
  {
    for $p in fn:doc('data.xml')//property
    return
      <tr>
        <td>{ fn:data($p/@idProperty) }</td>
        <td>{ $p/name/text() }</td>
        <td>
          { $p/features/feature[1]/text() }
          {
            for $t in $p/features/feature[position() != 1]
            return fn:concat(" ", $t/text())
          }
        </td>
      </tr>
  }
</table>

... .. return text { fn:concat(" ", $t/text()) } ...

... .. return (" ", $t/text()) ...

... <td> {
  let $last := $p/features/feature[last()]
  for $t in $p/features/feature
  return
    if ($last is $t)
    then $t/text()
    else ($t/text(), " ")
} </td> ...

... <td> {
  for $t in $p/features/feature
  return
    if (not($t/following-sibling::feature))
    then $t/text()
    else ($t/text(), " ")
} </td> ...

... <td> {
  let $count := count($p/features/feature)
  for $t at $pos in $p/features/feature
  return
    if ($pos = $count)
    then $t/text()
    else ($t/text(), " ")
} </td> ...

... <td>{ fn:string-join($p/features/feature, " ") }</td> ...

```

```

for $p in fn:doc('data.xml')//property
let $flats := fn:doc('data.xml')//flats/flat[@property = $p/@idProperty]
return
  if (every $f in $flats satisfies $f/rate > 10000)
  then <luxuryProperty name="{ $p/name/text() }"/>
  else <commonProperty name="{ $p/name/text() }"/>

... return
  element {
    if (every $f in $flats satisfies $f/rate > 10000)
    then "luxuryProperty"
    else "commonProperty"
  } {
    attribute name { $p/name/text() }
  }

... let $n :=
  if (every $f in $flats satisfies $f/rate > 10000)
  then "luxuryProperty"
  else "commonProperty"
return
  element { $n } {
    attribute name { $p/name/text() }
  }

... .. if (count($flats) = count($flats[rate > 10000])) ... ..
... .. if (not($flats[not(rate > 10000)])) ... ..

```

08

```
for $c in fn:distinct-values(fn:doc('data.xml')//flat/@comfort)
let $flats := fn:doc('data.xml')//flat[@comfort = $c]
let $count := fn:count($flats)
where $count >= 2
order by $c ascending
return
  <group comfort="{ $c }">
    {
      for $f in $flats
      order by $f/@idFlat
      return <flat id="{ $f/@idFlat }"/>
    }
  </group>

for $c in ("A", "B", "C", "D", "E", "F") ...
```

```

let $properties := fn:doc('data.xml')//property
let $values :=
  for $p in $properties
  return fn:count(fn:doc('data.xml')//flat[@property = $p/@idProperty])
let $max := fn:max($values)
for $p in $properties
let $count := fn:count(fn:doc('data.xml')//flat[@property = $p/@idProperty])
where $count = $max
order by $p/name ascending
return
  <property id="{ $p/@idProperty }" flats="{ $count }"/>

let $properties := fn:doc('data.xml')//property
let $values :=
  for $p in $properties
  let $count := fn:count(fn:doc('data.xml')//flat[@property = $p/@idProperty])
  return
    <item>
      <id>{ fn:data($p/@idProperty) }</id>
      <name>{ $p/name/text() }</name>
      <count>{ $count }</count>
    </item>
let $max := fn:max($values/count)
for $i in $values
where $i/count = $max
order by $i/name ascending
return
  <property id="{ $i/id }" flats="{ $i/count }"/>

```

```
for $a1 in fn:doc('data.xml')//agency
for $a2 in fn:doc('data-agencies.xml')//agency
where $a1/@idAgency = $a2/id
return
  <agency id="{ $a1/@idAgency }">
    <contact type="email">{ $a1/email/text() }</contact>
    <contact type="web">{ $a2/web/text() }</contact>
    <contact type="phone">{ $a1/phone/text() }</contact>
  </agency>
```

```

<flats>
{
  let $list1 :=
    for $f in fn:doc('data.xml')//flat
    return
      element flat {
        attribute id { $f/@idFlat },
        attribute name { $f/name },
        attribute property { $f/@property },
        attribute comfort { $f/@comfort }
      }
  let $list2 :=
    for $f in fn:doc('data-flats.xml')//flat
    let $p := fn:doc('data.xml')//property[name = $f/propertyName]
    return
      element flat {
        attribute id { $f/flatId },
        attribute name { $f/flatName },
        attribute property { $p/@idProperty },
        attribute comfort { $f/comfortLevel }
      }
  for $f in $list1 union $list2
  order by $f/@id ascending
  return $f
}
</flats>

... .. for $f in $list1 | $list2 ... ..

```