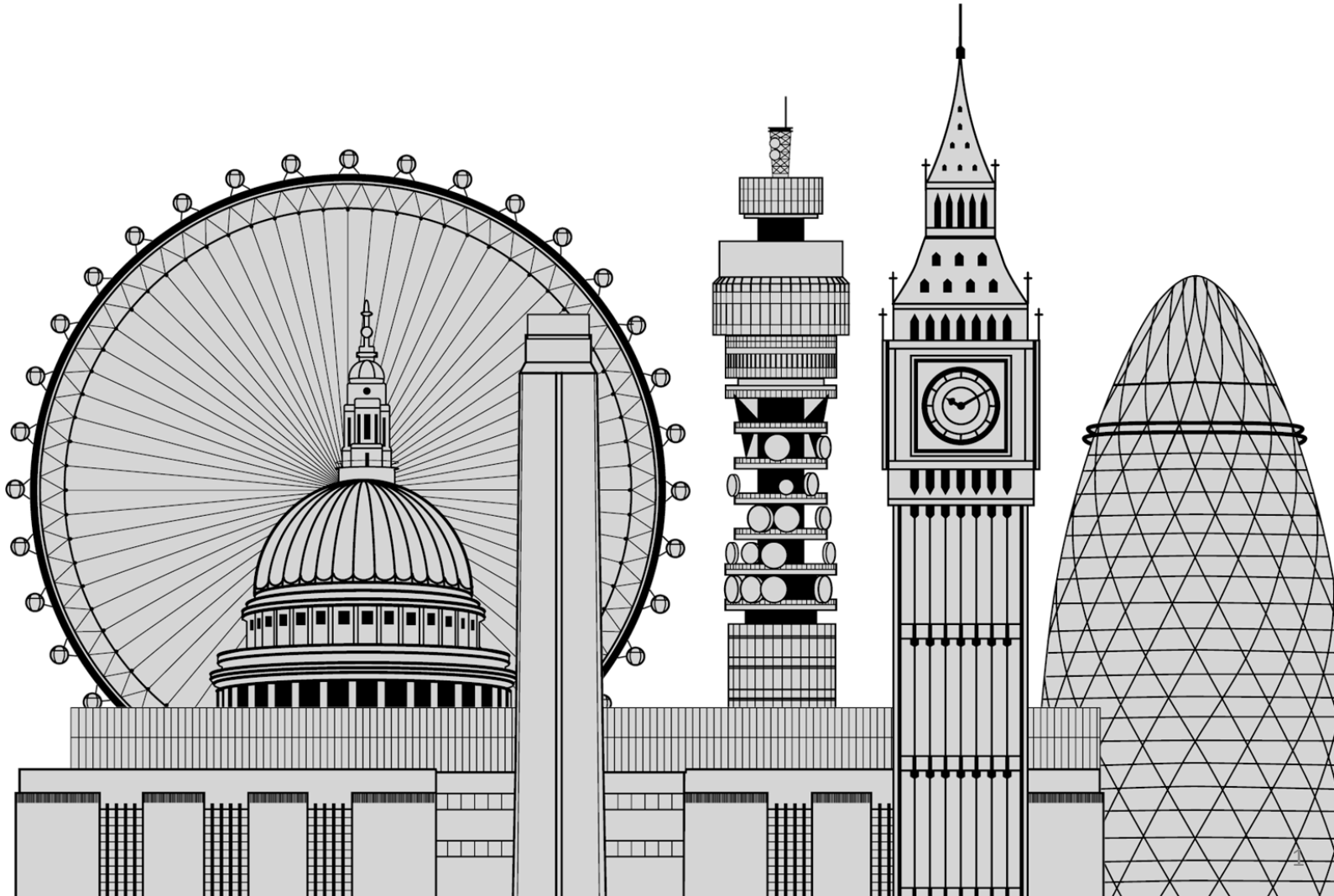
5COSC020W DATABASE SYSTEMS – LECTURE 10

XQuery – Querying XML docs and retrieving elements & attributes

UNIVERSITY OF WESTMINSTER

Dr Francois ROUBERT
F.Roubert@westminster.ac.uk



LECTURE 10 – OUTLINE

- **Extending XML** – XPath, XSLT, XQuery

- **XQuery – XML Query Language**
  - Definition
  - Benefits
  - Utilisations
  - Capabilities

- **XQuery**
  - Doc function and XPath Expressions
  - FLWOR Expressions: FOR, LET, WHERE, ORDER and RETURN
  - Sequences, sequence functions, string functions, date functions and regular expr.
  - If Then Else

Extending XML – XPath, XSLT and XQuery

Additional languages to (1) address and navigate through; (2) transform and render; and (3) query XML documents

1) XPath – XML Path Language

- Language to address and navigate through an XML document.
- It uses path expressions to select and return nodes by following tree structure.

2) XSLT – eXtensible Stylesheet Language Transformations

- Language to render and transform an XML document into a viewable output file.
- It uses XPath to navigate through and locate specific elements and attributes.

3) XQuery – XML Query Language

- Query-based language to extract and retrieve data stored in XML format.
- It uses XPath to navigate through and locate specific elements and attributes.

XQuery – XML Query Language

- Standardised language to retrieve information stored in XML format.

- **Functional Language**
  - Language to retrieve/querying XML data of interest, reorganise them, possibly transform them and return results in chosen structure .

- **Analogous to SQL**
  - XQuery is to XML what SQL is to databases.

- **Based on XPath**
  - XQuery uses XPath expressions to navigate through XML documents.

- **Universally accepted**
  - XQuery is supported by all major databases.

Benefits of using XQuery

- Using XQuery, both hierarchical and tabular data can be retrieved.
- XQuery can be used to query tree and graphical structures.
- XQuery can be directly used to query webpages.
- XQuery can be directly used to build webpages.
- XQuery can be used to transform XML documents.
- XQuery is ideal for XML-based databases and object-based databases.

Examples of utilisations of XQuery

- Finding textual docs in a native XML database and presenting styled results.
- Generating reports on data stored in a datbase for presentation as HTML.
- Extracting information from a relational database for use in a web service.
- Pulling data from databases or packaged software and transforming it for application integration.
- Combining content from traditionally non-XML sources to implement content management and delivery
- Ad hoc querying of standalone XML documents for the purposes of testing or research.
- Building entire complex web applications.

Capabilities of XQuery

- Selecting information based on specific criteria.
- Filtering out unwanted information.
- Searching for information within a document or set of documents.
- Joining data from multiple documents or collections of documents.
- Sorting, grouping, and aggregating data.
- Transforming and restructuring XML data into another XML structure.
- Performing arithmetic calculations on numbers and dates.
- Manipulating strings to reformat text.

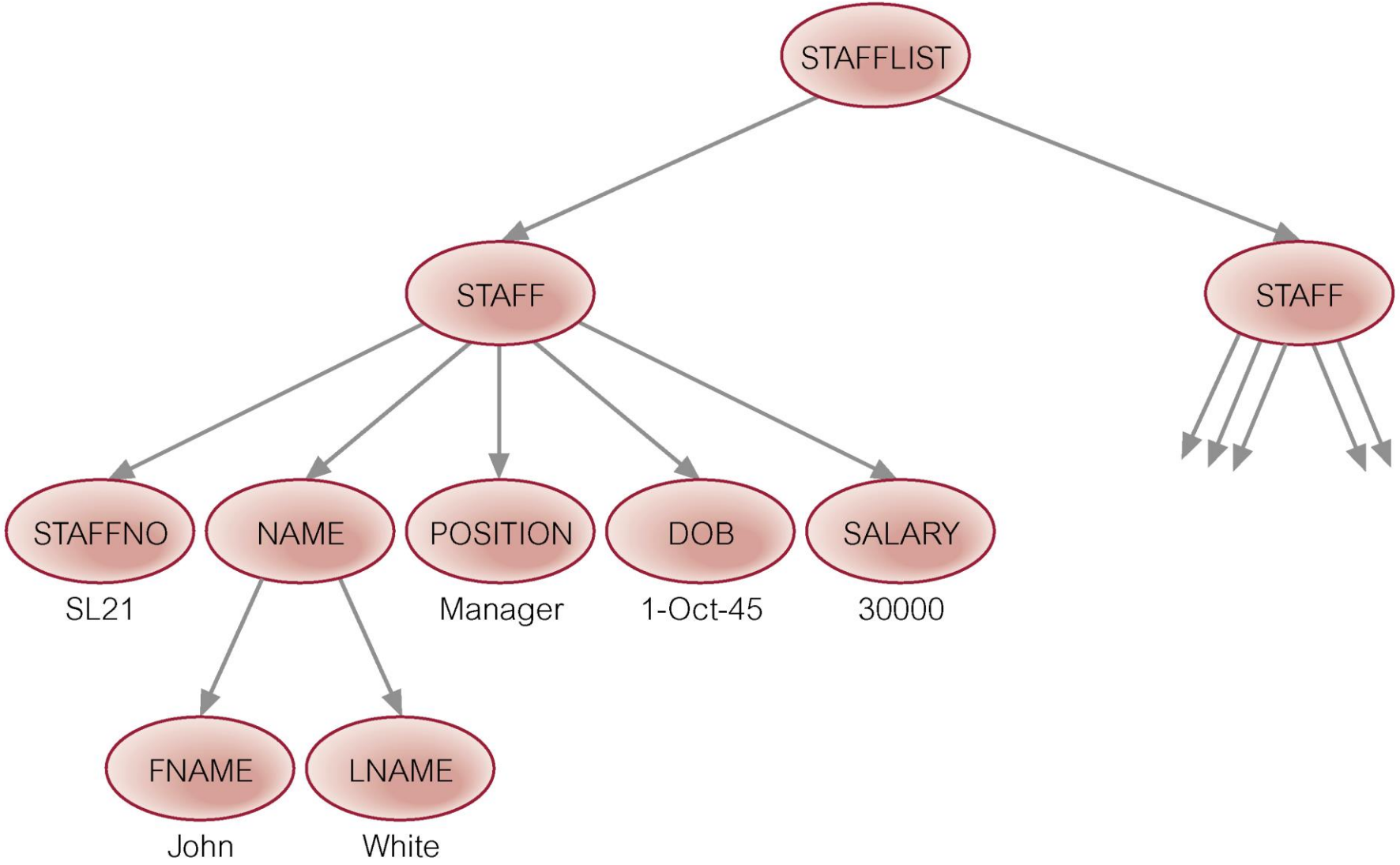
EXAMPLE: XML DOCUMENT – dreamhome_stafflist.xml

dreamhome_stafflist.xml

```xml
<?xml version="1.0" encoding="utf-8" standalone = "yes"?> <!--XML declaration-->
<?xml-stylesheet type="text/xsl" href="dreamhome_stylesheet.xsl"?> <!--XSL reference-->

<STAFFLIST> <!--STAFFLIST root elmt (1st element) -->
    <STAFF branchNo="B005"> <!--STAFF sub element (child element), branchNo attribute -->
        <STAFFNO>SL21</STAFFNO> <!--STAFFNO sub element of STAFF-->
        <NAME> <!--NAME sub element of STAFF-->
            <FNAME>John</FNAME><LNAME>White</LNAME><!--Elements must be properly nested-->
        </NAME>
        <POSITION>Manager</POSITION><!--Elements are case sensitive-->
        <DOB>1-Oct-45</DOB>
        <SALARY>30000</SALARY>
    </STAFF>
    <STAFF branchNo="B003">
        <STAFFNO>SG37</STAFFNO>
        <NAME>
            <FNAME>Ann</FNAME><LNAME>Beech</LNAME>
        </NAME>
        <POSITION>Assistant</POSITION>
        <DOB>12-Mar-63</DOB>
        <SALARY>12000</SALARY>
    </STAFF>
</STAFFLIST>
```

EXAMPLE: TREE STRUCTURE– dreamhome_stafflist.xml



Doc function and XPath Expressions

- **doc("dreamhome_stafflist.xml")**
  Use the doc function to locate the XML source file to be opened.

- **doc("dreamhome_stafflist.xml")/STAFFLIST/STAFF**
  Use XPath expression to navigate down the tree and locate right XML element.

- **doc("dreamhome_stafflist.xml")/STAFFLIST/STAFF/NAME/FNAME**
  Use XPath expression to navigate down the tree and locate right XML element.

- **doc("dreamhome_stafflist.xml")//FNAME**
  Use XPath expression with a // to locate element anywhere in the document.

- **doc("dreamhome_stafflist.xml")/STAFFLIST/*/DOB**
  Uses XPath expression with a wildcard to indicate any XML element.

Doc function and XPath Expressions with Predicates

- **doc("dreamhome_stafflist.xml")//STAFF[2]**
  Uses XPath expression with position predicate to select 2nd element of node.

- **doc("dreamhome_stafflist.xml")//STAFF[SALARY >= 30000]/NAME/FNAME**
  Uses XPath expression with condition predicate on element.

- **doc("dreamhome_stafflist.xml")//STAFF [POSITION = "Manager"]/SALARY**
  Uses XPath expression with condition predicate on element.

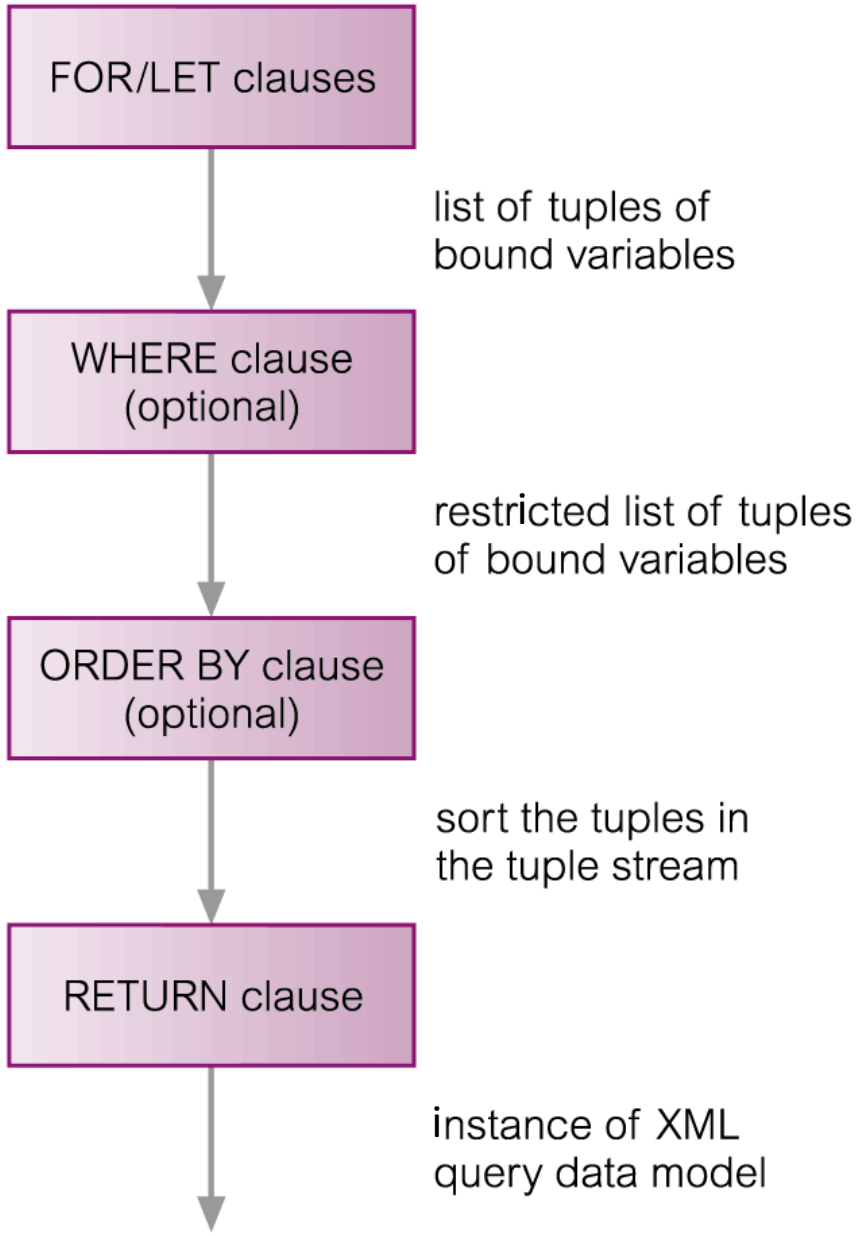
- **doc("dreamhome_stafflist.xml")//STAFF [@branchNo="B003"]/NAME**
  Uses XPath expression with condition predicate on attribute.

- **doc("dreamhome_stafflist.xml")//STAFF [@branchNo="B005"]//FNAME**
  Uses XPath expression with condition predicate on attribute.

XQuery – FLWOR Expressions (1)

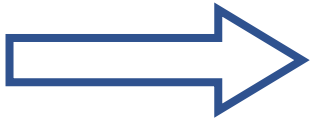
- **FOR**
  - Selects a sequence of nodes & sets up an iteration over nodes returned by expr.
- **LET**
  - Binds the value of a variable i.e. associates single expression to single variable.
- **WHERE**
  - Specifies a condition to restrict the nodes to be returned.
- **ORDER**
  - Specifies a criteria to sort (i.e. determine the order of) the nodes to be returned.
- **RETURN**
  - Indicates the final set of nodes to be returned (evaluated once for every node).

XQuery – FLWOR Expressions (2)



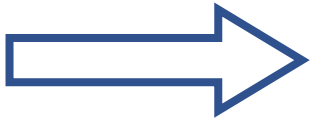
FOR to iterate (single clause)

```
for $i in 1 to 3
return <oneval>{$i}</oneval>
```



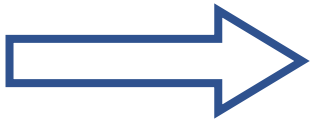
```
<oneval>1</oneval>
<oneval>2</oneval>
<oneval>3</oneval>
```

```
for $i in reverse (1 to 3)
return <oneval>{$i}</oneval>
```



```
<oneval>3</oneval>
<oneval>2</oneval>
<oneval>1</oneval>
```

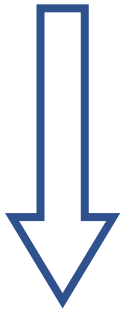
```
for $i in (1 to 10)[. mod 2 = 0]
return <oneval>{$i}</oneval>
```



```
<oneval>2</oneval>
<oneval>4</oneval>
<oneval>6</oneval>
<oneval>8</oneval>
<oneval>10</oneval>
```

FOR to iterate (multiple clauses)

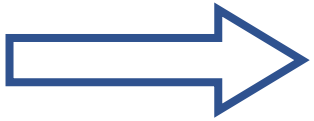
```
for $i in (1, 2)
for $j in ("a", "b", "c")
return <oneval>i is {$i} and j is {$j}</oneval>
```



```
<oneval>i is 1 and j is a</oneval>
<oneval>i is 1 and j is b</oneval>
<oneval>i is 1 and j is c</oneval>
<oneval>i is 2 and j is a</oneval>
<oneval>i is 2 and j is b</oneval>
<oneval>i is 2 and j is c</oneval>
```

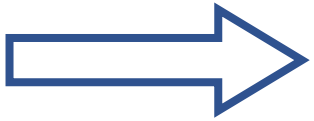
LET to bind a variable (1)

```
let $i := 5
return <oneval>{$i}</oneval>
```



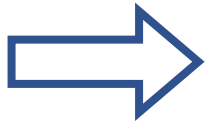
```
<oneval>5</oneval>
```

```
let $i := (1 to 5)
return <oneval>{$i}</oneval>
```



```
<oneval>1 2 3 4 5</oneval>
```

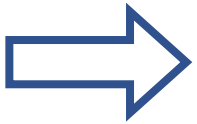
```
let $i := 5
let $j := 9
return <oneval>i is {$i}
and j is {$j} and the sum is {$i+$j}</oneval>
```



```
<oneval>i is 5 and j is 9 and the
sum is 14</oneval>
```

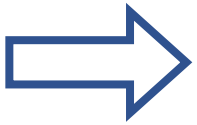
LET to bind a variable (2)

```
let $SAL := 30000
return
doc("dreamhome_stafflist.xml")//STAFF[SALARY=$SAL]/NAME
```



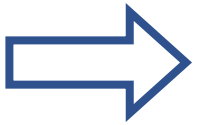
```
<NAME>
  <FNAME>John</FNAME>
  <LNAME>White</LNAME>
</NAME>
```

```
let $SAL := 10000
return
doc("dreamhome_stafflist.xml")//STAFF[SALARY >
$SAL]//FNAME
```



```
<FNAME>John</FNAME>
<FNAME>Ann</FNAME>
```

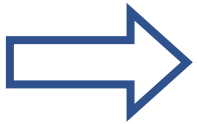
```
let $doc := doc("dreamhome_stafflist.xml")
let $position := "Manager"
return $doc//STAFF[POSITION=$position]//FNAME
```



```
<FNAME>John</FNAME>
```

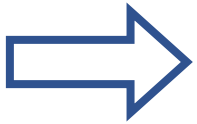
WHERE to restrict the nodes

```
let $doc := doc("dreamhome_stafflist.xml")
for $S in $doc//STAFF
where $S/POSITION ="Manager"
return $S//FNAME
```



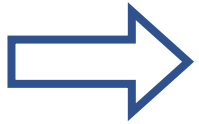
```
<FNAME>John</FNAME>
```

```
let $doc := doc("dreamhome_stafflist.xml")
for $S in $doc//STAFF
where $S/SALARY > 15000 and $S/@branchNo = "B005"
return $S/STAFFNO
```



```
<STAFFNO>SL21</STAFFNO>
```

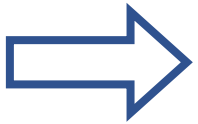
```
let $doc := doc("dreamhome_stafflist.xml")
for $S in $doc//STAFF
where $S/SALARY < 15000 or $S/SALARY > 25000
return $S//LNAME
```



```
<LNAME>White</LNAME>
<LNAME>Beech</LNAME>
```

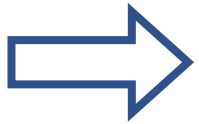
ORDER to sort the nodes

```
let $doc := doc("dreamhome_stafflist.xml")
for $S in $doc//STAFF
order by $S//LNAME
return ($S//FNAME, $S//LNAME)
```



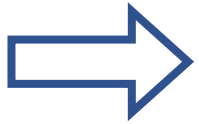
```
<FNAME>Ann</FNAME>
<LNAME>Beech</LNAME>
<FNAME>John</FNAME>
<LNAME>White</LNAME>
```

```
let $doc := doc("dreamhome_stafflist.xml")
for $S in  $doc//STAFF
order by $S//SALARY descending
return ($S/POSITION, $S/SALARY)
```



```
<POSITION>Manager</POSITION>
<SALARY>30000</SALARY>
<POSITION>Assistant</POSITION>
<SALARY>12000</SALARY>
```

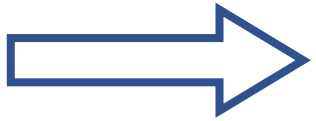
```
let $doc := doc("dreamhome_stafflist.xml")
for $S in $doc//STAFF
where $S/@branchNo = "B003"
        or $S/@branchNo = "B005"
order by $S/SALARY
return $S/SALARY
```



```
<SALARY>12000</SALARY>
<SALARY>30000</SALARY>
```

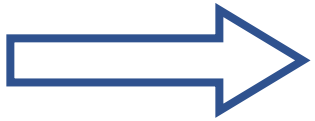
RETURN to retrieve the nodes

```
for $i in 1 to 3
return <oneval>{$i}</oneval>
```



```
<oneval>1</oneval>
<oneval>2</oneval>
<oneval>3</oneval>
```

```
for $i in (1 to 3)
return
(<single>{$i}</single>,
<double>{$i*2}</double>)
```

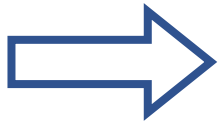


```
<single>1</single>
<double>2</double>
<single>2</single>
<double>4</double>
<single>3</single>
<double>6</double>
```

Sequences

- Sequences to represent an ordered collection of heterogeneous items

```
let $items := ('orange', <apple type="sour">
Golden </apple>,1,2,3,'a','b',"abc")
let $count := count($items)
return
<result>
  <count>{$count}</count>
    <items>
    {
       for $item in $items
          return <item>{$item}</item>
    }
  </items>
  </result>
```



```
<result>
 <count>8</count>
 <items>
  <item>orange</item>
  <item>
   <apple type="sour"> Golden </apple>
  </item>
  <item>1</item>
  <item>2</item>
  <item>3</item>
  <item>a</item>
  <item>b</item>
  <item>abc</item>
 </items>
</result>
```

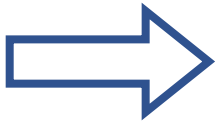
Sequence Functions

Sequence Function	Description
count($items)	Counts the items in a sequence.
sum($items)	Returns the sum of the items in a sequence.
avg($items)	Returns the average of the items in a sequence.
min($items)	Returns the minimum valued item in a sequence.
max($items)	Returns the maximum valued item in a sequence.
distinct-values($items)	Gets the sequence containing unique items in a sequence.
subsequence($items,2,4)	Returns a subset of provided sequence.
insert-before ($items,6,$extra-items)	Inserts an item in a sequence.
remove($items,4)	Removes an item from a sequence.
reverse($items)	Returns the reversed sequence.
$items[last()]	Returns the last element of a sequence (used in predicate expression).
$items[position()]	Get the position of an item in a sequence (used in FLOWR expr.).

Sequence Functions – Example (1)

- Using the distinct_values function to get unique items and the avg function to calculate the average.

```
let $doc := doc("dreamhome_stafflist.xml")
for $B in distinct-values($doc//@branchNo)
let $avgSalary :=
avg($doc//STAFF[@branchNo = $B]/SALARY)
return
<BRANCH>
        <BRANCHNO>{$B}</BRANCHNO>
        <AVGSAL>{$avgSalary}</AVGSAL>
</BRANCH>
```

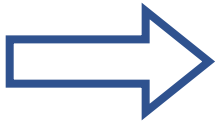


```
<BRANCH>
 <BRANCHNO>B005</BRANCHNO>
 <AVGSAL>30000</AVGSAL>
</BRANCH>
<BRANCH>
 <BRANCHNO>B003</BRANCHNO>
 <AVGSAL>12000</AVGSAL>
</BRANCH>
```

Sequence Functions – Example (2)

- Using the distinct_values function to get unique items and the count function to determine the number of items.

```
let $doc := doc("dreamhome_stafflist.xml")
for $B in distinct-values($doc//@branchNo)
let $S := $doc//STAFF/[@branchNo = $B]
where count($S) <= 2
return
<SMALLBRANCHES>
        <BRANCHNO>{$B}</BRANCHNO>
</SMALLBRANCHES>
```



```
<SMALLBRANCHES>
 <BRANCHNO>B005</BRANCHNO>
</SMALLBRANCHES>
<SMALLBRANCHES>
 <BRANCHNO>B003</BRANCHNO>
</SMALLBRANCHES>
```

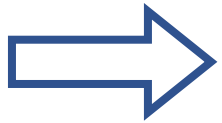
String Functions, Date Functions and Regular Expressions

String Function	Description
string-length($bookTitle)	Returns the length of the string as an integer
concat($bookTitle,",price: £25.99")	Returns the concatenated string as output.
string-join($fruits/fruit, ',')	Returns the combination of items in a sequence separated by a delimiter.
Date Function	**Description**
current-date()	Returns the current date.
current-time()	Returns the current time.
current-dateTime()	Returns both the current date and the current time.
Regular Expression Function	**Description**
matches($input, $regex)	Returns true if the input matches with the provided regular expression, otherwise false.
replace($input, $regex, $string)	Replaces the matched input string with given string.
tokenize($input, $regex)	Returns a sequence of items matching the regular expression.

String Functions – Example (1)

- Using the string-join function to combine elements and separate them with a comma delimiter.

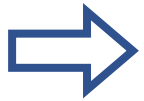
```
let $fruits :=
<fruits>
  <fruit>Apple</fruit>
  <fruit>Orange</fruit>
  <fruit>Mango</fruit>
</fruits>
return
<results>
    <fruits>
          {string-join($fruits/fruit,', ')}
    </fruits>
</results>
```



```
<results>
  <fruits>Apple, Orange, Mango</fruits>
</results>
```

If Then Else to check the validity of the input

```
<HIGHEARNERS>
{
  if(not(doc("dreamhome_stafflist.xml"))) then (
    <error>
      <msg>The staff list does not exist </msg>
    </error>
  )
  else (
    for $S in doc("dreamhome_stafflist.xml")//STAFF
    where $S/SALARY>28000
    return ($S/POSITION, $S/SALARY)
  )
}
</HIGHEARNERS>
```



```
<HIGHEARNERS>
 <POSITION>Manager</POSITION>
 <SALARY>30000</SALARY>
</HIGHEARNERS>
```

REFERENCES

- Connolly, T. & Begg, C. E. (2015). Database systems: a practical approach to design, implementation and management. 6th Edition (Global Edition). Pearson Ed.

- Elmasri, R. & Navathe, S. (2017). Fundamentals of Database Systems. 7th Edition (Global Edition). Pearson Education.

- Tutorialspoint (2021). XQuery Tutorial. Available from https://www.tutorialspoint.com/xquery/index.htm. [Accessed 30 May 2021]

- W3Schools (2021). XQuery Tutorial. Available from https://www.w3schools.com/xml/xquery_intro.asp. [Accessed 30 May 2021]

- Walmsley, P (2015). XQuery: search across a variety of XML data. 2nd Edition. O'Reilly.