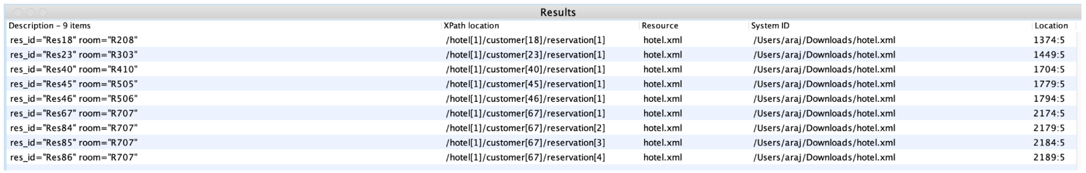
XPath Queries

1. Select all reservations made by customers whose name starts with the letter "a".

   XPath: //reservation[../@cust_id=//customer[starts-with(name,"a")]/@cust_id]
   Result:



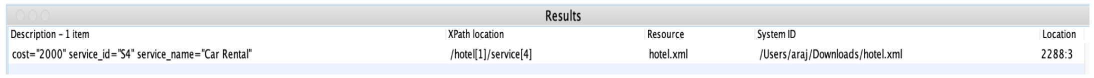
Results

Description – 9 items	XPath location	Resource	System ID	Location
res_id="Res18" room="R208"	/hotel[1]/customer[18]/reservation[1]	hotel.xml	/Users/araj/Downloads/hotel.xml	1374:5
res_id="Res23" room="R303"	/hotel[1]/customer[23]/reservation[1]	hotel.xml	/Users/araj/Downloads/hotel.xml	1449:5
res_id="Res40" room="R410"	/hotel[1]/customer[40]/reservation[1]	hotel.xml	/Users/araj/Downloads/hotel.xml	1704:5
res_id="Res45" room="R505"	/hotel[1]/customer[45]/reservation[1]	hotel.xml	/Users/araj/Downloads/hotel.xml	1779:5
res_id="Res46" room="R506"	/hotel[1]/customer[46]/reservation[1]	hotel.xml	/Users/araj/Downloads/hotel.xml	1794:5
res_id="Res67" room="R707"	/hotel[1]/customer[67]/reservation[1]	hotel.xml	/Users/araj/Downloads/hotel.xml	2174:5
res_id="Res84" room="R707"	/hotel[1]/customer[67]/reservation[2]	hotel.xml	/Users/araj/Downloads/hotel.xml	2179:5
res_id="Res85" room="R707"	/hotel[1]/customer[67]/reservation[3]	hotel.xml	/Users/araj/Downloads/hotel.xml	2184:5
res_id="Res86" room="R707"	/hotel[1]/customer[67]/reservation[4]	hotel.xml	/Users/araj/Downloads/hotel.xml	2189:5

   Description: This XPath query selects all the <reservation> elements whose parent element has a cust_id attribute equal to the cust_id attribute of any <customer> element whose name starts with the letter "a".

2. Select all services that have a cost greater than the average cost of all services

   XPath: //service[@cost > avg(//service/@cost)]
   Result:



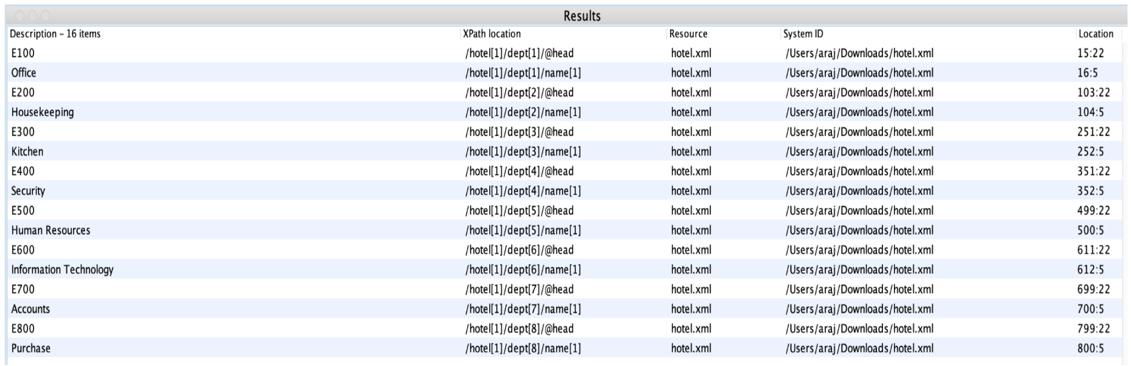
Results

Description – 1 item	XPath location	Resource	System ID	Location
cost="2000" service_id="S4" service_name="Car Rental"	/hotel[1]/service[4]	hotel.xml	/Users/araj/Downloads/hotel.xml	2288:3

   Description: This XPath query selects all the service elements that have a cost attribute greater than the average cost of all services in the XML document.

3. Select the names of all departments and their head employees.

   XPath: /hotel/dept/name | /hotel/dept/@head
   Result:



Results

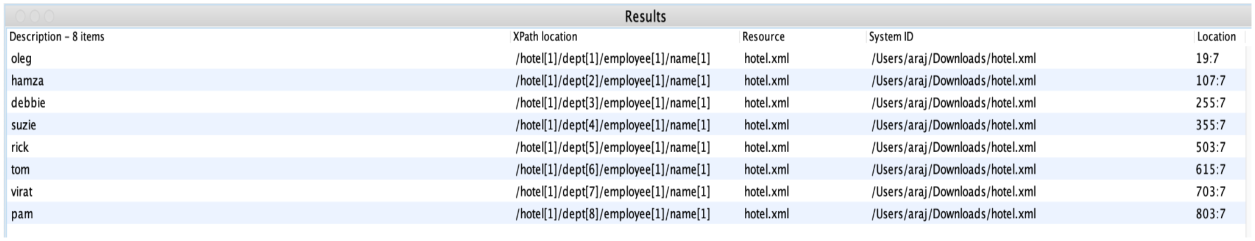
Description – 16 items	XPath location	Resource	System ID	Location
E100	/hotel[1]/dept[1]/@head	hotel.xml	/Users/araj/Downloads/hotel.xml	15:22
Office	/hotel[1]/dept[1]/name[1]	hotel.xml	/Users/araj/Downloads/hotel.xml	16:5
E200	/hotel[1]/dept[2]/@head	hotel.xml	/Users/araj/Downloads/hotel.xml	103:22
Housekeeping	/hotel[1]/dept[2]/name[1]	hotel.xml	/Users/araj/Downloads/hotel.xml	104:5
E300	/hotel[1]/dept[3]/@head	hotel.xml	/Users/araj/Downloads/hotel.xml	251:22
Kitchen	/hotel[1]/dept[3]/name[1]	hotel.xml	/Users/araj/Downloads/hotel.xml	252:5
E400	/hotel[1]/dept[4]/@head	hotel.xml	/Users/araj/Downloads/hotel.xml	351:22
Security	/hotel[1]/dept[4]/name[1]	hotel.xml	/Users/araj/Downloads/hotel.xml	352:5
E500	/hotel[1]/dept[5]/@head	hotel.xml	/Users/araj/Downloads/hotel.xml	499:22
Human Resources	/hotel[1]/dept[5]/name[1]	hotel.xml	/Users/araj/Downloads/hotel.xml	500:5
E600	/hotel[1]/dept[6]/@head	hotel.xml	/Users/araj/Downloads/hotel.xml	611:22
Information Technology	/hotel[1]/dept[6]/name[1]	hotel.xml	/Users/araj/Downloads/hotel.xml	612:5
E700	/hotel[1]/dept[7]/@head	hotel.xml	/Users/araj/Downloads/hotel.xml	699:22
Accounts	/hotel[1]/dept[7]/name[1]	hotel.xml	/Users/araj/Downloads/hotel.xml	700:5
E800	/hotel[1]/dept[8]/@head	hotel.xml	/Users/araj/Downloads/hotel.xml	799:22
Purchase	/hotel[1]/dept[8]/name[1]	hotel.xml	/Users/araj/Downloads/hotel.xml	800:5

   Description: This XPath query selects the names of all departments and the value of their "head" attribute in the XML document that conforms to the provided DTD. The "|" operator is used to combine two paths, which means the query selects the union of the nodes selected by each of the paths. The first path

"/hotel/dept/name" selects all "name" elements that are descendants of a "dept" element that is a child of the "hotel" element. The second path "/hotel/dept/@head" selects all "head" attributes of "dept" elements that are children of the "hotel" element.

4. Select the names of all employees who have at least one assistant.

   XPath: /hotel/dept/employee[assistant]/name
   Result:



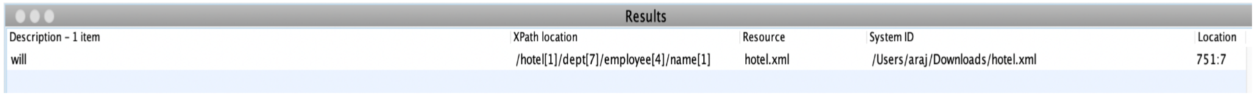
Results

Description - 8 items	XPath location	Resource	System ID	Location
oleg	/hotel[1]/dept[1]/employee[1]/name[1]	hotel.xml	/Users/araj/Downloads/hotel.xml	19:7
hamza	/hotel[1]/dept[2]/employee[1]/name[1]	hotel.xml	/Users/araj/Downloads/hotel.xml	107:7
debbie	/hotel[1]/dept[3]/employee[1]/name[1]	hotel.xml	/Users/araj/Downloads/hotel.xml	255:7
suzie	/hotel[1]/dept[4]/employee[1]/name[1]	hotel.xml	/Users/araj/Downloads/hotel.xml	355:7
rick	/hotel[1]/dept[5]/employee[1]/name[1]	hotel.xml	/Users/araj/Downloads/hotel.xml	503:7
tom	/hotel[1]/dept[6]/employee[1]/name[1]	hotel.xml	/Users/araj/Downloads/hotel.xml	615:7
virat	/hotel[1]/dept[7]/employee[1]/name[1]	hotel.xml	/Users/araj/Downloads/hotel.xml	703:7
pam	/hotel[1]/dept[8]/employee[1]/name[1]	hotel.xml	/Users/araj/Downloads/hotel.xml	803:7

   Description: This XPath query selects the name elements of all employee elements that have an assistant child element and are a child of a dept element that is a child of the root hotel element. In other words, it selects the names of all employees who have an assistant, and their corresponding department information.

5. Select the names of all employees who have a login ID that ends with the number "3" and work in the department with ID "D7".

   XPath: /hotel/dept[@dept_id='D7']/employee[substring(login_detail/@login_id, string-length(login_detail/@login_id))='3']/name
   Result:



Results

Description - 1 item	XPath location	Resource	System ID	Location
will	/hotel[1]/dept[7]/employee[4]/name[1]	hotel.xml	/Users/araj/Downloads/hotel.xml	751:7

   Description: This XPath expression selects the names of employees who belong to the department with the dept_id attribute value of "D7" and whose login ID ends with the character "3".
   The expression begins with /hotel/dept[@dept_id='D7'] which selects the dept element with dept_id attribute value of "D7". Next, it selects the employee element children of that dept element that have a login_detail/@login_id attribute whose value ends with the character "3". Finally, it selects the name element children of those employee elements, returning the names of the employees that meet both criteria.

—-----------------------—--------------—-------------—------------—---------------—------------------

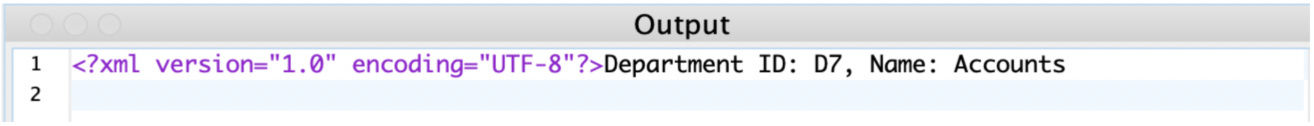
Xquery Queries

1. Retrieve the IDs and names of all departments that have at least one employee with the login ID "L701".

   Xquery:

   ```
   for $department in //dept
   for $employee in $department/employee
   where $employee/login_detail/@login_id="L701"
   return concat("Department ID: ", $department/@dept_id, ", Name: ", $department/name, "&#xa;")
   ```

   Result:

   Output

   ```
   <?xml version="1.0" encoding="UTF-8"?>Department ID: D7, Name: Accounts
   ```

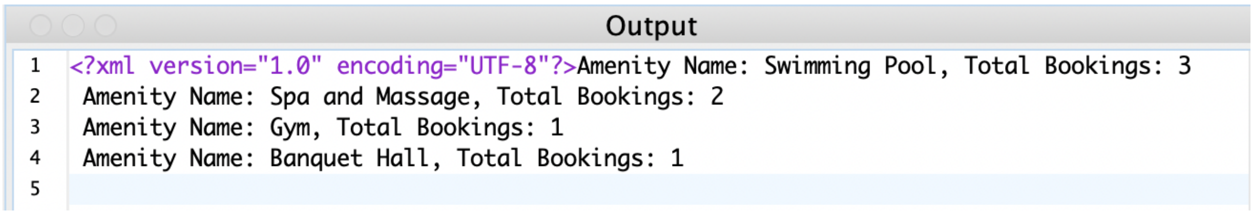
   Description:
   This XQuery iterates through all the dept elements in the XML document using the for loop. For each dept element, it iterates through all the employee elements using another for loop. Then, it checks if the login_id attribute of the login_detail element of the current employee element is equal to "L701". If it is true, the return statement concatenates the department ID and name with a new line character, using the concat() function. The resulting string shows the department ID and name of the department where an employee with the login ID "L701" is employed.

2. Retrieve the total number of bookings made for each amenity:

   Xquery:

   ```
   for $amenity in //amenity
   return concat("Amenity Name: ", $amenity/name, ", Total Bookings: ", count($amenity/booking), "&#xa;")
   ```

   Result:


```
<?xml version="1.0" encoding="UTF-8"?>Amenity Name: Swimming Pool, Total Bookings: 3
 Amenity Name: Spa and Massage, Total Bookings: 2
 Amenity Name: Gym, Total Bookings: 1
 Amenity Name: Banquet Hall, Total Bookings: 1

```

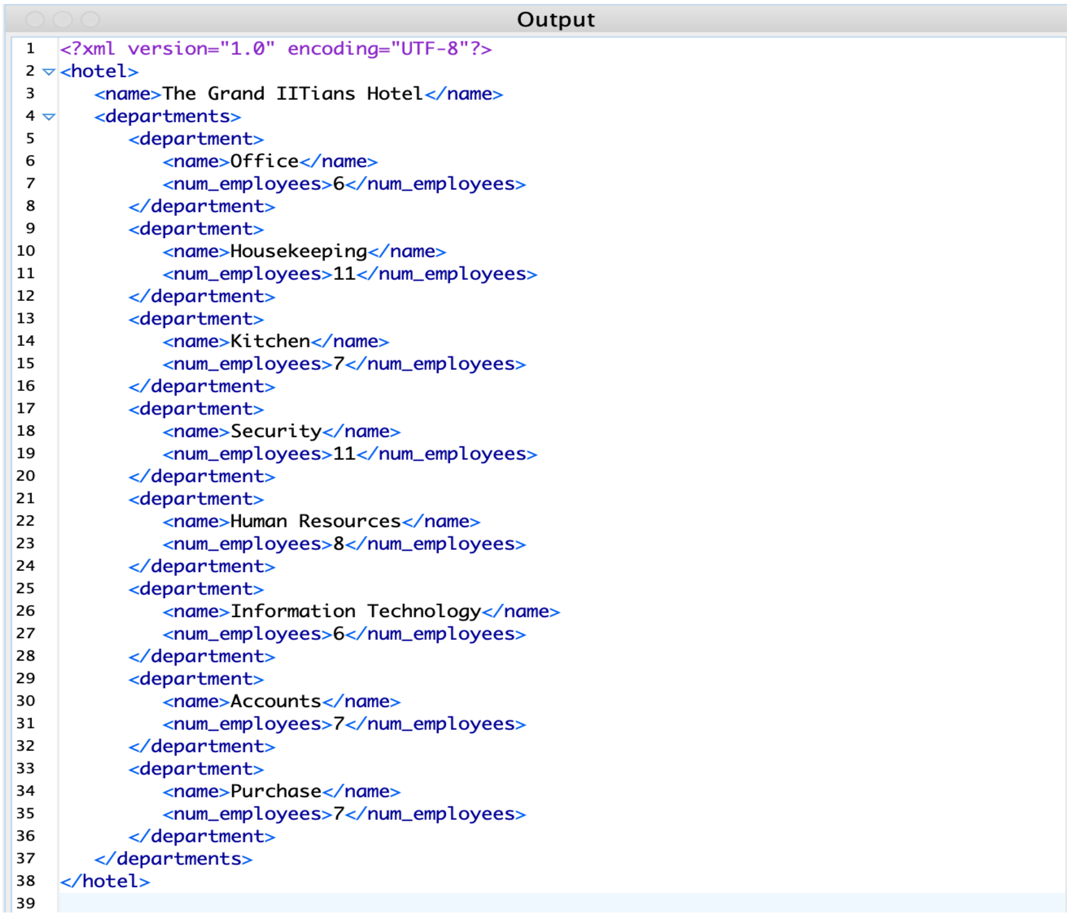
Description: This XQuery selects all the amenities in the hotel and returns a concatenated string that includes the name of the amenity and the total number of bookings made for that amenity.
The query starts with the "for" statement that iterates over all the amenities in the hotel using the path "//amenity". For each amenity, the query uses the "concat" function to construct a string that includes the amenity name and the total number of bookings. The "count" function is used to count the number of "booking" elements under the current amenity element. This is achieved using the path "$amenity/booking".
Finally, " " is used to add a newline character to the end of each concatenated string, which separates the results for each amenity.

3. Find the name of the hotel, the name of the departments with more than 3 employees and the number of employees in each department.

Xquery:

```
for $h in /hotel
return
  <hotel>
    <name>{data($h/name)}</name>
    <departments>
      {for $d in $h/dept[count(employee) > 3]
      return
        <department>
          <name>{data($d/name)}</name>
          <num_employees>{count($d/employee)}</num_employees>
        </department>}
    </departments>
  </hotel>
```

Result:


Output

```
<?xml version="1.0" encoding="UTF-8"?>
<hotel>
   <name>The Grand IITians Hotel</name>
   <departments>
      <department>
         <name>Office</name>
         <num_employees>6</num_employees>
      </department>
      <department>
         <name>Housekeeping</name>
         <num_employees>11</num_employees>
      </department>
      <department>
         <name>Kitchen</name>
         <num_employees>7</num_employees>
      </department>
      <department>
         <name>Security</name>
         <num_employees>11</num_employees>
      </department>
      <department>
         <name>Human Resources</name>
         <num_employees>8</num_employees>
      </department>
      <department>
         <name>Information Technology</name>
         <num_employees>6</num_employees>
      </department>
      <department>
         <name>Accounts</name>
         <num_employees>7</num_employees>
      </department>
      <department>
         <name>Purchase</name>
         <num_employees>7</num_employees>
      </department>
   </departments>
</hotel>
```

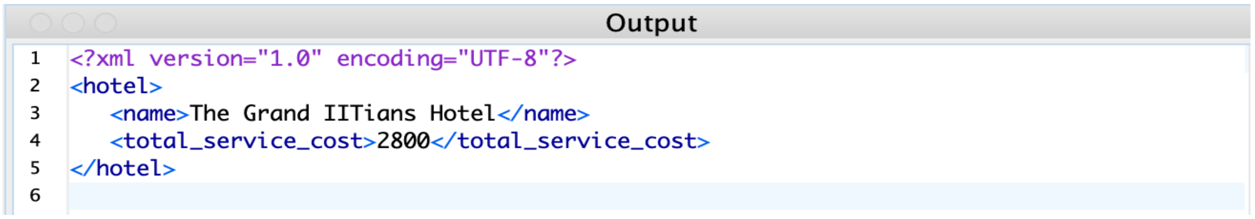
Description: This XQuery retrieves the name of all hotels along with the names of their departments that have more than 3 employees. It constructs an XML output with hotel names and a list of departments with their names and number of employees. The query uses a 'for' loop to iterate through each hotel and its departments. It then filters departments with more than 3 employees using the 'count' function and constructs the output XML with the relevant details using the 'return' statement.

4. Find the name of the hotel and the total cost of all the services booked by customers.

Xquery:

```
for $h in /hotel
return
  <hotel>
    <name>{data($h/name)}</name>
    <total_service_cost>{sum($h/service/@cost)}</total_service_cost>
  </hotel>
```

Result:



```
<?xml version="1.0" encoding="UTF-8"?>
<hotel>
   <name>The Grand IITians Hotel</name>
   <total_service_cost>2800</total_service_cost>
</hotel>
```

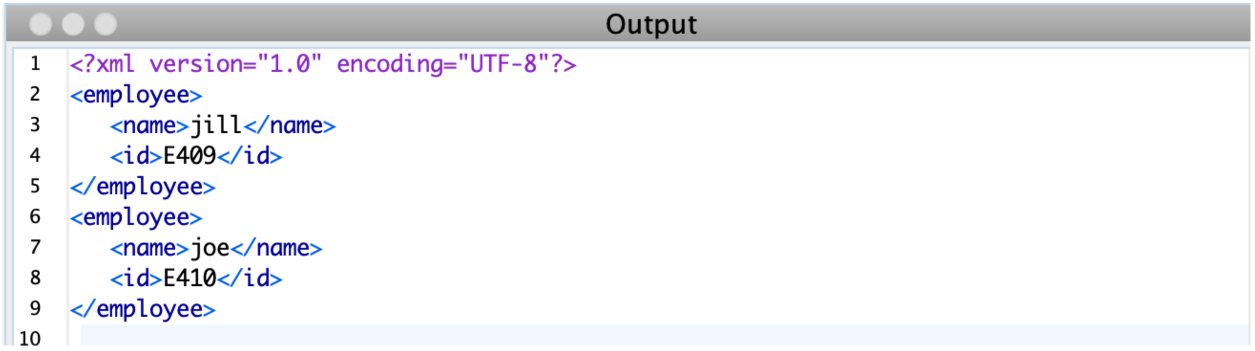
Description: This XQuery iterates through each hotel element in the XML document, and creates a new XML element hotel with the name of the hotel as its child element. It then calculates the sum of the cost of all services provided by that hotel and includes it as a child element total_service_cost within the hotel element. The resulting XML document will have one hotel element for each hotel in the original document, with its name and total service cost included.

5. Return the names and IDs of all employees who have the role of Bouncer.

Xquery:

```
for $employee in //employee[role = 'Bouncer']
return
   <employee>
      <name>{data($employee/name)}</name>
      <id>{data($employee/@emp_id)}</id>
   </employee>
```

Result:



```
<?xml version="1.0" encoding="UTF-8"?>
<employee>
   <name>jill</name>
   <id>E409</id>
</employee>
<employee>
   <name>joe</name>
   <id>E410</id>
</employee>
```

Description: This XQuery selects all the "employee" elements in the XML document whose "role" child element equals "Bouncer". For each such element, it creates an "employee" element in the result that contains its "name" and "emp_id" attributes.

—---------------------x—----------x—---------x—----------x—-----------x—------------------