Exercise session 4

Data bases 2

XQuery Solutions

marco.abbadini@unibg.it

XQuery - Real Estate 🏠 (1/5)

```
<!ELEMENT Catalogue ( Ad*, VisitRequest* )>

<!ELEMENT Ad ( Apartment, PublishedPrice, Owners,
MinimumAcceptablePrice?, MortgageLoan?, … )>

<!ATTLIST Ad code ID #REQUIRED PublicationDate CDATA
#REQUIRED >

<!ELEMENT Owners ( Person+ )>

<!ELEMENT Person ( FirstName, LastName, Email, Telephone )>

<!ELEMENT VisitRequest ( Person, DateOfRequest,
ScheduledDateForTheVisit?, OfferedPriceAfterVisit?, … )>

<!ATTLIST VisitRequest AdRef IDREF #REQUIRED >
```

Real Estate 🏘️ (2/5)

1. *the Apartments that received offers by at least 5 different potential buyers (Email is an identifier for people).*

```
for $a in //Ad

where 4 < count( distinct-values( //VisitRequest[@AdRef=$a/@code and
./OfferedPriceAfterVisit]/Person/Email ) )

return $a/Apartment
```

Real Estate 🏠 (3/5)

2. *the Apartment that received its first visit request after the longest wait after publication.*

```
let $ranking: ( for $a in //Ad let $firstdate := min(//VisitRequest[ @AdRef = $a/@code ]/DateOfRequest )
                    let $delta := $firstdate – $a/@PublicationDate
                    where count( $firstdate ) > 0
                    order by $delta descending
                    return { $delta } { $a/Apartment } )
for $r in $ranking
where $r/item/delay = $ranking[1]/item/delay
return $r/item/apt/*
```

Real Estate 🏡 (4/5)

2. *the Apartment that received its first visit request after the longest wait after publication. (Alternative solution)*

```
let $maxdelay: max( for $a in //Ad
                        let $firstdate := min(//VisitRequest[ @AdRef = $a/@code ]/DateOfRequest )
                        where count( $firstdate ) > 0
                        return $firstdate – $a/@PublicationDate )
for $a in //Ad
let $firstdate := min(//VisitRequest[ @AdRef = $a/@code ]/DateOfRequest )
where count( $firstdate ) > 0 and $firstdate – $a/@PublicationDate = $maxdelay
return $a/Apartment
```

Real Estate 🏡 (5/5)

3. *the potential buyers who always and only offered prices below the minimum threshold fixed by the owners.*

```
for $p in //VisitRequest/Person
 where 0 = count( for $vr in //VisitRequest
                           where $vr/OfferedPriceAfterVisit >= //Ad[ @code=$vr/@ARef
                    ]/MinimumAcceptablePrice and $vr/Person/Email = $p/Email
                     return <PlusOne/>)      (<PlusOne/> is a placeholder for each offer above the treshold)
return $p
```

Medical Center 🏥 (1/4)

In the following DTD, unspecified elements contain only PCDATA

```
<!ELEMENT MedicalCenter (Patient+, Exam+)>

<!ELEMENT Patient (Name, Age, Email, HighRisk)>

<!ATTLIST PatientId ID # REQUIRED>

<!ELEMENT Exam (Date, Time, Cost, Outcome +, Doctor)>

<!ATTLIST Exam PatientId IDREF # REQUIRED>

<!ELEMENT Outcome (Parameter, Value, MinVal, MaxVal)>
```

Medical Center 🏥 (2/4)

1. *Extract in XQuery the parameter that is regular (between the reference values) with the highest frequency (for the query, consider for each parameter the percentage of "normal" outcomes)*

```
let $rank := ( for $par in distinct-values( //Parameter )
                    let $OutForThatPar := //Outcome[ Parameter = $par ]
                    let $percOK := count( $OutForThatPar [ Value >= MinVal and Value <= MaxVal ] ) div
                    count( $OutForThatPar ) * 100
                    order by $percOK
                    return <par> <name> { $par } </name> <PercOk> { $percOK } </PercOk> </par>
let $max := $rank[1]/PercOk
return $rank[ PercOk = $max ]/name
```

Medical Center 🏥 (3/4)

2. *Extract in XQuery the doctors who have only prescribed exams to patients who came out as perfectly healthy*

```
for $d in distinct-values( //Doctor )

where 0 = count( for $o in //Exam[ Doctor = $d ]/Outcome[ Value < MinVal or
Value > MaxVal ] )

return <LuckyDoctor> { $d } </LuckyDoctor>
```

Medical Center 🏥 (4/4)

3. *Extract in XQuery the patient with the largest number of values outside of the healthy range in a single exam.*

```
let $max := max( for $ex in //Exam
                         return count( $ex/Outcome[ Value > MinVal or Value < MaxVal ]
                     ) )

for $e in //Exam

where count( $e/Outcome[ Value > MinVal or Value < MaxVal ] ) = $max

return $e/../Name
```