**Employee Management -MarkLogic Assignment**

1. **Create Document for each employee and add document to employee collection**

**Query:**

xquery version "1.0-ml";

xdmp:document-load("C:\mls-projects\Employee\EMP001.xml",

<options xmlns="xdmp:document-load">

<collections>

<collection>employee</collection>

</collections>

</options>)

xdmp:document-load("C:\mls-projects\Employee\EMP002.xml",

<options xmlns="xdmp:document-load">

<collections>

<collection>employee</collection>

</collections>

</options>),

xdmp:document-load("C:\mls-projects\Employee\EMP003.xml",

<options xmlns="xdmp:document-load">

<collections>

<collection>employee</collection>

</collections>

</options>),

xdmp:document-load("C:\mls-projects\Employee\EMP004.xml",

<options xmlns="xdmp:document-load">

<collections>

<collection>employee</collection>

</collections>

</options>),

xdmp:document-load("C:\mls-projects\Employee\EMP005.xml",

<options xmlns="xdmp:document-load">

<collections>

<collection>employee</collection>

</collections>

</options>),

xdmp:document-load("C:\mls-projects\Employee\EMP006.xml",

<options xmlns="xdmp:document-load">

<collections>

<collection>employee</collection>

</collections>

</options>),

xdmp:document-load("C:\mls-projects\Employee\EMP007.xml",

<options xmlns="xdmp:document-load">

<collections>

<collection>employee</collection>

</collections>

</options>),

xdmp:document-load("C:\mls-projects\Employee\EMP001.xml",

<options xmlns="xdmp:document-load">

<collections>

<collection>employee</collection>

</collections>

</options>),

xdmp:document-load("C:\mls-projects\Employee\EMP008.xml",

<options xmlns="xdmp:document-load">

<collections>

<collection>employee</collection>

</collections>

</options>),

xdmp:document-load("C:\mls-projects\Employee\EMP009.xml",

<options xmlns="xdmp:document-load">

<collections>

<collection>employee</collection>

</collections>

</options>),

xdmp:document-load("C:\mls-projects\Employee\EMP010.xml",

<options xmlns="xdmp:document-load">

<collections>

<collection>employee</collection>

</collections>

</options>),

xdmp:document-load("C:\mls-projects\Employee\EMP011.xml",

<options xmlns="xdmp:document-load">

<collections>

<collection>employee</collection>

</collections>

</options>),

xdmp:document-load("C:\mls-projects\Employee\EMP012.xml",

<options xmlns="xdmp:document-load">

<collections>

<collection>employee</collection>

</collections>

</options>)

1. **Get the employee id grouping By Salary into following format**

**Query:**

xquery version "1.0-ml";

<result>

{for $salary in distinct-values(collection("employee")/employee/salary)

let $employee := collection("employee")/employee

return <object>

<salary>{$salary}</salary>

{

for $i in $employee

where $i/salary eq $salary

order by $i

return $i/empId

}

</object>

}</result>

1. **Get Employee Id , Name , Salary, and date of joining is greater than 2010-07-21**

**Query:**

xquery version "1.0-ml";

for $employee in collection("employee")/employee

where $employee/joiningdttm ge "2010-07-21"

return<employee>{$employee/empId,$employee/name,$employee/salary,$employee/joiningdttm}</employee>

1. **Get Employee whose Salary is Greater than 15000 in below format with tax 10%**

**Query:**

xquery version "1.0-ml";

<result>{

for $employee in collection("employee")/employee

where $employee/salary gt "15000"

return<object>

{$employee/empId,$employee/name,$employee/salary}

<tax>{$employee/salary div 100 \* 10}</tax>

</object>

}</result>

1. **Get all employees who are in Ops department but not in Service Department**

**Query:**

xquery version "1.0-ml";

for $employee in collection("employee")/employee

where $employee/department[1] eq "Ops" and $employee/department[2] ne "Service"

return $employee

1. **Retrieve all the employees whose age is greater than 27 years**

**Query:**

xquery version "1.0-ml";

for $employee in collection("employee")/employee

let $age := xs:integer(fn:days-from-duration(fn:current-date() - xs:date($employee/dob/text())) div 365.25)

where $age gt xs:integer("27")

return $employee

1. **Retrieve all the employees who are born in November or December in below format**

**Query:**

xquery version "1.0-ml";

<result>{

for $employee in collection("employee")/employee

let $date := xdmp:month-name-from-date(xs:date($employee/dob))

where $date eq "December" or $date eq "November"

return <object>

{$employee/empId,$employee/name}

<bornMonth>{$date}</bornMonth>

</object>

}</result>

1. **Retrieve All employees from Ops Department[1] and group by Department[2]**

**Query:**

xquery version "1.0-ml";

<result>{

for $dep in distinct-values(collection("employee")/employee/department[2])

let $employee := collection("employee")/employee

return <object>

<department>{$dep}</department>

{

for $i in $employee

where $i/department[1] eq "Ops" and $i/department[2] eq $dep

order by $i

return $i

}

</object>

}

</result>

1. **Write a Function which return Interval For given input start datetime, end datetime and interval duration**

**Query:**

xquery version "1.0-ml";

declare function local:interval($start as xs:dateTime,$end as xs:dateTime,$interval as xs:dayTimeDuration,$att as xs:integer)

{

if($start ne $end) then

let $startDate := $start + $interval

return

(

<interval attr="{$att}">

<startTime>{$start}</startTime>

<endTime>{$startDate - xs:dayTimeDuration("PT1M")}</endTime>

</interval>,

local:interval($startDate,$end,$interval,$att+1))

else

(<interval attr="{$att}">

<startTime>{$start}</startTime>

<endTime>{$end}</endTime>

</interval>)

};

<IntervalDetails>{

let $startDate := xs:dateTime("2008-02-01T10:00:00")

let $endDate := xs:dateTime("2008-02-02T10:00:00")

return (local:interval($startDate,$endDate,xs:dayTimeDuration("PT1H"),1))

}</IntervalDetails>