Xuery for 3 tasks

7. XQuery Equivalent for Assignment 6

（1）Average kills descending

xquery version "3.1";

let $results :=

(

for $mp in /pma\_xml\_export/database/table[@name='matchparticipant']

group by

$pId := $mp/column[@name='playerId'],

$cId := $mp/column[@name='championId']

let $gamesPlayed := count($mp)

where $gamesPlayed > 10

return

<group>

<pId>{ $pId }</pId>

<cId>{ $cId }</cId>

<gamesPlayed>{ $gamesPlayed }</gamesPlayed>

<avgKills>{

let $sum := sum($mp/column[@name='kills']/xs:decimal(.))

return round(($sum div $gamesPlayed) \* 100) div 100

}</avgKills>

<avgDeaths>{

let $sum := sum($mp/column[@name='deaths']/xs:decimal(.))

return round(($sum div $gamesPlayed) \* 100) div 100

}</avgDeaths>

<avgAssists>{

let $sum := sum($mp/column[@name='assists']/xs:decimal(.))

return round(($sum div $gamesPlayed) \* 100) div 100

}</avgAssists>

</group>

)

return

for $g in $results

let $player := /pma\_xml\_export/database/table[@name='player']

[column[@name='playerId'] = $g/pId]

let $champion := /pma\_xml\_export/database/table[@name='champion']

[column[@name='championId'] = $g/cId]

order by xs:decimal($g/avgKills) descending

return

<PlayerChampionStats>

<summonerName>{ string($player/column[@name='summonerName']) }</summonerName>

<championName>{ string($champion/column[@name='name']) }</championName>

<avgKills>{ $g/avgKills }</avgKills>

<avgDeaths>{ $g/avgDeaths }</avgDeaths>

<avgAssists>{ $g/avgAssists }</avgAssists>

<gamesPlayed>{ $g/gamesPlayed }</gamesPlayed>

</PlayerChampionStats>

（2）Average assists descending

xquery version "3.1";

let $results := (

for $mp in /pma\_xml\_export/database/table[@name='matchparticipant']

group by

$pId := $mp/column[@name='playerId'],

$cId := $mp/column[@name='championId']

let $gamesPlayed := count($mp)

where $gamesPlayed > 10

return

<group>

<pId>{ $pId }</pId>

<cId>{ $cId }</cId>

<gamesPlayed>{ $gamesPlayed }</gamesPlayed>

<avgKills>{

let $sum := sum($mp/column[@name='kills']/xs:decimal(.))

return round(($sum div $gamesPlayed) \* 100) div 100

}</avgKills>

<avgDeaths>{

let $sum := sum($mp/column[@name='deaths']/xs:decimal(.))

return round(($sum div $gamesPlayed) \* 100) div 100

}</avgDeaths>

<avgAssists>{

let $sum := sum($mp/column[@name='assists']/xs:decimal(.))

return round(($sum div $gamesPlayed) \* 100) div 100

}</avgAssists>

</group>

)

return

for $g in $results

let $player := /pma\_xml\_export/database/table[@name='player']

[column[@name='playerId'] = $g/pId]

let $champion := /pma\_xml\_export/database/table[@name='champion']

[column[@name='championId'] = $g/cId]

order by xs:decimal($g/avgKills) descending

return

<PlayerChampionStats>

<summonerName>{ string($player/column[@name='summonerName']) }</summonerName>

<championName>{ string($champion/column[@name='name']) }</championName>

<avgKills>{ $g/avgKills }</avgKills>

<avgDeaths>{ $g/avgDeaths }</avgDeaths>

<avgAssists>{ $g/avgAssists }</avgAssists>

<gamesPlayed>{ $g/gamesPlayed }</gamesPlayed>

</PlayerChampionStats>

（3）Average deaths ascending

xquery version "3.1";

let $results :=

(

for $mp in /pma\_xml\_export/database/table[@name='matchparticipant']

group by

$pId := $mp/column[@name='playerId'],

$cId := $mp/column[@name='championId']

let $gamesPlayed := count($mp)

where $gamesPlayed > 10

return

<group>

<pId>{ $pId }</pId>

<cId>{ $cId }</cId>

<gamesPlayed>{ $gamesPlayed }</gamesPlayed>

<avgKills>{

let $sum := sum($mp/column[@name='kills']/xs:decimal(.))

return round(($sum div $gamesPlayed) \* 100) div 100

}</avgKills>

<avgDeaths>{

let $sum := sum($mp/column[@name='deaths']/xs:decimal(.))

return round(($sum div $gamesPlayed) \* 100) div 100

}</avgDeaths>

<avgAssists>{

let $sum := sum($mp/column[@name='assists']/xs:decimal(.))

return round(($sum div $gamesPlayed) \* 100) div 100

}</avgAssists>

</group>

)

return

for $g in $results

let $player := /pma\_xml\_export/database/table[@name='player']

[column[@name='playerId'] = $g/pId]

let $champion := /pma\_xml\_export/database/table[@name='champion']

[column[@name='championId'] = $g/cId]

order by xs:decimal($g/avgAssists) descending

return

<PlayerChampionStats>

<summonerName>{ string($player/column[@name='summonerName']) }</summonerName>

<championName>{ string($champion/column[@name='name']) }</championName>

<avgKills>{ $g/avgKills }</avgKills>

<avgDeaths>{ $g/avgDeaths }</avgDeaths>

<avgAssists>{ $g/avgAssists }</avgAssists>

<gamesPlayed>{ $g/gamesPlayed }</gamesPlayed>

</PlayerChampionStats>

9. XQuery Equivalent for Assignment 8

xquery version "3.1";

for $mp in //table[@name="matchparticipant"]

let $championId := $mp/column[@name="championId"]

let $buildId := $mp/column[@name="buildId"]

let $champion := //table[@name="champion"][column[@name="championId"] = $championId]

let $role := $champion/column[@name="role"]

let $championName := $champion/column[@name="name"]

for $ibi in //table[@name="itembuilditems"][column[@name="buildId"] = $buildId]

let $itemId := $ibi/column[@name="itemId"]

let $itemName := //table[@name="item"][column[@name="itemId"] = $itemId]/column[@name="name"]

group by

$role,

$championName,

$itemName

let $buildCount := count($mp)

order by $buildCount descending

return

<ItemBuildPopularity>

<role>{data($role)}</role>

<championName>{data($championName)}</championName>

<itemName>{data($itemName)}</itemName>

<buildCount>{$buildCount}</buildCount>

</ItemBuildPopularity>

11. XQuery Equivalent for Assignment 10

xquery version "3.1";

let $participants :=

for $t in doc("lol-data.xml")//table[@name="matchparticipant"]

let $mid := $t/column[@name="matchId"]

let $cid := $t/column[@name="championId"]

return

<participant matchId="{ $mid }" championId="{ $cid }"/>

let $combinations :=

for $match in distinct-values($participants/@matchId)

let $matchParticipants := $participants[@matchId = $match]

for $i in 1 to count($matchParticipants) - 1

for $j in $i + 1 to count($matchParticipants)

let $champ1 := xs:integer($matchParticipants[$i]/@championId)

let $champ2 := xs:integer($matchParticipants[$j]/@championId)

where $champ1 != $champ2

let $id1 := min(($champ1, $champ2))

let $id2 := max(($champ1, $champ2))

return

<pair>

<championId1>{ $id1 }</championId1>

<championId2>{ $id2 }</championId2>

</pair>

let $grouped :=

for $pair in $combinations

group by $id1 := $pair/championId1, $id2 := $pair/championId2

let $count := count($pair)

order by $count descending

return

<combination>

<championId1>{ $id1 }</championId1>

<championId2>{ $id2 }</championId2>

<combinationCount>{ $count }</combinationCount>

</combination>

return subsequence($grouped, 1, 10)