**FAQ reeks 4: aggregatie, grouping- & having-keywords**

**Algemene opmerkingen**

* Zowel de HAVING- als de WHERE-clausule kunnen gebruikt worden om rijen weg te filteren in de resultatentabel op basis van booleaanse proposities. Het enige verschil is dat de HAVING-clausule moet (en enkel kan) gebruikt worden voor het filteren van rijen op basis van geaggregeerde waarden.
* In PostgreSQL is het niet mogelijk om aggregatiefuncties te nesten in elkaar, bv. MAX(COUNT...), AVG(COUNT...)... Om dit probleem op te lossen kan je een subquery gebruiken waarin de geaggregeerde waarde eerst wordt berekend, bv. SELECT MAX(count) FROM (SELECT COUNT ... FROM ...) subq; Let er op dat alle subqueries die gebruikt worden in een FROM-clausule steeds een alias hebben.
* Opnieuw is het mogelijk om sommige oefeningen op te lossen door enkel gebruik te maken van joins of subqueries (door, met andere woorden, geen gebruik te maken van aggregatiefuncties). Je kan echter jouw SQL-inzichten en SQL-skills sterk verbeteren wanneer je probeert om de oefeningen op te lossen door gebruik te maken van de concepten die aangerijkt worden tijdens deze reeks. En natuurlijk is het nog nuttiger om te proberen om de oefeningen op verschillende manieren op te lossen...

**Radio 4.2**

Om deze oefening correct op te kunnen lossen, is het belangrijk om te weten wat het verschil is tussen DISTINCT COUNT(...) en COUNT(DISTINCT ...).

**Radio 4.3**

De geboortedatum van de oudste presentator bekom je niet door gebruik te maken van SELECT **MAX**(birthdate) FROM presenter;

**Radio 4.5**

Om deze oefening correct op te kunnen lossen, is het belangrijk om te weten wat het verschil is tussen de WHERE-clausule en de HAVING-clausule.

**Radio 4.6**

Let er op dat je geen gebruik kan maken van geneste aggregatiefuncties.

**Radio 4.7**

We ondervinden dat er vaak problemen zijn bij het oplossen van dit soort oefeningen, zeker wanneer je een oefening van deze soort de eerste keer probeert op te lossen. Dit soort oefeningen wordt gekenmerkt door het feit dat het niet volstaat om een geaggregeerde waarde op te vragen (bv. het maximaal aantal verschillende genres op een album), maar het ook noodzakelijk is om data terug te geven die corresponderen met een geaggregeerde waarde (bv. de naam van de albums die corresponderen met het maximaal aantal verschillende genres, of in dit geval, het begintijdstip van de uitzending met de langste playlist). Hieronder zullen wij een (algemene) methode aanrijken die je steeds kan gebruiken om dit oefeningen van deze soort op te lossen.

1. Inner query 1: De eerste subquery zal de data groeperen met behulp van GROUP BY en voor elke groep een geaggregeerde waarde berekenen. In dit geval stelt de opgave om op zoek te gaan naar data in verband met de radiouitzending met de langste playlist. Deze subquery zal daarom, per uitzending, de lengte van elke playlist (het aantal liedjes) berekenen. De resultatentabel van deze subquery zal dus, voor elke uitzending, het totale aantal liedjes dat gespeeld is tijdens deze uitzending bevatten.
2. Inner query 2: De tweede subquery gaat een aggregatiefunctie toepassen op de resultatentabel die werd teruggegeven door de eerste subquery. Je gebruikt dus inner query 1 in de FROM-clausule van deze subquery. In dit geval zal deze subquery het maximaal aantal liedjes dat gespeeld is tijdens een uitzending opvragen. Deze subquery zal, met andere woorden, zoeken naar de maximale waarde die voorkomt in de resultatentabel van de eerste subquery. De resultatentabel van deze subquery zal bestaan uit exact 1 waarde, die het maximaal aantal liedjes dat gespeeld is tijdens een uitzending aangeeft.
3. Outer query: De outer query gaat data opvragen die corresponderen met de geaggregeerde waarde die werd teruggegeven als resultaat van de tweede subquery. In dit geval gaat deze query het begintijdstip van de uitzending(en), waarbij het aantal liedjes dat gespeeld is tijdens deze uitzending gelijk is aan het resultaat dat door de tweede subquery werd teruggegeven (het maximaal aantal liedjes gespeeld tijdens een uitzending), opvragen. Je dient dus opnieuw het aantal liedjes per uitzending te tellen en de rijen te bewaren waarvoor het berekende aantal liedjes gelijk is aan de waarde berekent in inner query 2. Let opnieuw op met het gebruik van WHERE en HAVING.

De methode die hierboven beschreven is kan toegepast worden om elke oefening van deze soort op te lossen. Merk ook op dat deze methode rekening houdt met ex aequos. Dit betekent dat, wanneer er tijdens meerdere uitzendingen het maximaal aantal liedjes gespeeld zijn, de resultatentabel de starttijden van alle corresponderende uitzendingen zal teruggeven.

**Radio 4.8**

Een liedje is op z’n minst een keer afgespeeld als het ook voorkomt als playlistitem.

**Radio 4.9**

Deze oefening is van dezelfde soort als Radio 4.7. Vanaf nu verwachten we dat je dit soort oefeningen zelf herkent.

**Radio 4.10**

Vergeet niet om ook artiesten toe te voegen zonder ‘pop’ liedjes. Ter herinnering, de manier om dit te doen is expliciet aangehaald tijdens reeks 2. Daarnaast is het belangrijk dat het aantal ‘pop’ liedjes van deze artiesten gelijk is aan 0 en niet aan NULL. In de introductiepresentatie van reeks 1 kan je terugvinden hoe je dit het beste kan doen.

**Radio 4.11**

Als je COUNT(...) / COUNT(...) gebruikt om het gemiddelde te berekenen, vergeet dan niet om de teller te casten naar een numeric, om resultaten van het datatype numeric (en niet van het datatype integer) te bekomen.

**Radio 4.12**

‘[Untitled track]’ stelt ook een liedje voor met een onbekende titel.

**Radio 4.15**

Deze oefening kan relatief eenvoudig opgelost worden door gebruik te maken van 2 subqueries in de FROM-clausule: een eerste subquery waarmee, per radioprogramma, het aantal gespeelde liedjes geteld worden en een tweede subquery waarmee, per radioprogramma, het aantal albums waarvan liedjes gespeeld zijn geteld worden. Door middel van een INNER JOIN kan je de resultaten van beide subqueries combineren.

**Radio 4.18**

Hou rekening met genres die gespeeld zijn door slechts 1 presentator of door geen enkele presentator. Houdt je query daarnaast rekening met liedjes die afgespeeld zijn tijdens een uitzending van een radioprogramma dat gepresenteerd wordt door meerdere presentatoren? Als een liedje afgespeeld werd tijdens een uitzending van een radioprogramma dat gepresenteerd wordt door twee mannelijke presentatoren, moet het slechts 1 bijdragen aan mcount. Als het afgespeeld werd tijdens een uitzending van een radioprogramma dat gepresenteerd wordt door zowel een mannelijke presentator als een vrouwelijke presentatrice, moet het 1 bijdragen aan mcount en moet het 1 bijdragen aan fcount. Tot slot moet er in de ORDER BY-clausule rekening gehouden worden met de waarden in mcount en fcount, na het converteren van de waarde NULL naar 0.