**FAQ reeks 1: basisfunctionaliteit**

**Algemene opmerkingen**

* Probeer steeds de opgave volledig te lezen en correct te interpreteren vooraleer je een oefening oplost. Er worden veel fouten gemaakt door het verkeerd lezen of interpreteren van opgaves (bv. *mannelijke* presentatoren in plaats van alle presentatoren, *decennium* in plaats van eeuw, *naar beneden* afronden in plaats van afronden/afkappen, *sorteren van de resultaten*...).
* Wees alert bij het lezen van foutmeldingen wanneer je foute queries probeert te verbeteren. Sommige foutmeldingen geven de onderliggende oorzaak van de fout niet geheel nauwkeurig weer. Je mag echter veronderstellen dat deze foutmeldingen steeds een correcte eerste aanduiding geven van de fout, maar vaak moet je zelf nog wat verder zoeken.
* Om een query leesbaarder te maken, worden sleutelwoorden zoals SELECT, FROM en WHERE vaak in hoofdletters geschreven, maar dit is niet noodzakelijk om een query correct te kunnen uitvoeren.
* Elke query die je uitvoert in Q’exr moet een resultatentabel teruggeven waarin elke kolomnaam *uniek* is. Normaal gezien is dit geen vereiste om SELECT-queries correct te kunnen uitvoeren in PostgreSQL, maar aangezien Q’exr jouw resultatentabel zal vergelijken met de verwachte resultatentabel, is het tijdens deze oefeningen wel noodzakelijk dat elke kolom ondubbelzinnig bepaald kan worden.
* Heel vaak bestaan er meerdere queries die fungeren als correcte oplossing voor een oefening en is het mogelijk dat een bepaalde query beter is dan een andere query in termen van performantie (snelheid, geheugenvereisten...). Het kan daarom zeer nuttig zijn om elke oefening op verschillende manieren op te proberen lossen (in het bijzonder voor de oefeningen in reeksen 2, 3 en 4). Het is echter niet zo dat we de performantie van queries gaan evalueren tijdens de SQL oefeningenlessen en evaluaties, enkel de correctheid van queries zullen wij in beschouwing nemen. Het analyseren en optimaliseren van de performantie van queries is het onderwerp van de (latere) workshop ‘Query optimalisatie’.
* Als je hulp nodig hebt, kan je in eerste instantie proberen om een antwoord te vinden in de introductiepresentaties. Deze presentaties bevatten een overzicht van *alle* concepten die je nodig hebt (of kan gebruiken) om de oefeningen op te lossen en vaak worden er queries als voorbeeld gebruikt die gelijkaardig zijn aan de queries die je zelf moet schrijven. Daarnaast zullen we ook na elke reeks een FAQ bestand op Ufora plaatsen waarin veelvoorkomende vragen, fouten, tips en tricks worden opgelijst. Indien je niet meteen een antwoord vindt in de presentaties of in de FAQ bestanden, kan je nog altijd het probleem online eens opzoeken. Let op dat je zoekt naar antwoorden met betrekking tot het PostgreSQL-dialect (en dus niet met betrekking tot MySQL, Oracle...). Vooral de PostgreSQL documentatie[[1]](#footnote-1) en de PostgreSQL tutorial[[2]](#footnote-2) kunnen wel eens erg nuttig zijn. Tot slot, wanneer je zelf geen oplossing vindt voor jouw probleem of wanneer je extra uitleg wenst, mag je steeds de assistenten aanspreken tijdens de oefeningenlessen of via e-mail.

**Radio 1.4**

* Denk eens opnieuw na over het verschil tussen de =, LIKE en ILIKE operatoren. Kan je de oefening oplossen door gebruik te maken van elk van deze operatoren? Waarom (niet)?

**Radio 1.5**

* Wees aandachtig bij het gebruik van booleaanse operatoren, logica en proposities. NOT (A AND B) is niet equivalent aan NOT A AND B en ook niet aan NOT A AND NOT B, maar wel aan NOT A OR NOT B (regels van ‘De Morgan’).

**Radio 1.6**

* Maak gebruik van de *concat* functie (of gebruik ‘||’) om voornaam en achternaam te concateneren. Hernoem de resulterende kolom door middel van aliasing.
* Heb je een spatie toegevoegd tussen de voor- en de achternaam?
* Heb je enkel *mannelijke* presentatoren geselecteerd? Kijk eerst eens na hoe *mannelijk* wordt voorgesteld in de data.
* Heb je de resultaten correct gesorteerd?

**Radio 1.7**

* Gebruik een CASE-operator in de SELECT-clausule om een correcte conditionele expressie op te bouwen. Let op voor de syntax van deze operator.

**Radio 1.8 & Radio 1.9**

* Voor deze oefeningen kan je gebruik maken van de array functies *array\_length, string\_to\_array* en *unnest*. Voorbeelden van deze functies (en hoe ze te gebruiken) zijn gegeven in de introductiepresentatie.

**Radio 1.10**

* Heb je ook alle artiesten geselecteerd waarvan de naam niet begint met ‘the’?
* Heb je bv. ‘Theatre of Tragedy’ behouden en heb je bv. ‘The Beatles’ aangepast?
* Heb je enkel het woord ‘THE’ achteraan toegevoegd of heb je ook extra spaties toegevoegd?
* PostgreSQL begint te tellen vanaf index 1 en niet vanaf index 0 in bv. de *substr* functie.
* Heb je de resultaten correct gesorteerd?

**Radio 1.11**

* Rond het jaar af (naar beneden) tot het *decennium*.
* Maak gebruik van de wiskundige functies die opgelijst staan in de introductiepresentatie. Let op dat je het verschil tussen deze functies (*round*, *trunc*, *ceil*, *floor*...) volledig begrijpt en dat je weet wanneer je een bepaalde functie (niet) kan gebruiken. Gebruik de correcte functie voor deze oefening en denk eens na of je ook andere functies kan gebruiken.
* De *round* en *trunc* functies met een index als argument geven altijd een waarde van datatype numeric terug. Waarom is dit? Let op dat je het resultaat correct cast naar een integer om het correcte resultaat te bekomen.

**Radio 1.12**

* Zorg ervoor dat je oplossing ook volgend jaar nog het correcte resultaat geeft. Werk dus niet met ‘magic numbers’ zoals 2003, 2013, 2023,... maar werk met functies/operatoren om de leeftijd van een nummer te berekenen, om met het huidige jaartal te vergelijken...
* Je kan deze oefening oplossen op verschillende manieren. Je kan, bijvoorbeeld, de leeftijd van elk album berekenen of het jaar waarin het album is uitgebracht vergelijken met het huidige jaartal.

**Radio 1.13**

* Om deze oefening op te lossen kan je eventueel gebruik maken van de ‘z BETWEEN x AND y’ operator. Deze operator is gelijk aan volgende booleaanse propositie in SQL-code: z >= x AND z <= y, dus de onder en bovengrens (resp. x en y) zijn beiden inclusief.

**Radio 1.15**

* Let opnieuw op voor de voorrangsregels van booleaanse operatoren: A OR B AND C is equivalent aan A OR (B AND C) en niet aan (A OR B) AND C.

**Radio 1.17**

* Ter herinnering, wat was het sleutelwoord nu ook alweer om alle rijen uniek weer te geven in de resultatentabel?

**Garage 1.9**

* Voer eens ‘SELECT DISTINCT type FROM intervention;’ uit en bestudeer het resultaat.

**Garage 1.11**

* Let goed op de spaties.

**Garage 1.13**

* In principe kan je werken met ... licenseplate LIKE ‘%0’ OR licenseplate LIKE ‘%2’ OR ... maar uiteraard kan het ook korter. Maak hiervoor gebruik van de wiskundige modulo operator (%).
* Met welke operatoren/functies kan je het laatste karakter van een string te selecteren?

**Garage 1.14**

* Maak opnieuw gebruik van de wiskundige functies die je al hebt bestudeerd.

1. <https://www.postgresql.org/docs/current/index.html> [↑](#footnote-ref-1)
2. <https://www.postgresqltutorial.com/> [↑](#footnote-ref-2)