H2 SQL-DATABASES

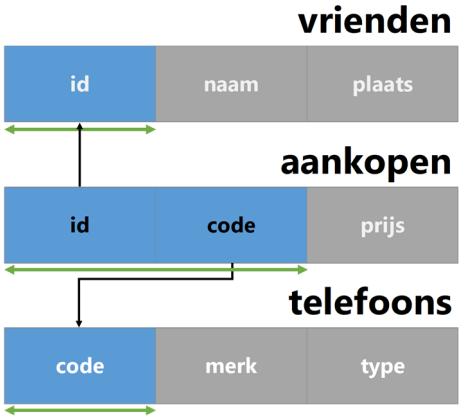
2.1 Inleiding

In hoofdstuk 1 heb je kennis kunnen maken met SQL-databases: **relationele databases** op basis van één of meerdere tabellen met rijen of **records** met dataelementen die een relatie met elkaar hebben.

In figuur 2.1 zie je een **strokendiagram** van een relationele database op basis van de casus *mobiel*:

Alan is geïnteresseerd in smartphones. Hij heeft een kleine database gemaakt waarin hij kan bijhouden welke telefoons er op de markt verschijnen. Van deze telefoons wil hij het merk en type bijhouden.

Omdat hij wil bijhouden wie van zijn vrienden welke telefoon heeft, houdt hij alle aankopen bij met de op dat moment betaalde prijs. Van zijn vrienden noteert hij alleen de naam en de woonplaats.



FIGUUR 2.1

In dit hoofdstuk komt een aantal databases vaker terug in de oefeningen. De database *mobiel* hoort daar bij. Hieronder zie je de drie tabellen met alle records:

vrienden

id	naam	plaats
1	Alan	Groningen
2	Bob	Assen
3	Christel	Heerlen
4	Daphne	Groningen
5	Eve	Heerlen
6	Frits	Delft
7	Gonny	Groningen
8	Hajar	Emmen
9	Ingo	Assen

telefoons

code	merk	type
a1	Apple	iPhone 13
a2	Apple	iPhone 14
a3	Apple	iPhone SE
g1	Google	Pixel 6a
g2	Google	Pixel 7
o1	OnePLus	11
s1	Samsung	Galaxy A23
s2	Samsung	Galaxy S23
s3	Samsung	Z flip 4
x1	Xiaomi	12 Lite
x2	Xiaomi	12X

aankopen

id	code	prijs
1	a1	819
6	a1	849
4	a2	929
8	g2	589
9	o1	869
2	s1	319
3	s1	299
8	s1	329
9	s3	929
7	x1	415
8	x2	499

FIGUUR 2.2

Als in een opgave wordt verwezen naar de casus *mobiel*, dan kun je deze bladzijde gebruiken bij het beantwoorden van vragen. De andere databases uit dit hoofdstuk worden beschreven in de opgave zelf of in een bijlage.

LET OP: in dit hoofdstuk staat niet alle theorie binnen de paragrafen uitgelegd. Je doet ook kennis en vaardigheid op door de opgaven te maken.

2.2 Een database maken met SQL

SQL staat voor *Structured Query Language*. Het is een standaardtaal die gebruikt wordt in een databasemanagementsysteem voor relationele databases. De SQL-commando's zijn op te delen in meerdere categorieën. In dit hoofdstuk behandelen we:

- DDL (*Data Definition Language*) voor het maken van databases en tabellen
- DML (*Data Manipulation Language*) voor het toevoegen, bewerken en verwijderen van records
- DQL (*Data Query Language*) voor het opvragen van data via zoekopdrachten

Het inrichten van een database binnen het databasemanagementsysteem begint bij het maken van een nieuwe, lege database. Voor de casus *mobiel* uit de inleiding hoort hierbij het **DDL**-commando *CREATE*:

```
CREATE DATABASE `mobiel`;          // maak een nieuwe database met de naam mobiel
```

Met onderstaande instructieregels wordt nu de tabel *vrienden* toegevoegd aan de database:

```
USE `mobiel`;                          // gebruik de database mobiel
CREATE TABLE `vrienden` (              // maak een tabel met de naam vrienden
    `id` int AUTO_INCREMENT PRIMARY KEY, // maak een primaire sleutel met de naam id
    `naam` text NOT NULL,              // maak een attribuut met de naam naam
    `plaats` text NOT NULL,            // maak een attribuut met de naam plaats
);
```

In bovenstaande SQL-code is te zien dat aan elk attribuut een **datatype** wordt meegegeven. Dat is verplicht. Het type *int* staat voor integer (geheel getal) en *text* voor tekst. Met *varchar(16)* maak je een attribuut dat kan bestaan uit cijfers, letters en andere symbolen met een maximale lengte die je zelf kunt kiezen (in dit geval 16).

De (niet noodzakelijke) toevoeging *NOT NULL* betekent dat het veld niet leeg mag blijven als records worden toegevoegd. Het databasemanagementsysteem controleert daar dan op. Het toevoegen van een primaire sleutel gaat met *PRIMARY KEY*. De bedoeling is dat het veld *id* niet alleen uniek is, maar ook dat de nummering bij het toevoegen van een nieuw record automatisch wordt verhoogd. Met *AUTO_INCREMENT* kan het databasemanagementsysteem dit zelf bijhouden.

Om de relaties tussen de tabellen te beschrijven, gebruiken we ook referentiesleutels. Hoe je dat doet leer je in een opgave in deze paragraaf. Het databasemanagementsysteem bewaakt dan de referentiële integriteit als records worden verwijderd, toegevoegd of gewijzigd.

Het toevoegen van records kan met het **DML**-commando *INSERT*:

```
INSERT INTO `vrienden` (`id`, `naam`, `plaats`) VALUES
(1, 'Alan', 'Groningen'),
(2, 'Bob', 'Assen');
```

Met bovenstaande SQL-code worden twee van de records in figuur 2.2 gemaakt. Merk op dat er voor het attribuut *id* geen aanhalingstekens worden gebruikt bij de ingevoerde waarden (*values*). Bij een getal doe je dat niet, bij een ander datatype wel. Natuurlijk kun je niet alleen data toevoegen, maar zijn er ook DML-commando's om data aan te passen of te verwijderen. Dat leer je bij het maken van de opdrachten.

Het belangrijkste **DQL**-commando is *SELECT*, waarover je juist meer in de volgende paragrafen leert:

```
SELECT *               // selecteer alle attributen van alle records
FROM `aankopen`        // uit de tabel aankopen
```

Opdracht 1 DDL: een database met tabellen maken

In de inleiding van dit hoofdstuk staat de casus *mobiel* beschreven. In de theorie kun je lezen hoe met SQL een database *mobiel* kan worden gemaakt met een tabel *vrienden*.

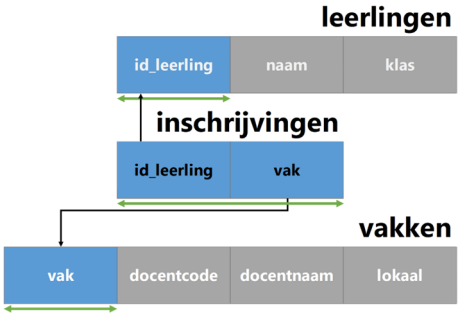
1. Geef de SQL-code waarmee de tabel *telefoons* wordt gemaakt (zie figuur 2.1). Zorgt dat de attributen *code* en *type* het datatype *varchar* krijgen met maximaal drie respectievelijk 16 tekens. Het veld *merk* krijgt het tekstformaat.
2. Voeg SQL-code toe waarmee de eerste twee records van de tabel *telefoons* worden toegevoegd.

De database *mobiel* is onderdeel van de online omgeving. We kunnen daarom het SQL-bestand waarmee de volledige database is gemaakt bekijken.

3. Open *mobiel.sql* (in de map *databases*) in je editor en controleer jouw antwoorden op bovenstaande twee vragen.
4. De tabel *aankopen* bevat een samengestelde sleutel. Met welke SQL-regel is deze gemaakt?
5. In figuur 2.1 zie je twee referentiesleutels (*foreign key*). Met welke SQL-regels zijn die gemaakt?

Opdracht 2 DDL & DML: examentraining

In figuur 2.3 zie je het eindresultaat van de database *examentraining* volgens het strokendiagram van figuur 1.23.



FIGUUR 2.3

6. Open het bestand *examentraining.sql* aan in de map *databases* van de werkomgeving.
7. Voeg een coderegel toe waarmee de database *examentraining* kan worden gecreëerd.
8. Gebruik de volgende regels om in de database *examentraining* de tabel *vakken* aan te maken:

```
USE `examentraining`;
CREATE TABLE `vakken` (
    `vak` varchar(16) PRIMARY KEY,
    `docentcode` varchar(3) NOT NULL,
    `docentnaam` text NOT NULL,
    `lokaal` tekst
);
```

9. Voeg regels toe, zodat de tabel *leerlingen* wordt gemaakt. Bedenk zelf voor elk veld het juiste datatype. Vergeet niet op de primaire sleutel toe te kennen.
10. Ga naar de startpagina van de werkomgeving en kies bij databasebeheer voor *Importeren*.
11. Ga nu naar de *Querier*. Controleer of de database *examentraining* in de lijst staat.
12. Selecteer de database en controleer of de tabellen *vakken* en *leerlingen* correct worden getoond.
13. Voeg SQL-coderegels toe om de tabel *inschrijvingen* te maken. Kies daarna opnieuw voor *Importeren* en controleer of nu alle drie de tabellen correct worden weergegeven.

Stel dat jij je voor twee vakken hebt ingeschreven voor de examentraining bij jou op school.

14. Voeg SQL-regels (DML) toe aan *examentraining.sql* waarmee alle benodigde data aan de drie tabellen wordt toegevoegd.
15. Ga naar de *Querier*. Vul bij de *Invoer* de query `SELECT * FROM vakken` in en klik op *Uitvoeren*.
16. Gebruik query's om ook de inhoud van de andere twee tabellen op te vragen met de *Querier*.
17. Voer in: `DESC inschrijvingen`. Wat is het resultaat?

Opdracht 3 DML: data manipuleren

In de theorie heb je kennis gemaakt met *INSERT*, waarmee je data in een tabel kunt invoegen. In deze opgave leer je om aanpassingen te doen aan data in de database, of gegevens te verwijderen.

18. Open de database *mobiel* in de *Querier*.
19. De *Querier* beschermd de database tegen het per ongeluk wijzigen of verwijderen van data en databases. Zorg dat het vinkje voor *Alleen lezen* uitstaat, zodat we wijzigingen kunnen aanbrengen.

In de theorie wordt een *INSERT*-opdracht getoond, waarbij Alan en Bob een *id* met een waarde meekrijgen. Echter, doordat bij de definitie van de tabel gebruik gemaakt is van *AUTO_INCREMENT*, is het niet nodig om dit nummer mee te geven. In plaats daarvan mag je *null* invullen.

20. Wat is de betekenis van *null*? Gebruik eventueel hoofdstuk 1 of het internet.
21. Voeg jezelf toe aan de database *vrienden*. Vul bij het veld *id* geen waarde in, maar gebruik *null*.
22. Gebruik *SELECT* om de inhoud van de tabel *vrienden* te tonen. Welk *id* heb je gekregen?
23. Voeg jouw mobiele telefoon toe aan de tabel *telefoons* en controleer met SELECT of dat gelukt is.

Je hebt nu jezelf en jouw telefoon toegevoegd aan de database. Maar de informatie dat jij de eigenaar bent van die telefoon ontbreekt nog.

24. Voeg deze ontbrekende data toe. Controleer of de data goed is toegevoegd aan de tabel.

Je hebt nu data toegevoegd aan de database. Maar wat nu als je een foutje had gemaakt? Ook daarvoor bestaan DML-commando's. Hieronder zie je twee regels voor het wijzigen respectievelijk verwijderen van data:

```
UPDATE vrienden SET naam = 'Jos', plaats = 'Amersfoort' WHERE id = 10
DELETE FROM aankopen WHERE id = 10
```

MERK OP: tot nu toe hebben we tabel- en veldnamen altijd met aanhalingstekens zoals `` `telefoons` `` geschreven. Hoewel dit gebruikelijk is, is het niet perse noodzakelijk. Voor het gemak laten we het van nu af aan dan ook weg. De aanhalingstekens bij teksten zoals `'Amersfoort'` moeten wel blijven!

25. Hierboven staan twee SQL-regels. Leg voor beide in je eigen woorden uit wat ze betekenen.
26. Voer de UPDATE-regel uit en controleer of het resultaat overeenkomt met jouw uitleg.
27. Voer de DELETE-regel uit. Controleer ook nu of het overeenkomt met jouw voorspelling.
28. Alan is verhuisd van Groningen naar Zwolle. Verwerk dit met behulp van een SQL-opdracht.
29. ☆ Verwijder alle records uit de tabel *aankopen* met een prijs onder de 500 euro.

Je hebt nu veel wijzigingen aangebracht in de database *mobiel*. Gelukkig kunnen we de database in zijn oorspronkelijke staat herstellen.

30. Ga naar de startpagina van de werkomgeving en klik op *Herstel standaard databases*.

☆ Opdracht 4 DDL: de databasestructuur bewerken

In de vorige opgave werd data bewerkt, maar uiteraard kun je ook de tabellen van de database zelf bewerken. Daarmee definieer je de database (en dus ook de records die je erin kunt plaatsen): DDL.

Gebruik het internet (b.v. w3schools) om zelf uit te zoeken hoe je de volgende opdrachten uit kunt voeren:

31. Voeg het attribuut *werkgeheugen* toe aan de tabel *telefoons* met datatype *int*. Het veld mag leeg blijven.
32. Verander het attribuut *naam* in de tabel *vrienden* naar *voornaam*.
33. Verwijder alle records (met één instructieregel) uit de tabel *aankopen*.
    HINT: gebruik *TRUNCATE*.
34. Ga naar de startpagina van de werkomgeving en klik op *Herstel standaard databases*.

2.3 Data opvragen uit de database: query's

In de vorige pagina hebben we vooral gekeken naar de vraag hoe je met SQL een database kunt aanmaken, de databasestructuur kunt bewerken en data kunt toevoegen, aanpassen of verwijderen. Veel data is opgeslagen in een database, omdat je de data later weer wil gebruiken. Je wil de data dan kunnen opvragen en soms data combineren om tot nieuwe kennis te komen. Daarom zijn de komende paragrafen gewijd aan **DQL**-zoekopdrachten, zoals het voorbeeld uit de vorige paragaaf:

```
SELECT *                    // selecteer alle attributen van de records
FROM aankopen               // uit de tabel aankopen
```

Als bovenstaande zoekopdracht of **query** wordt uitgevoerd, wordt de volledige tabel *aankopen* getoond. Dat is niet altijd praktisch. Vaak wil je een beperkt aantal attributen laten zien van een beperkt aantal **records**. Je stelt dan een eis *waaraan* de getoonde records moeten voldoen met de toevoeging WHERE:

```
SELECT id, naam             // maak een resultaattabel met de kolomnamen id en naam
FROM vrienden               // uit de tabel vrienden
WHERE plaats = 'Groningen'  // met records waarvoor geldt dat de plaats Groningen is
```

De combinatie van SELECT, FROM en WHERE is tamelijk standaard bij het opstellen van een query. Voor de overzichtelijkheid zetten we de drie delen hier op drie aparte regels, maar dat is niet perse noodzakelijk.

De voorbeeldquery hierboven bevat een enkele eis. Met **logische operatoren** zoals AND, OR en NOT kun je meerdere eisen met elkaar combineren, zoals met de volgende query

```
SELECT type
FROM telefoons
WHERE merk = 'Xiaomi' OR merk = 'Google'
```

waarmee een overzicht wordt gemaakt van alle types telefoons in de database die van het merk Xiaomi of Google zijn. In deze paragraaf staan opdrachten om hier meer mee te oefenen.

Opmerkingen tot slot: bij de komende opdrachten gebruiken we steeds de *Querier* tenzij anders aangegeven. Dit niet steeds opnieuw bij de opdracht vermeld.
We kiezen ervoor om alle SQL-commando's met hoofdletters te schrijven, zodat het onderscheid tussen SQL en tabelnamen en -attributen nog duidelijk wordt, maar kleine letters mag ook!

Opdracht 5 mobiel: basisquery's

Bij deze opgave gebruiken we de database *mobiel* en bekijken we de volgende query's:

a. `SELECT naam FROM vrienden WHERE plaats = 'Assen'`
b. `SELECT code, type FROM telefoons WHERE merk = 'OnePlus' OR merk = 'Xiaomi'`
c. `SELECT code, type FROM telefoons WHERE merk = 'OnePlus' AND merk = 'Xiaomi'`
d. `SELECT naam FROM vrienden WHERE plaats = 'Groningen' AND id > 4`
e. `SELECT naam FROM vrienden WHERE plaats = 'Groningen' AND id <= 4`
f. `SELECT naam FROM vrienden WHERE plaats = 'Groningen' AND NOT id < 4`
g. `SELECT id, code FROM aankopen WHERE prijs IS NULL`

35. Beschrijf in een Nederlandse zin voor elk van de query's welke vraag ze aan de database stellen.
36. Gebruik de tabellen in figuur 2.2 om (dus zonder *Querier*) uit te zoeken wat de uitkomst is.
37. Controleer jouw antwoord op de vorige vraag door de query's via de *Querier* uit te voeren.

Opdracht 6 top 2000: basisquery's

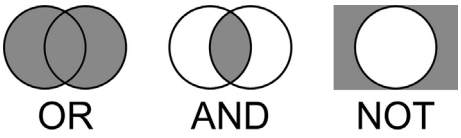
In deze opdracht maken we voor het eerst kennis met de top-2000 database. Hiervan bestaan twee versies. In deze opgave werken we met versie 1. Tenzij anders vermeld is het de bedoeling dat je een query gebruikt om de vraag te beantwoorden. Als antwoord noteer je uiteindelijk de gebruikte query.

38. Lees de beschrijving van de database *top 2000 versie 1* in de bijlage.
39. Selecteer *top 2000 v1* in de *Querier* en bekijk de inhoud van de database door de query `SELECT * FROM top2000` uit te voeren. Uit hoeveel records bestaat deze database?
40. Selecteer de titel en de naam van de artiest van het nummer dat in 2020 op positie 1 stond.
41. Welke nummers van Adele (figuur 2.4) die gemaakt zijn in 2011 hebben een notering gehad in de top 2000? Zorg dat in de resultaattabel alleen de titel en het jaar van uitgave te zien zijn.
42. Selecteer alle artiestennamen die een track met de titel *Sorry* in de top 2000 hebben gehad.
43. Gebruik query *g* van de vorige opgave om alleen die artiesten te selecteren die wel een liedje met de titel *Sorry* hebben, maar in de editie van 2018 niet in de top 2000 stonden.
44. Keer de selectie van de vorige vraag om: welke artiesten stonden juist wel in 2018 in de lijst met een nummer dat *Sorry* heet?

FIGUUR 2.4

Opdracht 7 top 2000: logische operatoren

In de theorie en de vorige opdrachten heb je al gebruik moeten maken van *AND*, *OR* en *NOT* om een combinatie van eisen aan de data die je selecteert. In figuur 2.5 is de betekenis van deze operatoren met Venndiagrammen weergegeven.



FIGUUR 2.5

Naast de hierboven genoemde woorden, heb je in de theorie kunnen zien dat je ook gebruik kunt maken van wiskunde tekens zoals > (groter dan), <= (kleiner dan *of gelijk aan*) en andere varianten hierop.

45. Beschrijf in woorden wat in algemene zin (dus geen records benoemen) de uitkomst is van de query
    ```
    SELECT titel, artiest FROM top2000 WHERE ed2021 <= 10 AND ed2020 > 10
    ```
46. Controleer jouw beschrijving door de query uit te voeren.
47. De query
    ```
    SELECT titel, artiest FROM top2000 WHERE ed2019 <= 10 OR ed2018 > 10
    ```
    zorgt voor een resultaattabel met 2000 records. Controleer dit en leg vervolgens uit of dit aantal vooraf te voorspellen zou zijn geweest.
48. Selecteer alle records met liedjes van voor 1970 die in de editie van 2019 wel in de top 2000 stonden maar een jaar later niet.
49. Geef een overzicht van alle nummers van de artiesten *Harry Styles* en *Taylor Swift*.
50. Maak een resultaattabel met de kolommen *artiest*, *titel* en *uit* van de nummers van *Harry Styles* en van *Taylor Swift* die zijn uitgekomen in 2015 of later. Kijk goed naar het resultaat: staan er geen liedjes van voor 2015 in jouw lijst? (Zie eventueel de volgende opdracht.)

Opdracht 8 top 2000: werken met haakjes

Gokje: bij de laatste vraag van opdracht 7 heb je (ongeveer) de volgende query geprobeerd:

```
SELECT artiest, titel, uit FROM top2000
WHERE uit >= 2015 AND artiest = 'Harry Styles' OR artiest = 'Taylor Swift'
```

Als dat zo is, heb je kunnen zien dat het nummer *Shake it Off* van *Taylor Swift* tot de resultaten behoort. Dit nummer is uit 2014 en dat mocht niet! Hoe kan het dat deze toch in de resultaattabel staat? Het huidige resultaat geeft *alle nummers uit 2015 of later van Harry Styles of nummers van Taylor Swift*. Als we haakjes toevoegen aan de query is dit beter te zien:

```
SELECT artiest, titel, uit FROM top2000
WHERE (uit >= 2015 AND artiest = 'Harry Styles') OR artiest = 'Taylor Swift'
```

Er geldt nu niet voor alle liedjes dat ze uit 2015 of later moeten komen. Dat geldt wel als we zeggen:

```
SELECT artiest, titel, uit FROM top2000
WHERE uit >= 2015 AND (artiest = 'Harry Styles' OR artiest = 'Taylor Swift')
```

51. Voer de laatste query hierboven uit en stel vast dat er nu geen records verschijnen van voor 2015.
52. Geef de titels van alle tracks van de artiest *Queen* die in 2019 of 2020 in de top 50 van de top 2000 stonden.
53. Geef de titel en de positie in de top 2000 van 2018 van alle tracks van *The Beatles* die in 1963, 1964 of 1965 zijn uitgebracht.
54. Geef de titel en de positie in de top 2000 van 2018 van alle tracks van *The Beatles* die niet in 1963, 1964 of 1965 zijn uitgebracht.

☆Opdracht 9 top 2000: LIKE met jokers / wildcards

Tot nu toe hebben we query's gemaakt waarbij we steeds de exacte zoekterm (zoals de artiestennaam) weten. Maar wat nu als je een naam niet exact weet? In dat geval gebruik je de operator *LIKE* in combinatie met de **wildcard**-symbolen `%` of `_`. De twee tekens `%` en `_` worden ook wel **jokers** genoemd.

55. Voer de query `SELECT artiest FROM top2000 WHERE artiest LIKE 'y%'` uit. Gebruik de resultaattabel om uit te leggen wat de betekening van `LIKE 'y%'` is.
56. Beschrijf in je eigen woorden wat de uitkomst van de query de query `SELECT artiest FROM top2000 WHERE artiest LIKE '%y%'` zal zijn. Voer de query uit en controleer je beschrijving.
57. Hoeveel liedjes met de term *Christmas* in de titel staan er in de database?
58. Zijn er records waarvoor geldt dat zowel de artiestennaam als de titel de letter *q* bevat? Zo ja, hoeveel?
59. Voer de query `SELECT artiest FROM top2000 WHERE artiest LIKE '__u_'` uit. Gebruik de resultaattabel om uit te leggen wat het verschil is tussen `%` en `_`.
60. Vervang het gedeelte na *LIKE* uit de vorige vraag door `'__u%'`. Wat is het resultaat?
61. Hoeveel artiestennamen met precies twee symbolen staan er in de database?
62. Khalid is fan van *Ed Sheeran* en vraagt zich af of er meer artiesten zijn met als voornaam *Ed*. Hij gebruikt `'Ed%'`. Leg uit dat dit niet tot het gewenste resultaat leidt. Hoe los je dit probleem op?
63. Met welke collega artiesten heeft de artiest *Snelle* (figuur 2.6) nummers in de top 2000 staan?

FIGUUR 2.6

2.4 Resultaten ordenen en statistiek bedrijven

Je hebt kennis gemaakt met het schrijven van zoekopdrachten. Misschien heb je al ervaren dat de resultaattabellen er soms wat onhandig uitzien. Soms staan resultaten dubbel vermeld, of is de ordening onhandig. En bij opdracht 9 werd een aantal keer de vraag gesteld *hoeveel* (liedjes of artiesten) ergens van was. Daarbij kon je wel een zoekopdracht maken, maar moest je nog wel zelf tellen. Dat kan beter.

Stel dat we op basis van de database *mobiel* een overzicht willen hebben van alle woonplaatsen waarin vrienden wonen. Op basis van de vorige paragraaf is `SELECT plaats FROM vrienden` dan een optie. Het probleem hierbij is, dat er veel vrienden eenzelfde woonplaats hebben, zodat plaatsen dubbel worden vermeld. Bovendien staan de namen door elkaar. We willen een alfabetisch gesorteerde lijst. Dat kan met:

```
SELECT DISTINCT plaats        // selecteer unieke attribuutwaarden uit plaats
FROM vrienden                 // uit de tabel vrienden
ORDER BY plaats ASC           // sorteer de kolom plaats oplopend (alfabetisch)
```

Met de toevoeging DISTINCT kun je unieke items selecteren. Dit voorkomt dubbele identieke waarden in de resultaattabel. Het **ordenen** of **sorteren** van de resultaattabel doe je met ORDER BY. Je kiest dan een kolomnaam van de resultaat, gevolg door ASC (Engels: *ascending*: oplopend) of DESC (Engels: *descending*: aflopend). Het verbeterde resultaat zie je in figuur 2.7.

plaats
Assen
Delft
Emmen
Groningen
Heerlen

FIGUUR 2.7

Je kunt met behulp van SQL aan een databasemanagementsysteem vragen om statistiek te bedrijven op de data in een database, zoals voor de vraag: *hoeveel vrienden staan er in de database?* Om hier rechtstreeks antwoord op te krijgen, gebruik je COUNT om de bijbehorende records te tellen:

```
SELECT COUNT(*) AS aantal     // tel het aantal records
FROM vrienden                 // van de tabel vrienden
```

Merk op dat COUNT een **functie** is. Wat je telt moet daarom tussen haakjes. Als je records wilt tellen, kun je de asterix (*) gebruiken. Er is geen attribuut *aantal* in de database. Met behulp van de **alias** *aantal* verwijs je naar de uitkomst van de COUNT-functie en zorg je dat de resultaattabel een kolomnaam heeft.

Wat moeten de leerlingen halen om een voldoende te blijven staan op hun rapport? Hoeveel stappen heb ik vandaag gezet (figuur 2.8)? Wat is mijn gemiddelde hartslag als ik lig te slapen?

FIGUUR 2.8

Bij het gebruik van databases spelen berekeningen en statistiek vaak een centrale rol. Dit is een overzicht van SQL-functies voor statistiek:

- `COUNT` telt het aantal records van een selectie
- `SUM` telt attribuutwaarden van records bij elkaar op
- `AVG` neemt het gemiddelde van de attribuutwaarden
- `MAX` zoekt de grootste waarde in een selectie
- `MIN` zoekt de kleinste waarde in een selectie

Deze statistische functies kun je gebruiken op een volledige selectie van records, maar het is ook mogelijk om de selectie op te delen in meerdere groepen en daar statistiek op te bedrijven. Hoe dat werkt, leer je tijdens het maken van de opdrachten van deze paragraaf.

Opdracht 10 mobiel: basisquery's

Bij deze opgave gebruiken we de database *mobiel* en bekijken we de volgende query's:

1. `SELECT DISTINCT merk FROM telefoons ORDER BY merk ASC`
2. `SELECT MIN(prijs) AS best FROM aankopen WHERE code = 's1'`
3. `SELECT MAX(prijs) AS duur FROM aankopen WHERE NOT (code = 'a1' OR code = 'a2')`
4. `SELECT DISTINCT plaats FROM vrienden WHERE plaats LIKE '%en' ORDER BY plaats ASC`

64. Beschrijf in een Nederlandse zin voor elk van de query's welke vraag ze aan de database stellen.
65. Gebruik de tabellen in figuur 2.2 om (dus zonder *Querier*) uit te zoeken wat de uitkomst is.
66. Controleer jouw antwoord op de vorige vraag door de query's via de *Querier* uit te voeren.

Opdracht 11 top 2000: ORDER BY en DISTINCT

Voor deze opdracht gebruiken we *top 2000 versie 1*.

67. Toon alle door *Muse* (figuur 2.9) gemaakte liedjes met een notering in de top 2000, op volgorde van het jaar van uitgave. Het resultaat moet alleen de titel en het jaar bevatten, met het meest recente nummer bovenaan.
68. Geef alle door *Muse* gemaakte liedjes met een notering in de top 100 van de top-2000 van 2020. Geef alleen de titel en de positie, oplopend gesorteerd op positie.
69. Is het mogelijk om de query van de vorige vraag aan te passen, zodat wordt gesorteerd op het jaar van uitgave (in plaats van positie), zonder dat de kolom *uit* in de resultaattabel verschijnt?

FIGUUR 2.9

70. Maak een alfabetische lijst van alle artiesten die een notering in de top-20 van de top-2000 van 2018 hadden. Elke artiestennaam mag maar één keer voorkomen. Hoeveel verschillende artiestennamen vind je in de lijst?

Opdracht 12 mobiel: statistische functies

In de theorie staan een aantal functies waarmee je statistiek kunt bedrijven op een database. Gebruik deze functies bij de volgende opdrachten.

71. Tel het aantal vrienden dat in Groningen of Assen woont. Geef het resultaat in een tabel met de kolomnaam *aantal*.
72. Voorspel met een Nederlandse zin wat het resultaat zal zijn van de query:
    `SELECT COUNT(DISTINCT plaats) AS N FROM vrienden`
73. Voer bovenstaande query uit en gebruik het resultaat om vast te stellen of jouw voorspelling juist was.
74. Wat is de gemiddelde prijs die vrienden hebben betaald voor een iPhone? Gebruik de codes *a1* en *a2* in je query.
75. Hoeveel geld hebben alle vrienden samen aan telefoons uitgegeven volgens deze database? Zorg voor een resultaattabel met de kolom *totaal*.
76. Voorspel met een Nederlandse zin wat het resultaat zal zijn van de query:
    `SELECT COUNT(DISTINCT plaats) AS N FROM vrienden`
77. Voer bovenstaande query uit en gebruik het resultaat om vast te stellen of jouw voorspelling juist was.
78. De functies `MAX` en `MIN` geven de maximale respectievelijk minimale waarde van een attribuut in een dataset, zoals *de hoogste prijs*. Zoek uit of deze functies ook werken attributen met tekst. Zo ja, leg dan uit hoe de functies werken.

Opdracht 13 top 2000: ordenen en statistiek

Voor deze opdracht gebruiken we opnieuw *top 2000 versie 1*.

79. De zanger *Phil Collins* (figuur 2.10) maakt ook deel uit van de band *Genesis*. Maak een overzicht van alle titels die van *Phil Collins* en van *Genesis* in de database staan. De resultaattabel moet de naam van de artiest (alfabetisch gesorteerd), de titel van het liedje en het jaar waarin het liedje is uitgekomen geven.
80. Als het goed is heb je bij de vorige vraag voor de sortering gekozen voor `ORDER BY artiest ASC`. Breid deze regel uit naar `ORDER BY artiest DESC, uit ASC`. Beschrijf vervolgens het resultaat.
81. Voorspel wat er gebeurt wanneer je de -regel vervangt door `ORDER BY 1 DESC, 2 ASC`. Bekijk vervolgens het resultaat.

FIGUUR 2.10

Bekijk de volgende query's:

```
SELECT COUNT(artiest) AS x FROM top2000 WHERE NOT ed2021 IS NULL
SELECT COUNT(DISTINCT artiest) AS y FROM top2000 WHERE NOT ed2021 IS NULL
```

82. Wat is de uitkomst van de waarden *x* en *y*? Als je denkt dat het niet precies te voorspellen is, mag je een schatting doen.
83. Wat is de betekenis van de uitkomsten *x* en *y* en dus van de query's?
84. Leg uit waarom de toevoeging `WHERE NOT ed2021 IS NULL` voor deze query's noodzakelijk is.

We willen graag weten welke liedjes het meest gestegen of gedaald zijn in de top-2000. We proberen de volgende query:

```
SELECT artiest, titel, (ed2020 - ed2019) AS verschil
FROM top2000 ORDER BY 3 DESC
```

FIGUUR 2.11

85. Bekijk het resultaat van deze query. Wat is de exacte betekenis van *verschil*? Wat kun je zeggen over het liedje (en de artiest; zie figuur 2.11) dat bovenaan staat?
86. Navigeer naar het einde van de resultaattabel. Daar zie je dat de 3e kolom voor heel veel liedjes geen waarde bevat. Geef hiervoor de verklaring.
87. Breid je zoekopdracht uit. Zorg dat de in de vorige opdracht bedoelde records niet meer worden getoond.

☆ Opdracht 14 oudste vrouwen: werken met datums

In deze paragraaf hebben we statistiek bedreven met getalswaarden en teksten en getallen gesorteerd. In speciaal dataformaat is de **datum**. Dat bestuderen we aan de hand van de casus *oudste vrouwen*.

88. Lees de beschrijving van de database *oudste vrouwen* in de bijlage.
89. Geef een overzicht van records van vrouwen die ouder dan 116 zijn geworden, gesorteerd op leeftijd met de oudste bovenaan.
90. Bepaal de gemiddelde leeftijd van de vrouwen in de database die voor 1880 zijn geboren.
91. Voer de volgende query uit:
    `SELECT DATEDIFF(sterfdatum,geboortedatum) AS L FROM vrouwen ORDER BY 2 DESC`
    en bekijk het resultaat. Wat doet de functie DATEDIFF? Gebruik eventueel het internet.
92. Wat is de betekenis en eenheid van de kolom *L*?
93. Opvallend veel oude vrouwen komen uit Japan en Amerika. Selecteer alle vrouwen die een leeftijd van meer dan 116 met daarin de kolom *L* omgerekend naar jaren in de vorm 116.2 (jaar) die juist niet uit Japan of Amerika komen.

2.5 Meer statistiek: groeperen

In de theorie van de vorige paragraaf staat beschreven hoe je het totale aantal vrienden in de vriendentabel van de database *mobiel* kunt bepalen. Dat is een statistiek die betrekking heeft op alle records in de tabel.

Stel nu dat we *het aantal vrienden per woonplaats* willen weten. Dat kan door per woonplaats een query op te stellen (als je eerst hebt uitgezocht welke woonplaatsen er allemaal zijn), maar het kan ook in één keer. Het resultaat in figuur 2.12 is bereikt met de query:

```
SELECT plaats, COUNT(*) AS aantal
FROM vrienden
GROUP BY plaats              // maak groepjes per plaats
```

plaats	aantal
Groningen	3
Assen	2
Heerlen	2
Delft	1
Emmen	1

FIGUUR 2.12

Met de SQL-opdracht GROUP BY geef je aan dat het je records met dezelfde attribuutwaarden wilt groeperen. In dit geval: maak groepjes van records op basis van dezelfde plaats. Het tellen met de functie COUNT wordt vervolgens per groepje (dus per woonplaats) uitgevoerd.

Nu we dit resultaat hebben bereikt, kunnen we nog een stap verder gaan, zoals voor de zoekopdracht: *Geef een overzicht van alle plaatsen waar meer dan één vriend woont*. We hebben geleerd om eisen te stellen aan records met behulp van WHERE. Maar merk op: het aantal vrienden van een woonplaats is geen attribuut (-waarde) in de database. Deze waarde staat niet (rechtstreeks) in de database, maar is een eigenschap van de groepjes die we hebben gemaakt.

Om eisen te stellen aan de groepswaarden die worden getoond, gebruik je:

```
SELECT plaats, COUNT(*) AS aantal FROM vrienden
GROUP BY plaats              // maak groepjes per plaats
HAVING aantal > 1            // en toon groepen waarvoor geldt: aantal > 1
```

Het commando HAVING kun je dus uitleggen als *(groepjes) die de eigenschap hebben dat*. Vervolgens kun je tellen met de COUNT-functie zoals we hier hebben gedaan. Maar ook de overige statistische functies die je in de vorige paragraaf hebt geleerd, kun je toepassen op de groep.

Opdracht 15 mobiel: groeperen

Bij deze opgave gebruiken we de database *mobiel*.

94. Bekijk figuur 2.13. Geef de query waarmee deze resultaattabel is verkregen. Vergeet niet om ook de ordening aan te brengen.
95. Pas de query van de vorige vraag aan, zodanig dat alleen merken met precies twee modellen worden getoond.

merk	N
Apple	3
Samsung	3
Google	2
Xiaomi	2
OnePlus	1

FIGUUR 2.13

De volgende vragen gaan over de query:

```
SELECT code, MIN(prijs) AS bestBuy FROM aankopen
GROUP BY code ORDER BY bestBuy ASC
```

96. Leg in woorden uit (dus niet de resultaattabel geven) wat de uitkomst van deze query is.
97. Voer de query uit. (Had je gelijk?)
98. Maak een overzicht van de gemiddelde prijs en de code van mobieltjes waarvoor geldt dat het gemiddelde aankoopbedrag van dat mobieltje minder dan € 600,- is.

Opdracht 16 top oudste vrouwen: groeperen

Voor deze opdracht gebruiken we de database *oudste vrouwen*.

Van de in 1997 overleden Française Jeanne Calment (figuur 2.14) wordt aangenomen dat ze de oudste mens is die ooit geleefd heeft (waarvan de geboorte- en sterdatum onomstreden zijn). Ze werd ruim 122 jaar. In de database staat slechts één vrouw uit Frankrijk.

FIGUUR 2.14

99. Geef een overzicht van landen met het aantal in de database opgenomen oudste vrouwen per land, gesorteerd op het land met de meeste vermeldingen.
100. Verfijn het resultaat door alleen landen op te nemen met meer dan één vermelding in de database.

Japan heeft wereldwijd (relatief gezien) de meeste mensen die meer dan 100 jaar oud worden.

101. Maak een query waarmee de gemiddelde leeftijd van de oudste vrouwen uit Japan kan worden berekend.

Je hebt bij de vorige vraag en eerdere opdrachten met het gemiddelde waarschijnlijk gemerkt, dat dit standaard met veel cijfers achter de komma wordt weergegeven. Gelukkig kunnen we resultaten netter weergeven, door deze af te ronden met de ROUND-functie. Voorbeeld: ROUND(1.174,1) geeft 1.2 als antwoord omdat er op één cijfer achter de komma wordt afgerond. ROUND(1.174,2) geeft 1.17.

102. Verbeter de query van de vorige vraag, door de gemiddelde leeftijd op één decimaal achter de komma af te ronden.
103. Geef een overzicht van de gemiddelde leeftijd van de oudste vrouwen per land, afgerond op twee decimalen, oplopend gesorteerd op de gemiddelde leeftijd voor landen met meer dan één vrouw in de database.

Opdracht 17 top 2000: groeperen

104. Maak een overzicht van artiestennamen met het aantal liedjes dat van hen in de top 2000 van 2019 stond. Natuurlijk sorteer je van hoog naar laag op aantal. Welke artiest (figuur 2.15) komt het meest voor?
105. Maak een overzicht van artiesten die met 20 of meer liedjes in de database staan vermeld.
106. Veel liedjes hebben een unieke titel, maar dat geldt niet voor alle liedjes. Maak een overzicht van titels die meer dan twee keer in de database voorkomen.

FIGUUR 2.15

Met de commando's die we hebben geleerd kunnen we complexe query's maken die veel instructies combineren, zoals:

```
SELECT artiest, AVG(ed2021) AS gem FROM top2000
WHERE ed2021 <= 100 GROUP BY artiest
HAVING COUNT(*) >= 4 ORDER BY gem ASC
```

MERK OP we gebruiken hier achter HAVING geen alias maar een andere functie (COUNT) dan de functie die we bij het selecteren hebben gebruikt (AVG).

107. Beschrijf in algemene bewoordingen wat de uitkomst van de query zal zijn.
108. Voer de query uit en vergelijk jouw omschrijving met de uitkomst.
109. Onder de query staat een MERK OP over het gebruik van functies bij HAVING. Noem een reden waarom je deze techniek zou willen gebruiken.