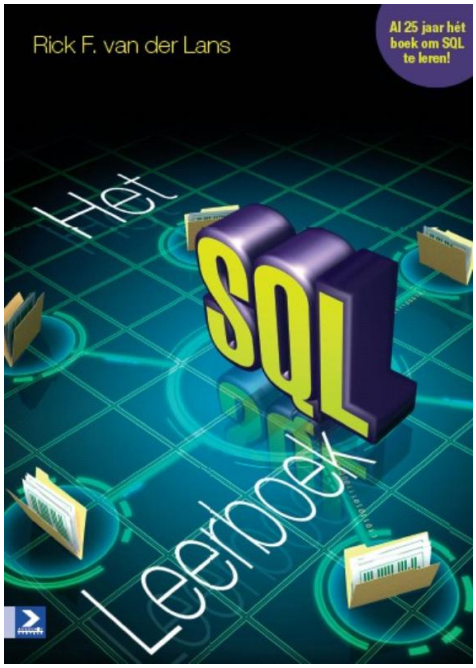
Het SQL Leerboek – zevende editie
Antwoorden op Opgaven



Auteur: Rick F. van der Lans
Versie: 1.0
Datum: Februari 2012

Hoofdstuk 1

Antwoorden

1.1 Antwoorden hoofdstuk 5

5.1

1. goed; float-datatype
2. fout; omdat vóór en achter een alfanumerieke constante aanhalingstekens moeten staan
3. goed; alfanumeriek datatype
4. fout, omdat er zich tekens buiten de aanhalingstekens van de alfanumerieke constante bevinden
5. goed; alfanumeriek datatype
6. goed; integer-datatype
7. goed; alfanumeriek datatype
8. goed; alfanumeriek datatype
9. goed; datum-datatype
10. als het een alfanumerieke constante moet zijn, dan is deze goed, als het een datum moet zijn dan is de constante fout, omdat de maanden-component te hoog is
11. goed; datum-datatype
12. goed; tijd-datatype
13. als het een alfanumerieke constante moet zijn, dan is deze goed, als het een tijd moet zijn, dan is de constante fout, want als de uren-component gelijk is aan 24 dan moeten beide andere componenten gelijk zijn aan 0
14. als het een alfanumerieke constante moet zijn, dan is deze goed, als het een timestamp moet zijn, dan is de constante fout, want de maanden-component moet tussen de 1 en 12 liggen
15. fout; een hexadecimal-datatype moet bestaan uit een even aantal tekens
16. goed; boolean datatype

5.2 De waarde van een constante staat vast, die van een expressie moet door SQL berekend worden.

5.3 Expressies kunnen gerangschikt worden naar datatype, complexiteit van de waarde en naar vorm. Rangschikken naar datatype slaat op het datatype van de waarde van de expressie, bijvoorbeeld integer, datum of alfanumeriek. Rangschikken naar complexiteit slaat op of het een 'gewone', een rij- of een tabelexpressie is. Rangschikken naar vorm houdt in of het een enkelvoudige of samengestelde expressie is.

5.4
```
SELECT   WEDSTRIJDNR, GEWONNEN - VERLOREN AS VERSCHIL
FROM     WEDSTRIJDEN
```

5.5 Ja, deze instructie is correct. Er mag gesorteerd worden op kolomnamen die in de SELECT-component geïntroduceerd worden.

5.6
```
SELECT   SPELERS.SPELERSNR, SPELERS.NAAM,
         SPELERS.VOORLETTERS
FROM     SPELERS
WHERE    SPELERS.SPELERSNR > 6
ORDER BY SPELERS.NAAM
```

5.7 Deze instructie bevat twee fouten. Ten eerste, de kolomspecificatie TEAMS.SPELERSNR is incorrect, omdat de TEAMS-tabel niet in de FROM-component voorkomt. De SQL-instructie kan dus niet naar kolommen van deze tabel refereren. Ten tweede, in de SELECT-component refereert de kolomspecificatie SPELERSNR.SPELERSNR naar een niet-bestaande SPELERSNR-tabel.

5.8
```
SELECT   PRIVILEGE, WITHGRANTOPTION
FROM     DATABASE_AUTHS
WHERE    GRANTEE = CURRENT_USER
```

5.9
```
SELECT   SPELERSNR
FROM     BESTUURSLEDEN
WHERE    BEGIN_DATUM = CURRENT_DATE
```

5.10
```
SELECT   TEAMNR,
         CASE DIVISIE
            WHEN 'ere' then 'ere divisie'
            WHEN 'tweede' THEN 'tweede divisie'
            ELSE 'onbekend'
         END AS DIVISIE
FROM     TEAMS
```

5.11
```
SELECT   BETALINGSNR, BEDRAG,
         CASE
            WHEN BEDRAG >= 0 AND BEDRAG <= 40
               THEN 'laag'
            WHEN BEDRAG >= 41 AND BEDRAG <= 80
               THEN 'middelmatig'
            WHEN BEDRAG >= 81
               THEN 'hoog'
            ELSE 'fout'
         END AS CATEGORIE
FROM     BOETES
```

5.12
```
SELECT   BETALINGSNR, BEDRAG
FROM     BOETES
WHERE    CASE
            WHEN BEDRAG >= 0 AND BEDRAG <= 40
               THEN 'laag'
            WHEN BEDRAG > 40 AND BEDRAG <= 80
               THEN 'middelmatig'
            WHEN BEDRAG > 80
               THEN 'hoog'
            ELSE 'fout'
         END = 'laag'
```

5.13
1. 100
2. 0
3. 9
4. SQL
5. Deeteebeese

5.14
```
SELECT   BETALINGSNR
FROM     BOETES
WHERE    DAYNAME(DATUM) = 'Monday'
```

5.15
```
SELECT   BETALINGSNR
FROM     BOETES
WHERE    YEAR(DATUM) = 1984
```

5.16
```
CAST('2004-03-12' AS DATE)
```

5.17 Alfanumerieke constante.

5.18 Niet elke alfanumerieke constante kan omgezet worden. Het kan alleen als de constante voldoet aan de eisen van een datum. Het omzetten van een datumconstante naar een alfanumerieke constante lukt altijd.

5.19 Nee, als de null-waarde via een gelijk-aan-operator met een andere expressie vergeleken wordt, evalueert de gehele conditie naar *onbekend* en wordt de betreffende rij niet in het eindresultaat opgenomen.

5.20 Geen enkele rij.

5.21
1. 200
2. 200
3. 3800
4. 200
5. 333.33
6. 111.11
7. 150.0000

5.22
```
SELECT  SPELERSNR, SUBSTR(VOORLETTERS,1,1) || '. ' || NAAM
FROM    SPELERS
```

5.23
```
SELECT  TEAMNR, RTRIM(DIVISIE) || ' divisie'
FROM    TEAMS
```

5.24
1. 2000-03-07
2. 2000-01-29
3. 2000-02-29
4. 2001-03-01
5. 2001-02-28

5.25
```
SELECT   SPELERSNR, BEGIN_DATUM,
         BEGIN_DATUM + INTERVAL 2 MONTH + INTERVAL 3 DAY
FROM     BESTUURSLEDEN
```

5.26
```
ADDTIME('11:34:34', '10:00:00')
```

5.27 Het resultaat is niet `11:34:34` wat men misschien zou verwachten, maar `35:34:34`.

5.28
```
'1995-12-12 11:34:34' + INTERVAL 1000 MINUTE
```

5.29
```
SELECT   BETALINGSNR, DATUM, DATUM + INTERVAL 3 HOUR +
         INTERVAL 50 SECOND + INTERVAL 99 MICROSECOND
FROM     BOETES
```

5.30
```
SELECT   BETALINGSNR
FROM     BOETES
WHERE   (BEDRAG, SPELERSNR, DATUM) = (25, 44, '1980-12-08')
```

5.31
```
SELECT   SPELERSNR
FROM     SPELERS
WHERE   (NAAM, VOORLETTERS) = (PLAATS, STRAAT)
```

1.2 Antwoorden hoofdstuk 6

6.1

1. Deze instructie is in haar totaliteit een SELECT-instructie en ook een tabelexpressie. Het is in haar totaliteit echter geen kopdeel van een select-blok, omdat een ORDER BY-component hoort bij het staartdeel van een select-blok.
2. Deze instructie is in haar totaliteit een SELECT-instructie, een tabelexpressie en het kopdeel van een select-blok.
3. Deze instructie is in haar totaliteit geen SELECT- maar een CREATE VIEW-instructie. Vanaf het woord SELECT is het wel een tabelexpressie en ook het kopdeel van een select-blok.

6.2 In de instructie is het geheel een tabelexpressie en de ORDER BY-component is het staartdeel.

6.3 Een SELECT-instructie bestaat uit minimaal één component, en wel de SELECT-component.

6.4 Ja.

6.5 Nee, als een SELECT-instructie een HAVING-component bevat, is een GROUP BY-component verplicht.

6.6

1. De FROM-component ontbreekt.
2. De GROUP BY-component moet vóór de HAVING-component gespecificeerd worden.
3. De ORDER BY-component behoort de laatste component te zijn.

6.7 De FROM-component:

```
BETALINGSNR  SPELERSNR  DATUM       BEDRAG
-----------  ---------  ----------  ------
          1          6  1980-12-08  100.00
          2         44  1981-05-05   75.00
          3         27  1983-09-10  100.00
          4        104  1984-12-08   50.00
          5         44  1980-12-08   25.00
          6          8  1980-12-08   25.00
          7         44  1982-12-30   30.00
          8         27  1984-11-12   75.00
```

De WHERE-component:

```
BETALINGSNR  SPELERSNR  DATUM       BEDRAG
-----------  ---------  ----------  ------
          2         44  1981-05-05   75.00
          3         27  1983-09-10  100.00
          4        104  1984-12-08   50.00
          7         44  1982-12-30   30.00
          8         27  1984-11-12   75.00
```

De GROUP BY-component:

```
BETALINGSNR  SPELERSNR  DATUM                     BEDRAG
-----------  ---------  ------------------------  ---------------
{2, 7}            44    {1981-05-05, 1982-12-30}  {75.00, 30.00}
{3, 8}            27    {1983-09-10, 1984-11-12}  {100.00, 75.00}
{4}              104    {1984-12-08}              {50.00}
```

De HAVING-component:

```
BETALINGSNR  SPELERSNR  DATUM                    BEDRAG
-----------  ---------  -----------------------  ---------------
{2, 7}              44  {1981-05-05, 1982-12-30}  {75.00, 30.00}
{3, 8}              27  {1983-09-10, 1984-11-12}  {100.00, 75.00}
```

De ORDER BY-component:

```
BETALINGSNR  SPELERSNR  DATUM                    BEDRAG
-----------  ---------  -----------------------  ---------------
{3, 8}              27  {1983-09-10, 1984-11-12}  {100.00, 75.00}
{2, 7}              44  {1981-05-05, 1982-12-30}  {75.00, 30.00}
```

De SELECT-component:

```
      SPELERSNR
      ---------
             27
             44
```

6.8

```
SELECT   SPELERSNR, BEGIN_DATUM
FROM     BESTUURSLEDEN
UNION
SELECT   SPELERSNR, EIND_DATUM
FROM     BESTUURSLEDEN
ORDER BY SPELERSNR
```

6.9

```
SELECT   SPELERSNR, BEGIN_DATUM, 'Begindatum'
FROM     BESTUURSLEDEN
UNION
SELECT   SPELERSNR, EIND_DATUM, 'Einddatum'
FROM     BESTUURSLEDEN
ORDER BY SPELERSNR
```

6.10

```
SELECT  SPELERSNR
FROM   (SELECT  SPELERSNR
        FROM   (SELECT  SPELERSNR, EIND_DATUM
                FROM   (SELECT  SPELERSNR, BEGIN_DATUM,
                                EIND_DATUM
                         FROM    BESTUURSLEDEN
                         WHERE   FUNCTIE = 'Secretaris')
                         AS SECRETARISSEN
                WHERE   BEGIN_DATUM >= '1990-01-01')
                AS NA1989
        WHERE   EIND_DATUM <= '1994-12-31') AS VOOR1995
```

6.11

```
SELECT   TEAMNR
FROM     TEAMS
WHERE    SPELERSNR =
        (SELECT   SPELERSNR
         FROM     SPELERS
         WHERE    NAAM = 'Permentier'
         AND      VOORLETTERS = 'R')
```

6.12

```
SELECT   NAAM
FROM     SPELERS
WHERE    SPELERSNR =
        (SELECT   SPELERSNR
         FROM     TEAMS
         WHERE    TEAMNR =
                 (SELECT   TEAMNR
                  FROM     WEDSTRIJDEN
                  WHERE    WEDSTRIJDNR = 6))
```

6.13
```
SELECT   BETALINGSNR
FROM     BOETES
WHERE    BEDRAG >
        (SELECT   BEDRAG
         FROM     BOETES
         WHERE    BETALINGSNR = 4)
```

6.14
```
SELECT   SPELERSNR
FROM     SPELERS
WHERE    DAYNAME(GEB_DATUM) =
        (SELECT   DAYNAME(GEB_DATUM)
         FROM     SPELERS
         WHERE    SPELERSNR = 2)
```

6.15
```
SELECT   SPELERSNR
FROM     BESTUURSLEDEN
WHERE   (BEGIN_DATUM, EIND_DATUM) =
        (SELECT   BEGIN_DATUM, EIND_DATUM
         FROM     BESTUURSLEDEN
         WHERE    SPELERSNR = 8
         AND      FUNCTIE = 'Penningmeester')
AND      SPELERSNR <> 8
```

6.16
```
SELECT  (SELECT   DIVISIE
         FROM     TEAMS
         WHERE    TEAMNR = 1),
        (SELECT   DIVISIE
         FROM     TEAMS
         WHERE    TEAMNR = 2)
```

6.17
```
SELECT  (SELECT   BEDRAG
         FROM     BOETES
         WHERE    BETALINGSNR = 1) +
        (SELECT   BEDRAG
         FROM     BOETES
         WHERE    BETALINGSNR = 2) +
        (SELECT   BEDRAG
         FROM     BOETES
         WHERE    BETALINGSNR = 3)
```

1.3 Antwoorden hoofdstuk 7

7.1

1. Beide tabellen hebben een kolom met de naam SPELERSNR.
2. In de SELECT-component wordt gerefereerd naar de SPELERS-tabel terwijl deze niet in de FROM-component is gespecificeerd.

7.2 De vraag was: 'Geef de naam van elke speler die aanvoerder van een team is.'

De FROM-component:

```
TEAMNR  SPELERSNR  DIVISIE  SPELERSNR  NAAM          ...
------  ---------  -------  ---------  ----------  ---
     1          6  ere              6  Permentier  ...
     1          6  ere             44  Bakker, de  ...
     1          6  ere             83  Hofland     ...
     1          6  ere              2  Elfring     ...
     1          6  ere             27  Cools       ...
     1          6  ere            104  Moerman     ...
     1          6  ere              7  Wijers      ...
     1          6  ere             57  Bohemen, van...
     1          6  ere             39  Bischoff    ...
     1          6  ere            112  Baalen, van ...
     1          6  ere              8  Niewenburg  ...
```

```
         1          6  ere           100  Permentier  ...
         1          6  ere            28  Cools       ...
         1          6  ere            95  Meuleman    ...
         2         27  tweede          6  Permentier  ...
         2         27  tweede         44  Bakker, de  ...
         2         27  tweede         83  Hofland     ...
         2         27  tweede          2  Elfring     ...
         2         27  tweede         27  Cools       ...
         2         27  tweede        104  Moerman     ...
         2         27  tweede          7  Wijers      ...
         2         27  tweede         57  Bohemen, van...
         2         27  tweede         39  Bischoff    ...
         2         27  tweede        112  Baalen, van ...
         2         27  tweede          8  Niewenburg  ...
         2         27  tweede        100  Permentier  ...
         2         27  tweede         28  Cools       ...
         2         27  tweede         95  Meuleman    ...
```

De WHERE-component:

```
TEAMNR  SPELERSNR  DIVISIE  SPELERSNR  NAAM        ...
------  ---------  -------  ---------  ----------  ---
     1          6  ere              6  Permentier  ...
     2         27  tweede          27  Cools       ...
```

De SELECT-component en tevens het eindresultaat:

```
NAAM
----------
Permentier
Cools
```

7.3
```
SELECT   BETALINGSNR, BEDRAG, SPELERS.SPELERSNR, NAAM
FROM     BOETES, SPELERS
WHERE    BOETES.SPELERSNR = SPELERS.SPELERSNR
```

7.4
```
SELECT   BETALINGSNR, NAAM
FROM     BOETES, SPELERS, TEAMS
WHERE    BOETES.SPELERSNR = TEAMS.SPELERSNR
AND      TEAMS.SPELERSNR = SPELERS.SPELERSNR
```

7.5
```
SELECT   T.TEAMNR, S.NAAM
FROM     TEAMS AS T, SPELERS AS S
WHERE    T.SPELERSNR = S.SPELERSNR
```

7.6
```
SELECT   W.WEDSTRIJDNR, S.NAAM, T.DIVISIE
FROM     WEDSTRIJDEN AS W, SPELERS AS S, TEAMS AS T
WHERE    W.SPELERSNR = S.SPELERSNR
AND      W.TEAMNR = T.TEAMNR
```

7.7
```
SELECT   S.SPELERSNR, S.NAAM
FROM     SPELERS AS S, BESTUURSLEDEN AS B
WHERE    S.SPELERSNR = B.SPELERSNR
AND      B.FUNCTIE = 'Voorzitter'
```

7.8
```
SELECT   DISTINCT BL.SPELERSNR
FROM     BESTUURSLEDEN AS BL, BOETES AS B
WHERE    BL.SPELERSNR = B.SPELERSNR
AND      BL.BEGIN_DATUM = B.DATUM
```

7.9
```
SELECT   S.SPELERSNR, S.NAAM
FROM     SPELERS AS S, SPELERS AS S27
WHERE    S.PLAATS = S27.PLAATS
AND      S27.SPELERSNR = 27
AND      S.SPELERSNR <> 27
```

7.10
```
SELECT   DISTINCT S.SPELERSNR AS SPELERS_SPELERSNR,
         S.NAAM AS SPELERS_NAAM,
         AANV.SPELERSNR AS AANVOERDER_SPELERSNR,
         AANV.NAAM AS AANVOERDER_NAAM
FROM     SPELERS AS S, SPELERS AS AANV,
         WEDSTRIJDEN AS W, TEAMS AS T
WHERE    W.SPELERSNR = S.SPELERSNR
AND      T.TEAMNR = W.TEAMNR
AND      W.SPELERSNR <> T.SPELERSNR
AND      AANV.SPELERSNR = T.SPELERSNR
```

7.11
```
SELECT   B1.BETALINGSNR, B1.SPELERSNR
FROM     BOETES AS B1, BOETES AS B2
WHERE    B1.BEDRAG = B2.BEDRAG
AND      B2.SPELERSNR = 44
AND      B1.SPELERSNR <> 44
```

7.12
```
SELECT   T.TEAMNR, S.NAAM
FROM     TEAMS AS T INNER JOIN SPELERS AS S
         ON T.SPELERSNR = S.SPELERSNR
```

7.13
```
SELECT   S.SPELERSNR, S.NAAM
FROM     SPELERS AS S INNER JOIN SPELERS AS S27
         ON S.PLAATS = S27.PLAATS
AND      S27.SPELERSNR = 27
AND      S.SPELERSNR <> 27
```

7.14
```
SELECT   W.WEDSTRIJDNR, S.NAAM, T.DIVISIE
FROM     WEDSTRIJDEN AS W INNER JOIN SPELERS AS S
         ON W.SPELERSNR = S.SPELERSNR
         INNER JOIN TEAMS AS T
         ON W.TEAMNR = T.TEAMNR
```

7.15
```
SELECT   SPELERS.SPELERSNR, BOETES.BEDRAG
FROM     SPELERS LEFT OUTER JOIN BOETES
         ON SPELERS.SPELERSNR = BOETES.SPELERSNR
```

7.16
```
SELECT   S.SPELERSNR, W.TEAMNR
FROM     SPELERS AS S LEFT OUTER JOIN WEDSTRIJDEN AS W
         ON S.SPELERSNR = W.SPELERSNR
```

7.17
```
SELECT   S.SPELERSNR, B.BEDRAG, W.TEAMNR
FROM     SPELERS AS S LEFT OUTER JOIN WEDSTRIJDEN AS W
         ON S.SPELERSNR = W.SPELERSNR
         LEFT OUTER JOIN BOETES AS B
         ON S.SPELERSNR = B.SPELERSNR
```

7.18

1. Met de left-outer-join wordt aangegeven dat alle rijen die mogelijkerwijs uit de linker tabel (de BOETES-tabel) wegvallen, alsnog in het eindresultaat opgenomen moeten worden. Maar er bestaan geen rijen in de BOETES-tabel waarvan het spelersnummer niet in de SPELERS-tabel voorkomt. De outer-join in deze FROM-component heeft dus geen nut, een inner-join zou een gelijk resultaat geven.
2. Met de left-outer-join wordt aangegeven dat alle rijen die mogelijkerwijs uit de linker tabel (de BOETES-tabel) wegvallen, alsnog in het eindresultaat opgenomen moeten worden. In dit voorbeeld zouden er wel rijen kunnen wegvallen, omdat er een groter-dan-operator in de join-conditie gebruikt wordt. Deze FROM-component heeft dus wel nut.
3. Met de right-outer-join wordt aangegeven dat alle rijen die mogelijkerwijs uit de rechter tabel (de WEDSTRIJDEN-tabel) wegvallen, alsnog in het eindresultaat opgenomen moeten worden. Maar er bestaan geen rijen in de WEDSTRIJDEN-tabel waarvan het teamnummer niet in de TEAMS-tabel voorkomt. Deze FROM-component heeft dus geen nut, een inner-join zou een vergelijkbaar resultaat geven.

4. Met de full-outer-join wordt aangegeven dat alle rijen die mogelijkerwijs uit de linker (de BOETES-tabel) en rechter tabel (de TEAMS-tabel) wegvallen, alsnog in het eindresultaat opgenomen moeten worden. Dit is in deze situatie wel degelijk mogelijk. Deze FROM-component heeft dus wel nut.

7.19

1.
```
T1.K  T2.K
----  ----
   2     3
   2     3
```
2.
```
T1.K  T2.K
----  ----
   1     ?
   2     2
   3     3
```
3.
```
T1.K  T2.K
----  ----
   2     2
   3     3
   ?     4
```
4.
```
T1.K  T2.K
----  ----
   3     2
   ?     3
   ?     4
```
5.
```
T1.K  T3.K
----  ----
   2     2
   ?     ?
```
6.
```
T1.K  T3.K
----  ----
   1     ?
   2     2
   3     ?
```
7.
```
T3.K  T4.K
----  ----
   ?     ?
   2     2
```
8.
```
T3.K  T4.K
----  ----
   ?     ?
   2     2
   ?     3
```
9.
```
T1.K  T2.K
----  ----
   1     ?
   2     2
   3     3
   ?     4
```
10.
```
T1.K  T2.K  T3.K
----  ----  ----
   1     ?     ?
   2     2     2
   3     ?     ?
   ?     3     ?
   ?     4     ?
```

7.20

1. Correct.
2. Incorrect.
3. Correct.
4. Correct.
5. Incorrect.

7.21
```
SELECT   SPELERSNR, VERSCHIL
FROM    (SELECT   SPELERSNR,
                  JAARTOE - YEAR(GEB_DATUM) AS VERSCHIL
         FROM     SPELERS) AS VERSCHILLEN
WHERE    VERSCHIL > 20
```

7.22
```
SELECT   LETTER1 || LETTER2 || LETTER3
FROM    (SELECT 'a' AS LETTER1 UNION SELECT 'b'
         UNION SELECT 'c' UNION SELECT 'd') AS LETTERS1,
        (SELECT 'a' AS LETTER2 UNION SELECT 'b'
         UNION SELECT 'c' UNION SELECT 'd') AS LETTERS2,
        (SELECT 'a' AS LETTER3 UNION SELECT 'b'
         UNION SELECT 'c' UNION SELECT 'd') AS LETTERS3
```

7.23
```
SELECT   ROUND(RAND() * 1000)
FROM    (SELECT 0 AS GETAL UNION SELECT 1 UNION SELECT 2
         UNION
         SELECT 3 UNION SELECT 4 UNION SELECT 5
         UNION
         SELECT 6 UNION SELECT 7 UNION SELECT 8
         UNION
         SELECT 9) AS GETALLEN
```

1.4 Antwoorden hoofdstuk 8

8.1
```
SELECT   BETALINGSNR
FROM     BOETES
WHERE    BEDRAG > 60
```

of

```
SELECT   BETALINGSNR
FROM     BOETES
WHERE    60 < BEDRAG
```

of

```
SELECT   BETALINGSNR
FROM     BOETES
WHERE    BEDRAG - 60 > 0
```

8.2
```
SELECT   TEAMNR
FROM     TEAMS
WHERE    SPELERSNR <> 27
```

8.3 Geen enkele rij in de SPELERS-tabel voldoet aan de conditie. Elke rij waar de BONDSNR-kolom gevuld is voldoet niet, omdat de conditie *onwaar* is. Ook elke rij waar de BONDSNR-kolom ongevuld is, dus de null-waarde bevat, wordt niet getoond.

8.4
```
SELECT   DISTINCT SPELERSNR
FROM     WEDSTRIJDEN
WHERE    GEWONNEN > VERLOREN
```

8.5
```
SELECT   DISTINCT SPELERSNR
FROM     WEDSTRIJDEN
WHERE    GEWONNEN + VERLOREN = 5
```

8.6
```
SELECT   SPELERSNR, NAAM, VOORLETTERS
FROM     SPELERS
WHERE    SPELERSNR =
        (SELECT   SPELERSNR
         FROM     BOETES
         WHERE    BETALINGSNR = 4)
```

8.7
```
SELECT   SPELERSNR, NAAM, VOORLETTERS
FROM     SPELERS
WHERE    SPELERSNR =
        (SELECT   SPELERSNR
         FROM     TEAMS
         WHERE    TEAMNR =
                 (SELECT   TEAMNR
                  FROM     WEDSTRIJDEN
                  WHERE    WEDSTRIJDNR = 2))
```

8.8
```
SELECT   SPELERSNR, NAAM
FROM     SPELERS
WHERE    GEB_DATUM =
        (SELECT   GEB_DATUM
         FROM     SPELERS
         WHERE    NAAM = 'Permentier'
         AND      VOORLETTERS = 'R')
AND      NOT (NAAM = 'Permentier' AND VOORLETTERS = 'R')
```

8.9
```
SELECT   WEDSTRIJDNR
FROM     WEDSTRIJDEN
WHERE    GEWONNEN =
        (SELECT   GEWONNEN
         FROM     WEDSTRIJDEN
         WHERE    WEDSTRIJDNR = 6)
AND      WEDSTRIJDNR <> 6
AND      TEAMNR = 2
```

8.10
```
SELECT   WEDSTRIJDNR
FROM     WEDSTRIJDEN
WHERE   (GEWONNEN, VERLOREN) =
       ((SELECT   GEWONNEN
         FROM     WEDSTRIJDEN
         WHERE    WEDSTRIJDNR = 2),
        (SELECT   VERLOREN
         FROM     WEDSTRIJDEN
         WHERE    WEDSTRIJDNR = 8))
```

8.11
```
SELECT   SPELERSNR, PLAATS, STRAAT, HUISNR
FROM     SPELERS
WHERE   (PLAATS, STRAAT, HUISNR) <
        (SELECT   PLAATS, STRAAT, HUISNR
         FROM     SPELERS
         WHERE    SPELERSNR = 100)
ORDER BY PLAATS, STRAAT, HUISNR
```

8.12
```
SELECT   BETALINGSNR
FROM     BOETES
WHERE    1965 <
        (SELECT   YEAR(GEB_DATUM)
         FROM     SPELERS
         WHERE    SPELERS.SPELERSNR = BOETES.SPELERSNR)
```

8.13
```
SELECT   BETALINGSNR, SPELERSNR
FROM     BOETES
WHERE    SPELERSNR =
        (SELECT   SPELERSNR
         FROM     TEAMS
         WHERE    TEAMS.SPELERSNR = BOETES.SPELERSNR)
```

8.14
```
SELECT   SPELERSNR, NAAM, PLAATS
FROM     SPELERS
WHERE    GESLACHT = 'V'
AND      PLAATS <> 'Den Haag'
```

of

```
SELECT   SPELERSNR, NAAM, PLAATS
FROM     SPELERS
WHERE    GESLACHT = 'V'
AND      NOT (PLAATS = 'Den Haag')
```

8.15
```
SELECT   SPELERSNR
FROM     SPELERS
WHERE    JAARTOE >= 1970
AND      JAARTOE <= 1980
```

of

```
SELECT   SPELERSNR
FROM     SPELERS
WHERE    NOT (JAARTOE < 1970 OR JAARTOE > 1980)
```

8.16
```
SELECT   SPELERSNR, NAAM, GEB_DATUM
FROM     SPELERS
WHERE    MOD(YEAR(GEB_DATUM), 400) = 0
OR      (MOD(YEAR(GEB_DATUM), 4) = 0
         AND NOT(MOD(YEAR(GEB_DATUM), 100) = 0))
```

8.17
```
SELECT   WEDSTRIJDNR, NAAM, VOORLETTERS, DIVISIE
FROM     WEDSTRIJDEN AS W, SPELERS AS S, TEAMS AS T
WHERE    W.SPELERSNR = S.SPELERSNR
AND      W.TEAMNR = T.TEAMNR
AND      YEAR(GEB_DATUM) > 1965
AND      GEWONNEN > VERLOREN
```

8.18
```
SELECT   BETALINGSNR
FROM     BOETES
WHERE    BEDRAG IN (50, 75, 100)
```

8.19
```
SELECT   SPELERSNR
FROM     SPELERS
WHERE    PLAATS NOT IN ('Den Haag', 'Voorburg')
```

of

```
SELECT   SPELERSNR
FROM     SPELERS
WHERE    NOT (PLAATS IN ('Den Haag', 'Voorburg'))
```

of

```
SELECT   SPELERSNR
FROM     SPELERS
WHERE    PLAATS <> 'Den Haag'
AND      PLAATS <> 'Voorburg'
```

8.20
```
SELECT   BETALINGSNR
FROM     BOETES
WHERE    BEDRAG IN
        (100, BETALINGSNR * 5,
        (SELECT   BEDRAG
         FROM     BOETES
         WHERE    BETALINGSNR = 2))
```

8.21
```
SELECT   SPELERSNR, PLAATS, STRAAT
FROM     SPELERS
WHERE   (PLAATS, STRAAT) IN
       (('Den Haag','Hazensteinln'),
        ('Den Haag','Erasmusweg'))
```

8.22
```
SELECT   SPELERSNR, NAAM
FROM     SPELERS
WHERE    SPELERSNR IN
        (SELECT   SPELERSNR
         FROM     BOETES)
```

8.23
```
SELECT   SPELERSNR, NAAM
FROM     SPELERS
WHERE    SPELERSNR IN
        (SELECT   SPELERSNR
         FROM     BOETES
         WHERE    BEDRAG > 50)
```

8.24
```
SELECT   TEAMNR, SPELERSNR
FROM     TEAMS
WHERE    DIVISIE = 'ere'
AND      SPELERSNR IN
        (SELECT   SPELERSNR
         FROM     SPELERS
         WHERE    PLAATS = 'Den Haag')
```

8.25
```
SELECT   SPELERSNR, NAAM
FROM     SPELERS
WHERE    SPELERSNR IN
        (SELECT   SPELERSNR
         FROM     BOETES)
AND      SPELERSNR NOT IN
        (SELECT   SPELERSNR
         FROM     TEAMS
         WHERE    DIVISIE = 'ere')
```

of

```
SELECT   SPELERSNR, NAAM
FROM     SPELERS
WHERE    SPELERSNR IN
        (SELECT   SPELERSNR
         FROM     BOETES
         WHERE    SPELERSNR NOT IN
                 (SELECT   SPELERSNR
                  FROM     TEAMS
                  WHERE    DIVISIE = 'ere'))
```

8.26 Het resultaat is leeg.

8.27
```
SELECT   WEDSTRIJDNR, SPELERSNR
FROM     WEDSTRIJDEN
WHERE   (GEWONNEN, VERLOREN) IN
        (SELECT   GEWONNEN, VERLOREN
         FROM     WEDSTRIJDEN
         WHERE    TEAMNR IN
                 (SELECT   TEAMNR
                  FROM     TEAMS
                  WHERE    DIVISIE = 'tweede'))
```

8.28
```
SELECT   SPELERSNR, NAAM
FROM     SPELERS AS S1
WHERE   (PLAATS, STRAAT, HUISNR, POSTCODE) IN
        (SELECT   PLAATS, STRAAT, HUISNR, POSTCODE
         FROM     SPELERS AS S2
         WHERE    S1.SPELERSNR <> S2.SPELERSNR)
```

8.29
```
SELECT   BETALINGSNR
FROM     BOETES
WHERE    BEDRAG BETWEEN 50 AND 100
```

8.30
```
SELECT   BETALINGSNR
FROM     BOETES
WHERE    NOT (BEDRAG BETWEEN 50 AND 100)
```

of

```
SELECT   BETALINGSNR
FROM     BOETES
WHERE    BEDRAG NOT BETWEEN 50 AND 100
```

of

```
SELECT   BETALINGSNR
FROM     BOETES
WHERE    BEDRAG < 50
OR       BEDRAG > 100
```

8.31
```
SELECT   SPELERSNR
FROM     SPELERS
WHERE    JAARTOE BETWEEN
         YEAR(GEB_DATUM + INTERVAL 16 YEAR +
         INTERVAL 1 DAY)
         AND YEAR(GEB_DATUM + INTERVAL 40 YEAR +
         INTERVAL -1 DAY)
```

8.32
```
SELECT   SPELERSNR, NAAM
FROM     SPELERS
WHERE    NAAM LIKE '%en%'
```

8.33
```
SELECT   SPELERSNR, NAAM
FROM     SPELERS
WHERE    NAAM LIKE '_____'
```

8.34
```
SELECT   SPELERSNR, NAAM
FROM     SPELERS
WHERE    NAAM LIKE '_____%'
```

of

```
SELECT   SPELERSNR, NAAM
FROM     SPELERS
WHERE    NAAM LIKE '%_____'
```

of

```
SELECT   SPELERSNR, NAAM
FROM     SPELERS
WHERE    NAAM LIKE '%_____%'
```

of

```
SELECT   SPELERSNR, NAAM
FROM     SPELERS
WHERE    LENGTH(RTRIM(NAAM)) > 6
```

8.35

```
SELECT   SPELERSNR, NAAM
FROM     SPELERS
WHERE    NAAM LIKE '_e%e_'
```

8.36

```
SELECT   KOL1
FROM    (SELECT 'A%B%C' AS KOL1
         UNION
         SELECT '%ABC%'
         UNION
         SELECT 'ABC%D') AS T1
WHERE    KOL1 LIKE '_@%%@%_' ESCAPE '@'
```

8.37

```
SELECT   SPELERSNR, NAAM
FROM     SPELERS
WHERE    NAAM REGEXP 'en'
```

8.38

```
SELECT   SPELERSNR, NAAM
FROM     SPELERS
WHERE    NAAM REGEXP '^n.*g$'
```

8.39

```
SELECT   SPELERSNR, NAAM
FROM     SPELERS
WHERE    NAAM REGEXP '[a-z]{10}'
```

8.40

```
SELECT   BOEKNR, SAMENVATTING
FROM     BOEKEN
WHERE    MATCH(SAMENVATTING)
         AGAINST ('students' IN NATURAL LANGUAGE MODE)
```

8.41

```
SELECT   BOEKNR, SAMENVATTING
FROM     BOEKEN
WHERE    MATCH(SAMENVATTING)
         AGAINST ('database' IN BOOLEAN MODE)
```

8.42

```
SELECT   BOEKNR, SAMENVATTING
FROM     BOEKEN
WHERE    MATCH(SAMENVATTING)
         AGAINST ('database languages' IN NATURAL LANGUAGE MODE)
```

8.43

```
SELECT   BOEKNR, SAMENVATTING
FROM     BOEKEN
WHERE    MATCH(SAMENVATTING)
         AGAINST ('+database -languages' IN BOOLEAN MODE)
```

8.44

```
SELECT   SPELERSNR
FROM     SPELERS
WHERE    BONDSNR IS NULL
```

8.45 De NAAM-kolom is gedefinieerd als NOT NULL. De kolom zal dus nooit een null-waarde bevatten. De conditie zal daarom voor elke rij *onwaar* zijn.

8.46

```
SELECT   NAAM, VOORLETTERS
FROM     SPELERS
WHERE    EXISTS
        (SELECT   *
         FROM     TEAMS
         WHERE    SPELERSNR = SPELERS.SPELERSNR)
```

8.47
```
SELECT   NAAM, VOORLETTERS
FROM     SPELERS AS S
WHERE    NOT EXISTS
        (SELECT   *
         FROM     TEAMS AS T
         WHERE    T.SPELERSNR = S.SPELERSNR
         AND      EXISTS
                 (SELECT   *
                  FROM     WEDSTRIJDEN AS W
                  WHERE    W.TEAMNR = T.TEAMNR
                  AND      W.SPELERSNR = 112))
```

8.48
```
SELECT   SPELERSNR
FROM     SPELERS
WHERE    GEB_DATUM <= ALL
        (SELECT   GEB_DATUM
         FROM     SPELERS
         WHERE    PLAATS = 'Den Haag')
AND      PLAATS = 'Den Haag'
```

8.49
```
SELECT   SPELERSNR, NAAM
FROM     SPELERS
WHERE    SPELERSNR = ANY
        (SELECT   SPELERSNR
         FROM     BOETES)
```

8.50
```
SELECT   BETALINGSNR, BEDRAG, DATUM
FROM     BOETES AS B1
WHERE    BEDRAG >= ALL
        (SELECT   BEDRAG
         FROM     BOETES AS B2
         WHERE    YEAR(B1.DATUM) = YEAR(B2.DATUM))
```

8.51
```
SELECT  (SELECT   SPELERSNR
         FROM     SPELERS
         WHERE    SPELERSNR <= ALL
                 (SELECT   SPELERSNR
                  FROM     SPELERS)),
        (SELECT   SPELERSNR
         FROM     SPELERS
         WHERE    SPELERSNR >= ALL
                 (SELECT   SPELERSNR
                  FROM     SPELERS))
```

8.52
1. A.K_1: S_1, S_2, S_3, S_4, S_5
2. B.K_1: S_2, S_3, S_4
3. C.K_1: S_3
4. D.K_1: S_4
5. E.K_1: S_5

1.53
```
SELECT   NAAM, VOORLETTERS
FROM     SPELERS
WHERE    SPELERSNR IN
        (SELECT   SPELERSNR
         FROM     WEDSTRIJDEN
         WHERE    TEAMNR IN
                 (SELECT   TEAMNR
                  FROM     TEAMS
                  WHERE    DIVISIE = 'ere'))
AND      SPELERSNR IN
        (SELECT   SPELERSNR
         FROM     WEDSTRIJDEN
         WHERE    GEWONNEN > VERLOREN)
AND      SPELERSNR NOT IN
        (SELECT   SPELERSNR
         FROM     BOETES)
```

8.54
```
SELECT   SPELERSNR, NAAM
FROM     SPELERS
WHERE    SPELERSNR IN
        (SELECT   SPELERSNR
         FROM     WEDSTRIJDEN
         WHERE    TEAMNR = 1)
AND      SPELERSNR IN
        (SELECT   SPELERSNR
         FROM     WEDSTRIJDEN
         WHERE    TEAMNR = 2)
```

8.55
```
SELECT   SPELERSNR, NAAM
FROM     SPELERS
WHERE    EXISTS
        (SELECT   *
         FROM     BOETES
         WHERE    SPELERSNR = SPELERS.SPELERSNR)
```

8.56
```
SELECT   SPELERSNR, NAAM
FROM     SPELERS
WHERE    SPELERSNR IN
        (SELECT  SPELERSNR
         FROM    WEDSTRIJDEN AS W1
         WHERE   GEWONNEN > VERLOREN
         AND     EXISTS
                (SELECT  *
                 FROM    WEDSTRIJDEN AS W2
                 WHERE   W1.SPELERSNR = W2.SPELERSNR
                 AND     GEWONNEN > VERLOREN
                 AND     W1.WEDSTRIJDNR <> W2.WEDSTRIJDNR))
```

of

```
SELECT   SPELERSNR, NAAM
FROM     SPELERS
WHERE    1 < (SELECT   COUNT(*)
              FROM     WEDSTRIJDEN
              WHERE    GEWONNEN > VERLOREN
              AND      SPELERS.SPELERSNR = SPELERSNR)
```

8.57
```
SELECT   NAAM, VOORLETTERS
FROM     SPELERS
WHERE    NOT EXISTS
        (SELECT   *
         FROM     BOETES
         WHERE    SPELERS.SPELERSNR = SPELERSNR
         AND      DATUM BETWEEN '1980-01-01'
                  AND '1980-12-31')
```

8.58
```
SELECT   DISTINCT SPELERSNR
FROM     BOETES AS B1
WHERE    EXISTS
        (SELECT   *
         FROM     BOETES AS B2
         WHERE    B1.BEDRAG = B2.BEDRAG
         AND      B1.BETALINGSNR <> B2.BETALINGSNR)
```

8.59
```
SELECT   SPELERSNR
FROM     SPELERS
WHERE    SPELERSNR NOT IN
        (SELECT   SPELERSNR
         FROM     WEDSTRIJDEN WHERE GEWONNEN = 3)
```

8.60
```
SELECT   TEAMNR, DIVISIE
FROM     TEAMS
WHERE    TEAMNR NOT IN
        (SELECT   TEAMNR
         FROM     WEDSTRIJDEN
         WHERE    SPELERSNR = 6)
```

8.61
```
SELECT   DISTINCT SPELERSNR
FROM     WEDSTRIJDEN
WHERE    SPELERSNR NOT IN
        (SELECT   SPELERSNR
         FROM     WEDSTRIJDEN
         WHERE    TEAMNR IN
                 (SELECT   TEAMNR
                  FROM     WEDSTRIJDEN
                  WHERE    SPELERSNR = 57))
```

1.5 Antwoorden hoofdstuk 9

9.1
1. Niet overbodig.
2. Niet overbodig.
3. Wel overbodig omdat er een conditie op de primaire sleutel staat.
4. Niet overbodig.
5. Niet overbodig.
6. Niet overbodig.

9.2
1.
```
K2
--
k2
?
```
2.
```
K2  K3
--  --
k2  k3
k2  ?
?   ?
```
3.
```
K2
--
k2
?
```

9.3 Deze instructie is niet correct. Er wordt een aggregatiefunctie in de SELECT-component gebruikt, dus moeten alle andere kolomnamen ook binnen een aggregatie-functie voorkomen.

9.4
```
SELECT   COUNT(*), MAX(BEDRAG)
FROM     BOETES
```

9.5
```
SELECT   COUNT(DISTINCT FUNCTIE)
FROM     BESTUURSLEDEN
```

9.6
```
SELECT   COUNT(BONDSNR)
FROM     SPELERS
WHERE    PLAATS = 'Rijswijk'
```

9.7
```
SELECT   TEAMNR, DIVISIE,
        (SELECT   COUNT(*)
         FROM     WEDSTRIJDEN
         WHERE    TEAMS.TEAMNR = WEDSTRIJDEN.TEAMNR)
FROM     TEAMS
```

9.8

```
SELECT   SPELERSNR, NAAM,
        (SELECT   COUNT(*)
         FROM     WEDSTRIJDEN
         WHERE    WEDSTRIJDEN.SPELERSNR = SPELERS.SPELERSNR
         AND      GEWONNEN > VERLOREN)
FROM     SPELERS
```

9.9

```
SELECT 'Aantal spelers' AS TABELLEN,
        (SELECT COUNT(*) FROM SPELERS) AS AANTAL UNION
SELECT 'Aantal teams',
        (SELECT COUNT(*) FROM TEAMS) UNION
SELECT 'Aantal wedstrijden',
        (SELECT COUNT(*) FROM WEDSTRIJDEN)
```

9.10

```
SELECT   MIN(GEWONNEN)
FROM     WEDSTRIJDEN
WHERE    GEWONNEN > VERLOREN
```

9.11

```
SELECT   SPELERSNR,
        (SELECT   MAX(BEDRAG)
         FROM     BOETES
         WHERE    BOETES.SPELERSNR =
                  SPELERS.SPELERSNR) -
        (SELECT   MIN(BEDRAG)
         FROM     BOETES
         WHERE    BOETES.SPELERSNR =
                  SPELERS.SPELERSNR)
FROM     SPELERS
```

9.12

```
SELECT   SPELERSNR, GEB_DATUM
FROM     SPELERS
WHERE    YEAR(GEB_DATUM) =
        (SELECT   MAX(YEAR(GEB_DATUM))
         FROM     SPELERS
         WHERE    SPELERSNR IN
                 (SELECT   SPELERSNR
                  FROM     WEDSTRIJDEN
                  WHERE    TEAMNR = 1))
```

9.13

1. 9
2. 8
3. 1
4. 5
5. 24
6. 3
7. 5
8. 1
9. 5
10. 15
11. $15 / 5 = 3$

9.14

```
SELECT   AVG(BEDRAG)
FROM     BOETES
WHERE    SPELERSNR IN
        (SELECT   SPELERSNR
         FROM     WEDSTRIJDEN
         WHERE    TEAMNR = 1)
```

9.15

```
SELECT   SPELERSNR, NAAM
FROM     SPELERS
WHERE   (SELECT   SUM(BEDRAG)
         FROM     BOETES
         WHERE    BOETES.SPELERSNR = SPELERS.SPELERSNR)
         > 100
```

9.16
```
SELECT   NAAM, VOORLETTERS
FROM     SPELERS
WHERE    SPELERSNR IN
        (SELECT   SPELERSNR
         FROM     WEDSTRIJDEN
         WHERE    GEWONNEN >
                 (SELECT   SUM(GEWONNEN)
                  FROM     WEDSTRIJDEN
                  WHERE    SPELERSNR = 27))
```

9.17
```
SELECT   SPELERSNR, NAAM
FROM     SPELERS
WHERE   (SELECT   SUM(GEWONNEN)
         FROM     WEDSTRIJDEN
         WHERE    WEDSTRIJDEN.SPELERSNR = SPELERS.SPELERSNR) = 8
```

9.18
```
SELECT   SPELERSNR, NAAM
FROM     SPELERS
WHERE    LENGTH(RTRIM(NAAM)) >
        (SELECT   AVG(LENGTH(RTRIM(NAAM)))
         FROM     SPELERS)
```

9.19
```
SELECT   SPELERSNR,
        (SELECT   MAX(BEDRAG)
         FROM     BOETES
         WHERE    BOETES.SPELERSNR =
                  SPELERS.SPELERSNR) -
        (SELECT   AVG(BEDRAG)
         FROM     BOETES
         WHERE    BOETES.SPELERSNR =
                  SPELERS.SPELERSNR)
FROM     SPELERS
```

9.20
```
SELECT   SPELERSNR,
         REPEAT('*',
            CAST((SELECT   AVG(BEDRAG)
                  FROM     BOETES
                  WHERE    BOETES.SPELERSNR =
                           SPELERS.SPELERSNR)/10
                  AS SIGNED INTEGER))
FROM     SPELERS
```

9.21
```
SELECT   SQRT(SUM(P) /
        (SELECT COUNT(*) FROM BOETES WHERE SPELERSNR = 44))
FROM    (SELECT   POWER(BEDRAG -
                 (SELECT   AVG(BEDRAG)
                  FROM     BOETES
                  WHERE    SPELERSNR = 44),2) AS P
         FROM     BOETES
         WHERE    SPELERSNR = 44) AS POWERS
```

9.22
```
SELECT   WEDSTRIJDNR, SPELERSNR, TEAMNR, ROW_NUMBER() OVER()
FROM     WEDSTRIJDEN
ORDER BY WEDSTRIJDNR
```

9.23 Zie paragraaf 9.11.1 in Het SQL Leerboek.

9.24
```
SELECT   WEDSTRIJDNR, SPELERSNR, TEAMNR,
         ROW_NUMBER() OVER(ORDER BY TEAMNR, SPELERSNR) AS VOLGNR
FROM     WEDSTRIJDEN
ORDER BY WEDSTRIJDNR
```

9.25
```
SELECT   SPELERSNR, (SELECT   NAAM
                     FROM     SPELERS AS S
                     WHERE    S.SPELERSNR = T.SPELERSNR)
FROM    (SELECT   SPELERSNR,
                  ROW_NUMBER() OVER(ORDER BY BEDRAG DESC) AS VOLGNR
         FROM     BOETES) AS T
WHERE    VOLGNR <= 3
```

9.26
```
SELECT   MAX(VOLGNR)
FROM    (SELECT   DENSE_RANK() OVER(ORDER BY PLAATS) AS VOLGNR
         FROM     SPELERS) AS T
```

9.27
```
SELECT   TEAMNR, SUM(GEWONNEN),
         ROW_NUMBER() OVER (ORDER BY SUM(GEWONNEN))
FROM     WEDSTRIJDEN
GROUP BY TEAMNR
```

9.28
```
SELECT   WEDSTRIJDNR, TEAMNR, SPELERSNR,
         ROW_NUMBER() OVER (PARTITION BY TEAMNR ORDER BY SPELERSNR)
FROM     WEDSTRIJDEN
```

9.29
```
SELECT   BETALINGSNR, BEDRAG, DATUM,
         ROW_NUMBER() OVER (PARTITION BY YEAR(DATUM) ORDER BY BEDRAG)
FROM     BOETES
ORDER BY BETALINGSNR
```

9.30
```
SELECT   WEDSTRIJDNR, TEAMNR, GEWONNEN,
         SUM(GEWONNEN) OVER (PARTITION BY TEAMNR)
FROM     WEDSTRIJDEN
```

9.31
```
SELECT   WEDSTRIJDNR, TEAMNR, GEWONNEN,
         SUM(GEWONNEN) OVER (ORDER BY WEDSTRIJDNR)
FROM     WEDSTRIJDEN
ORDER BY WEDSTRIJDNR
```

9.32
```
SELECT   WEDSTRIJDNR, TEAMNR, GEWONNEN,
         SUM(GEWONNEN) OVER (PARTITION BY TEAMNR ORDER BY WEDSTRIJDNR)
FROM     WEDSTRIJDEN
ORDER BY WEDSTRIJDNR
```

1.6 Antwoorden hoofdstuk 10

10.1
```
SELECT   JAARTOE
FROM     SPELERS
GROUP BY JAARTOE
```

10.2
```
SELECT   JAARTOE, COUNT(*)
FROM     SPELERS
GROUP BY JAARTOE
```

10.3
```
SELECT   SPELERSNR, AVG(BEDRAG), COUNT(*)
FROM     BOETES
GROUP BY SPELERSNR
```

10.4
```
SELECT   TEAMNR, COUNT(*), SUM(GEWONNEN)
FROM     WEDSTRIJDEN
WHERE    TEAMNR IN
        (SELECT   TEAMNR
         FROM     TEAMS
         WHERE    DIVISIE = 'ere')
GROUP BY TEAMNR
```

10.5

```
SELECT   GEWONNEN, VERLOREN, COUNT(*)
FROM     WEDSTRIJDEN
WHERE    GEWONNEN > VERLOREN
GROUP BY GEWONNEN, VERLOREN
ORDER BY GEWONNEN, VERLOREN
```

10.6

```
SELECT   S.PLAATS, T.DIVISIE, SUM(GEWONNEN)
FROM    (WEDSTRIJDEN AS W INNER JOIN SPELERS AS S
         ON W.SPELERSNR = S.SPELERSNR)
         INNER JOIN TEAMS AS T
         ON W.TEAMNR = T.TEAMNR
GROUP BY S.PLAATS, T.DIVISIE
ORDER BY S.PLAATS
```

10.7

```
SELECT   NAAM, VOORLETTERS, COUNT(*)
FROM     SPELERS AS S INNER JOIN BOETES AS BT
         ON S.SPELERSNR = BT.SPELERSNR
WHERE    S.PLAATS = 'Rijswijk'
GROUP BY S.SPELERSNR, NAAM, VOORLETTERS
```

10.8

```
SELECT   T.TEAMNR, DIVISIE, SUM(GEWONNEN)
FROM     TEAMS AS T, WEDSTRIJDEN AS W
WHERE    T.TEAMNR = W.TEAMNR
GROUP BY T.TEAMNR, DIVISIE
```

10.9

```
SELECT   LENGTH(RTRIM(NAAM)), COUNT(*)
FROM     SPELERS
GROUP BY LENGTH(RTRIM(NAAM))
```

10.10

```
SELECT   ABS(GEWONNEN - VERLOREN), COUNT(*)
FROM     WEDSTRIJDEN
GROUP BY ABS(GEWONNEN - VERLOREN)
```

10.11

```
SELECT   YEAR(BEGIN_DATUM), MONTH(BEGIN_DATUM), COUNT(*)
FROM     BESTUURSLEDEN
GROUP BY YEAR(BEGIN_DATUM), MONTH(BEGIN_DATUM)
ORDER BY YEAR(BEGIN_DATUM), MONTH(BEGIN_DATUM)
```

10.12

1. Op de DIVISIE-kolom is niet gegroepeerd, terwijl deze kolom wel in de SELECT-component voorkomt.
2. De NAAM-kolom mag niet op deze wijze voorkomen in de SELECT-component, omdat er niet gegroepeerd is op de volle NAAM-kolom.
3. De SPELERSNR-kolom komt voor in de SELECT-component terwijl er niet op gegroepeerd is en tevens komt de kolom niet voor als parameter van een aggregatiefunctie.

10.13

1. Wel overbodig.
2. Niet overbodig.
3. Wel overbodig.

10.14

```
SELECT   AVG(AANTALLEN)
FROM    (SELECT   COUNT(*) AS AANTALLEN
         FROM     SPELERS
         GROUP BY PLAATS) AS PLAATSEN
```

10.15

```
SELECT   TEAMS.TEAMNR, DIVISIE, AANTAL_SPELERS
FROM     TEAMS LEFT OUTER JOIN
        (SELECT   TEAMNR, COUNT(*) AS AANTAL_SPELERS
         FROM     WEDSTRIJDEN
         GROUP BY TEAMNR) AS W
         ON (TEAMS.TEAMNR = W.TEAMNR)
```

10.16
```
SELECT   SPELERS.SPELERSNR, NAAM, TOTAALBEDRAG,
         AANTAL_TEAMS
FROM    (SPELERS LEFT OUTER JOIN
        (SELECT   SPELERSNR, SUM(BEDRAG) AS TOTAALBEDRAG
         FROM     BOETES
         GROUP BY SPELERSNR) AS TOTALEN
         ON (SPELERS.SPELERSNR = TOTALEN.SPELERSNR))
            LEFT OUTER JOIN
           (SELECT   SPELERSNR, COUNT(*) AS AANTAL_TEAMS
            FROM     TEAMS
            WHERE    DIVISIE = 'ere'
            GROUP BY SPELERSNR) AS AANTALLEN
            ON (SPELERS.SPELERSNR = AANTALLEN.SPELERSNR)
```

10.17
```
SELECT   TEAMNR, COUNT(DISTINCT SPELERSNR)
FROM     WEDSTRIJDEN
WHERE    TEAMNR IN
        (SELECT   TEAMNR
         FROM     SPELERS AS S INNER JOIN TEAMS AS T
                  ON S.SPELERSNR = T.SPELERSNR
         AND      PLAATS = 'Den Haag')
AND      GEWONNEN > VERLOREN
GROUP BY TEAMNR
```

10.18
```
SELECT   SPELERSNR, NAAM, JAARTOE - GEMIDDELDE
FROM     SPELERS,
        (SELECT   AVG(JAARTOE) AS GEMIDDELDE
         FROM     SPELERS) AS T
```

10.19
```
SELECT   SPELERSNR, NAAM, JAARTOE - GEMIDDELDE
FROM     SPELERS,
        (SELECT   PLAATS, AVG(JAARTOE) AS GEMIDDELDE
         FROM     SPELERS
         GROUP BY PLAATS) AS PLAATSEN
WHERE    SPELERS.PLAATS = PLAATSEN.PLAATS
```

10.20
```
SELECT   TEAMNR, COUNT(*)
FROM     WEDSTRIJDEN
GROUP BY TEAMNR WITH ROLLUP
```

10.21
```
SELECT   S.NAAM, T.DIVISIE, SUM(GEWONNEN)
FROM    (WEDSTRIJDEN AS W INNER JOIN SPELERS AS S
         ON W.SPELERSNR = S.SPELERSNR)
         INNER JOIN TEAMS AS T
         ON W.TEAMNR = T.TEAMNR
GROUP BY S.NAAM, T.DIVISIE WITH ROLLUP
```

10.22 Met de WITH ROLLUP-specificatie worden alle aggregatieniveaus berekend met onderaan een groepering op basis van de expressies die gespecificeerd staan. De WITH CUBE-specificatie geeft veel meer gegevens. Voor alle mogelijke combinaties van expressies die gespecificeerd staan, worden groeperingen uitgevoerd.

10.23
```
SELECT   ROW_NUMBER() OVER () AS VOLGNR,
         TEAMNR, SPELERSNR, GEWONNEN, COUNT(*)
FROM     WEDSTRIJDEN
GROUP BY TEAMNR, SPELERSNR, GEWONNEN WITH CUBE
ORDER BY TEAMNR, SPELERSNR
```

10.24
```
SELECT   COUNT(*)
FROM     WEDSTRIJDEN
GROUP BY GROUPING SETS (())
```

10.25
```
SELECT   TEAMNR, SPELERSNR, COUNT(*)
FROM     WEDSTRIJDEN
GROUP BY GROUPING SETS ((TEAMNR, SPELERSNR), (TEAMNR), ())
ORDER BY TEAMNR, SPELERSNR
```

10.26

1. [] ∪ [E_1] ∪ [E_2]
2. [E_1] ∪ [E_2, E_3] ∪ [E_3, E_4, E_5]
3. [E_1, E_2] ∪ [] ∪ [E_3]

10.27

```
SELECT   TEAMNR, SPELERSNR, COUNT(*)
FROM     WEDSTRIJDEN
WHERE    GEWONNEN > VERLOREN
GROUP BY ROLLUP (TEAMNR, SPELERSNR)
ORDER BY TEAMNR, SPELERSNR
```

10.28

```
SELECT   S.PLAATS, S.GESLACHT, W.TEAMNR, COUNT(*)
FROM     WEDSTRIJDEN AS W INNER JOIN SPELERS AS S
         ON W.SPELERSNR = S.SPELERSNR
GROUP BY CUBE (S.PLAATS, S.GESLACHT, W.TEAMNR)
ORDER BY S.PLAATS, S.GESLACHT, W.TEAMNR
```

1.7 Antwoorden hoofdstuk 11

11.1

```
SELECT   PLAATS
FROM     SPELERS
GROUP BY PLAATS
HAVING   COUNT(*) > 4
```

11.2

```
SELECT   SPELERSNR
FROM     BOETES
GROUP BY SPELERSNR
HAVING   SUM(BEDRAG) > 150
```

11.3

```
SELECT   NAAM, VOORLETTERS, COUNT(*)
FROM     SPELERS INNER JOIN BOETES
         ON SPELERS.SPELERSNR = BOETES.SPELERSNR
GROUP BY SPELERS.SPELERSNR, NAAM, VOORLETTERS
HAVING   COUNT(*) > 1
```

11.4

```
SELECT   TEAMNR, COUNT(*)
FROM     WEDSTRIJDEN
GROUP BY TEAMNR
HAVING   COUNT(*) >= ALL
        (SELECT   COUNT(*)
         FROM     WEDSTRIJDEN
         GROUP BY TEAMNR)
```

11.5

```
SELECT   TEAMNR, DIVISIE
FROM     TEAMS
WHERE    TEAMNR IN
        (SELECT   TEAMNR
         FROM     WEDSTRIJDEN
         GROUP BY TEAMNR
         HAVING   COUNT(DISTINCT SPELERSNR) > 4)
```

11.6

```
SELECT   NAAM, VOORLETTERS
FROM     SPELERS
WHERE    SPELERSNR IN
        (SELECT   SPELERSNR
         FROM     BOETES
         WHERE    BEDRAG > 40
         GROUP BY SPELERSNR
         HAVING   COUNT(*) >= 2)
```

11.7
```
SELECT   NAAM, VOORLETTERS
FROM     SPELERS
WHERE    SPELERSNR IN
        (SELECT   SPELERSNR
         FROM     BOETES
         GROUP BY SPELERSNR
         HAVING   SUM(BEDRAG) >= ALL
                 (SELECT   SUM(BEDRAG)
                  FROM     BOETES
                  GROUP BY SPELERSNR))
```

11.8
```
SELECT   SPELERSNR
FROM     BOETES
WHERE    SPELERSNR <> 104
GROUP BY SPELERSNR
HAVING   SUM(BEDRAG)  =
        (SELECT   SUM(BEDRAG) * 2
         FROM     BOETES
         WHERE    SPELERSNR = 104)
```

11.9
```
SELECT   SPELERSNR
FROM     BOETES
WHERE    SPELERSNR <> 6
GROUP BY SPELERSNR
HAVING   COUNT(*) =
        (SELECT   COUNT(*)
         FROM     BOETES
         WHERE    SPELERSNR = 6)
```

11.10
```
SELECT   S.SPELERSNR, S.NAAM
FROM     SPELERS AS S, WEDSTRIJDEN AS W1
WHERE    S.SPELERSNR = W1.SPELERSNR
GROUP BY S.SPELERSNR, S.NAAM
HAVING   SUM(GEWONNEN) >
        (SELECT   SUM(VERLOREN)
         FROM     WEDSTRIJDEN AS W2
         WHERE    W2.SPELERSNR = S.SPELERSNR
         GROUP BY W2.SPELERSNR)
```

1.8 Antwoorden hoofdstuk 12

12.1
1. `ORDER BY 1`
2. `ORDER BY SPELERSNR`
3. `ORDER BY 1 ASC`
4. `ORDER BY SPELERSNR ASC`

12.2
1. Goed.
2. Fout, want er is geen twintigste kolom in de SPELERS-tabel.
3. Goed, ook al wordt er tweemaal op de VOORLETTERS-kolom gesorteerd. De laatste sortering zal echter genegeerd worden.
4. Goed, ook al wordt er tweemaal op de SPELERSNR-kolom gesorteerd. De laatste sortering zal echter genegeerd worden.

12.3
```
SELECT   SPELERSNR, TEAMNR, GEWONNEN - VERLOREN
FROM     WEDSTRIJDEN
ORDER BY 3 ASC
```

1.9 Antwoorden hoofdstuk 13

13.1
```
SELECT   BETALINGSNR, BEDRAG, DATUM
FROM     BOETES
ORDER BY BEDRAG DESC, DATUM DESC
LIMIT    4
```

13.2
```
 (SELECT   WEDSTRIJDNR
 FROM     WEDSTRIJDEN
 ORDER BY WEDSTRIJDNR ASC
 LIMIT    2)
UNION
(SELECT   WEDSTRIJDNR
 FROM     WEDSTRIJDEN
 ORDER BY WEDSTRIJDNR DESC
 LIMIT    2)
```

13.3
```
SELECT   SPELERSNR, NAAM
FROM    (SELECT   SPELERSNR, NAAM
         FROM     SPELERS
         ORDER BY SPELERSNR ASC
         LIMIT    10) AS S10
ORDER BY NAAM DESC
LIMIT    5
```

13.4
```
SELECT   SPELERSNR, NAAM
FROM     SPELERS
WHERE    SPELERSNR IN
        (SELECT   SPELERSNR
         FROM    (SELECT   SPELERSNR, COUNT(*) AS AANTAL
                  FROM     WEDSTRIJDEN
                  WHERE    GEWONNEN > VERLOREN
                  GROUP BY SPELERSNR) AS WINNAARS
         ORDER BY AANTAL DESC, SPELERSNR ASC
         LIMIT    2)
```

13.5
```
SELECT   SPELERSNR, NAAM
FROM     SPELERS
WHERE    SPELERSNR IN
        (SELECT   BOETES.SPELERSNR
         FROM     BOETES INNER JOIN SPELERS
                  ON BOETES.SPELERSNR = SPELERS.SPELERSNR
         ORDER BY BEDRAG DESC, NAAM ASC
         LIMIT    4)
```

13.6
```
SELECT   BETALINGSNR, BEDRAG
FROM     BOETES
ORDER BY BEDRAG DESC
LIMIT    1 OFFSET 2
```

1.10 Antwoorden hoofdstuk 14

14.1
```
SELECT   SPELERSNR
FROM     BESTUURSLEDEN
UNION
SELECT   SPELERSNR
FROM     BOETES
GROUP BY SPELERSNR
HAVING   COUNT(*) >= 2
```

14.2

```
SELECT   MAX(DATUM)
FROM    (SELECT   MAX(GEB_DATUM) AS DATUM
         FROM     SPELERS
         UNION
         SELECT   MAX(DATUM)
         FROM     BOETES) AS TWEEDATUMS
```

14.3

1. Goed.
2. Goed, ook al zijn de lengtes van de kolommen NAAM en POSTCODE niet gelijk.
3. Goed.
4. Goed, al is DISTINCT in een SELECT-component bij een UNION-operator overbodig.
5. Fout, want bij een UNION-operator mag alleen de laatste SELECT-instructie een ORDER BY-component bevatten.

14.4

1. 6
2. 14
3. 12

14.5

```
SELECT   SPELERSNR
FROM     BESTUURSLEDEN
INTERSECT
SELECT   SPELERSNR
FROM     BOETES
GROUP BY SPELERSNR
HAVING   COUNT(*) >= 2
```

14.6

```
SELECT   COUNT(*)
FROM    (SELECT   SPELERSNR
         FROM     BESTUURSLEDEN
         INTERSECT
         SELECT   SPELERSNR
         FROM     BOETES
         GROUP BY SPELERSNR
         HAVING   COUNT(*) >= 2) AS SPELERS
```

14.7

```
SELECT   SPELERSNR
FROM     BESTUURSLEDEN
EXCEPT
SELECT   SPELERSNR
FROM     BOETES
```

14.8

```
SELECT   1 UNION SELECT 2 UNION SELECT 3 UNION SELECT 4
         UNION SELECT 5 UNION SELECT 6 UNION SELECT 7
         UNION SELECT 8 UNION SELECT 9 UNION SELECT 10
         UNION SELECT 11 UNION SELECT 12 UNION SELECT 13
         UNION SELECT 14 UNION SELECT 15 UNION SELECT 16
         UNION SELECT 17 UNION SELECT 18 UNION SELECT 19
         UNION SELECT 20
EXCEPT
SELECT   BETALINGSNR
FROM     BOETES
```

14.9

```
SELECT   SUM(AANTAL)
FROM    (SELECT   COUNT(*) AS AANTAL
         FROM     SPELERS
         UNION ALL
         SELECT   COUNT(*) AS AANTAL
         FROM     TEAMS) AS AANTALLEN
```

14.10

```
SELECT   POWER(CIJFER,2)
FROM    (SELECT 0 AS CIJFER UNION SELECT 1 UNION
         SELECT 2 UNION SELECT 3 UNION
         SELECT 4 UNION SELECT 5 UNION
         SELECT 6 UNION SELECT 7 UNION
         SELECT 8 UNION SELECT 9) AS CIJFERS1
UNION ALL
SELECT   POWER(CIJFER,3)
FROM    (SELECT 0 AS CIJFER UNION SELECT 1 UNION
         SELECT 2 UNION SELECT 3 UNION
         SELECT 4 UNION SELECT 5 UNION
         SELECT 6 UNION SELECT 7 UNION
         SELECT 8 UNION SELECT 9) AS CIJFERS2
ORDER BY 1
```

1.11 Antwoorden hoofdstuk 15

15.1

```
WITH VERSCHILLEN (WEDSTRIJDNR, VERSCHIL) AS
        (SELECT   WEDSTRIJDNR,
         ABS(GEWONNEN - VERLOREN)
         FROM     WEDSTRIJDEN)
SELECT   WEDSTRIJDNR, VERSCHIL
FROM     VERSCHILLEN
WHERE    VERSCHIL > 2
```

15.2

```
WITH VOORNAMEN (NAAM) AS
        (SELECT 'John' AS VOORNAAM
         UNION
         SELECT 'Mark'
         UNION
         SELECT 'Arnold'),
     ACHTERNAMEN (NAAM) AS
        (SELECT 'Berg' AS ACHTERNAAM
         UNION
         SELECT 'Johnson'
         UNION
         SELECT 'Willems')
SELECT   VOORNAMEN.NAAM, ACHTERNAMEN.NAAM
FROM     VOORNAMEN, ACHTERNAMEN
```

15.3

```
WITH     CIJFERS (CIJFER) AS
        (VALUES (0),(1),(2),(3),(4),(5),(6),(7),(8),(9))
SELECT   POWER(GETAL,2) AS POWERS
FROM    (SELECT    CIJFER1.CIJFER * 10 + CIJFER2.CIJFER
         FROM      CIJFERS AS CIJFER1,
                   CIJFERS AS CIJFER2) AS GETALLEN (GETAL)
WHERE    POWER(GETAL,2) < 5000
INTERSECT
SELECT   POWER(GETAL,3)
FROM    (SELECT    CIJFER1.CIJFER * 10 + CIJFER2.CIJFER
         FROM      CIJFERS AS CIJFER1,
                   CIJFERS AS CIJFER2) AS GETALLEN (GETAL)
WHERE    POWER(GETAL,3) < 5000
```

of

```
WITH     CIJFERS (CIJFER) AS
        (VALUES (0),(1),(2),(3),(4),(5),(6),(7),(8),(9)),
         GETALLEN (GETAL) AS
        (SELECT    CIJFER1.CIJFER * 10 + CIJFER2.CIJFER
         FROM      CIJFERS AS CIJFER1,
                   CIJFERS AS CIJFER2)
SELECT   POWER(GETAL,2) AS POWERS
FROM     GETALLEN
WHERE    POWER(GETAL,2) < 5000
INTERSECT
SELECT   POWER(GETAL,3)
FROM     GETALLEN
WHERE    POWER(GETAL,3) < 5000
```

15.4

```
WITH     ALLE_SPELERSNRS (SPELERSNR) AS
        (VALUES  (1)
         UNION ALL
         SELECT   SPELERSNR + 1
         FROM     ALLE_SPELERSNRS
         WHERE    SPELERSNR <
                 (SELECT   MAX(SPELERSNR)
                  FROM     SPELERS))
SELECT   SPELERSNR
FROM     ALLE_SPELERSNRS
EXCEPT
SELECT   SPELERSNR
FROM     SPELERS
```

of

```
WITH     MAX_SPELERSNR (SPELERSNR) AS
        (SELECT   MAX(SPELERSNR)
         FROM     SPELERS),
         ALLE_SPELERSNRS (SPELERSNR) AS
        (VALUES  (1)
         UNION ALL
         SELECT   ALLE_SPELERSNRS.SPELERSNR + 1
         FROM     ALLE_SPELERSNRS, MAX_SPELERSNR
         WHERE    ALLE_SPELERSNRS.SPELERSNR < MAX_SPELERSNR.SPELERSNR)
SELECT   SPELERSNR
FROM     ALLE_SPELERSNRS
EXCEPT
SELECT   SPELERSNR
FROM     SPELERS
```

15.5

```
WITH     RELATIES (SUPER, SUB, AANTAL) AS
        (SELECT   *
         FROM     ONDERDELEN
         WHERE    SUB = 'O8'
         UNION ALL
         SELECT   O.*
         FROM     ONDERDELEN AS O, RELATIES AS R
         WHERE    O.SUB = R.SUPER)
SELECT   SUPER
FROM     RELATIES
```

15.6

```
WITH     RELATIES (SUPER, SUB, AANTAL) AS
        (SELECT   SUPER, SUB, AANTAL
         FROM     ONDERDELEN
         WHERE    SUPER = 'O2'
         UNION ALL
         SELECT   O.SUPER, O.SUB, O.AANTAL*R.AANTAL
         FROM     ONDERDELEN AS O, RELATIES AS R
         WHERE    O.SUPER = R.SUB)
SELECT   SUM(AANTAL)
FROM     RELATIES
```

15.7

```
WITH     RELATIES (TOP, START, EINDE, PAD, TOTALE_TIJDSDUUR, STAP) AS
        (SELECT   START, START, EINDE,
                  CAST(START||'-('||ACTIVITEIT||')-'||EINDE AS VARCHAR(100)),
                  TIJDSDUUR, 1
         FROM     ACTIVITEITEN
         WHERE    START = 'A'
         UNION ALL
         SELECT   R.TOP, A.START,
                  A.EINDE, R.PAD||'-('||A.ACTIVITEIT||')-'||A.EINDE,
                  R.TOTALE_TIJDSDUUR + A.TIJDSDUUR, R.STAP+1
         FROM     ACTIVITEITEN AS A, RELATIES AS R
         WHERE    A.START = R.EINDE
         AND      R.STAP < 20)
SELECT   PAD, TOTALE_TIJDSDUUR
FROM     RELATIES
WHERE    EINDE = 'K'
AND      TOTALE_TIJDSDUUR =
        (SELECT   MAX(TOTALE_TIJDSDUUR)
         FROM     RELATIES
         WHERE    EINDE =  'K')
```

15.8

```
WITH     VLUCHTPLAN(VLUCHTNR, PLAN_VLIEGVELDEN, PLAN_VLUCHTEN,
                    START_VLIEGVELD, EIND_VLIEGVELD, STARTTIJD, EINDTIJD,
                    VERTREK_VLIEGVELD, AANKOMST_VLIEGVELD,
                    VERTREKTIJD, AANKOMSTTIJD, PRIJS, STOPS) AS
        (SELECT   VLUCHTNR, CAST(VERTREK_VLIEGVELD || '->' ||
                  AANKOMST_VLIEGVELD AS VARCHAR(100)),
                  CAST(RTRIM(CHAR(VLUCHTNR)) AS VARCHAR(100)),
                  VERTREK_VLIEGVELD, AANKOMST_VLIEGVELD,
                  VERTREKTIJD, AANKOMSTTIJD,
                  VERTREK_VLIEGVELD, AANKOMST_VLIEGVELD,
                  VERTREKTIJD, AANKOMSTTIJD, PRIJS, 0
         FROM     VLUCHTEN
         WHERE    VERTREK_VLIEGVELD='AMS'
         AND      CAST(VERTREKTIJD AS DATE) = '2007-03-01'
         UNION ALL
         SELECT   P.VLUCHTNR, P.PLAN_VLIEGVELDEN || '->' || F.AANKOMST_VLIEGVELD,
                  P.PLAN_VLUCHTEN || '->' || RTRIM(CHAR(F.VLUCHTNR)),
                  P.START_VLIEGVELD, F.AANKOMST_VLIEGVELD,
                  P.STARTTIJD, F.AANKOMSTTIJD,
                  P.VERTREK_VLIEGVELD, P.AANKOMST_VLIEGVELD,
                  P.VERTREKTIJD, P.AANKOMSTTIJD,
                  P.PRIJS + F.PRIJS, STOPS+1
         FROM     VLUCHTPLAN AS P, VLUCHTEN AS F
         WHERE    P.AANKOMST_VLIEGVELD = F.VERTREK_VLIEGVELD
         AND      P.AANKOMSTTIJD < F.VERTREKTIJD
         AND      F.VERTREK_VLIEGVELD <> 'PHX'
         AND      LOCATE(F.AANKOMST_VLIEGVELD, P.PLAN_VLIEGVELDEN) = 0
         AND      STOPS < 1
         AND      P.AANKOMSTTIJD + 4 HOURS > F.VERTREKTIJD)
SELECT   PLAN_VLIEGVELDEN, PLAN_VLUCHTEN, START_VLIEGVELD, EIND_VLIEGVELD,
         STARTTIJD, EINDTIJD, PRIJS
FROM     VLUCHTPLAN
WHERE    EIND_VLIEGVELD = 'PHX'
AND      PRIJS =
        (SELECT   MIN(PRIJS)
         FROM     VLUCHTPLAN
         WHERE    EIND_VLIEGVELD = 'PHX')
```

1.12 Antwoorden hoofdstuk 16

16.1

```
INSERT INTO BOETES
VALUES (15, 27, '1985-11-08', 75)
```

16.2
```
INSERT   INTO BOETES
SELECT   BETALINGSNR + 1000, SPELERSNR, DATUM, BEDRAG
FROM     BOETES
WHERE    BEDRAG >
        (SELECT   AVG(BEDRAG)
         FROM     BOETES)
UNION
SELECT   BETALINGSNR + 2000, SPELERSNR, DATUM, BEDRAG
FROM     BOETES
WHERE    SPELERSNR = 27
```

16.3
```
UPDATE   SPELERS
SET      GESLACHT = 'F'
WHERE    GESLACHT = 'V'
```

16.4
```
UPDATE   SPELERS
SET      GESLACHT = 'X'
WHERE    GESLACHT = 'V'

UPDATE   SPELERS
SET      GESLACHT = 'V'
WHERE    GESLACHT = 'M'

UPDATE   SPELERS
SET      GESLACHT = 'M'
WHERE    GESLACHT = 'X'
```

of:

```
UPDATE   SPELERS
SET      GESLACHT = CASE GESLACHT
                        WHEN 'V' THEN 'M'
                        ELSE 'V'
                    END
```

16.5
```
UPDATE   BOETES
SET      BEDRAG = BEDRAG * 1.2
WHERE    BEDRAG >
        (SELECT   AVG(BEDRAG)
         FROM     BOETES)
```

16.6
```
DELETE
FROM     BOETES
WHERE    SPELERSNR = 44
AND      YEAR(DATUM) = 1980
```

16.7
```
DELETE
FROM     BOETES
WHERE    SPELERSNR IN
        (SELECT   SPELERSNR
         FROM     WEDSTRIJDEN
         WHERE    TEAMNR IN
                 (SELECT   TEAMNR
                  FROM     TEAMS
                  WHERE    DIVISIE = 'tweede'))
```

16.8
```
DELETE
FROM     SPELERS
WHERE    PLAATS =
        (SELECT   PLAATS
         FROM     SPELERS
         WHERE    SPELERSNR = 28)
AND      SPELERSNR <> 28
```

1.13 Antwoorden hoofdstuk 18

18.1 Ja, een datatype is verplicht.

18.2 Eerst het datatype.

18.3 Eigen woorden.

18.4 Variabele lengte is nuttig als het verschil tussen de langst mogelijke waarde voor een kolom en de gemiddelde lengte groot is. Als beide gelijkwaardig zijn, is een kolom met een vaste lengte aan te bevelen.

18.5
1. CHARACTER(13); elk telefoonnummer in de wereld is maximaal 13 cijfers lang.
2. SMALLINT of DECIMAL(3,0).
3. VARCHAR(50); namen van bedrijven kunnen zeer lang zijn.
4. SMALLINT.
5. DATE.

18.6
```
CREATE TABLE AFDELING (
   AFDNR     CHAR(5) NOT NULL PRIMARY KEY,
   BUDGET    DECIMAL(8,2),
   LOCATIE   VARCHAR(30))
```

18.7
```
CREATE TABLE S_KOPIE AS
(SELECT   *
 FROM     SPELERS
 WHERE    SPELERSNR IS NULL)
```

18.8
```
CREATE TABLE S2_KOPIE AS (SELECT * FROM SPELERS)
```

18.9
```
CREATE TABLE NUMMERS AS
(SELECT   SPELERSNR
 FROM     SPELERS
 WHERE    PLAATS = 'Den Haag')
```

18.10 De TABLES-tabel:

```
CREATOR TABLE_NAME CREATE_TIMESTAMP    COMMENT
------- ---------- ------------------- --------------------
TENNIS  AFDELING   2005-08-29 11:43:48 InnoDB free: 10240 kB
```

De COLUMNS-tabel:

```
TABLE_CREATOR TABLE_NAME COLUMN_NAME COLUMN_NO
------------- ---------- ----------- ---------
TENNIS        AFDELING   AFDNR               1
TENNIS        AFDELING   BUDGET              2
TENNIS        AFDELING   LOCATIE             3

DATA_TYPE CHAR_LENGTH PRECISION SCALE NULLABLE COMMENT
--------- ----------- --------- ----- -------- -------
CHAR                5         ?     ? NO       ?
DECIMAL             ?         8     2 YES      ?
VARCHAR            30         ?     ? YES      ?
```

1.14 Antwoorden hoofdstuk 19

19.1 Een primaire sleutel kan en mag geen null-waarden bevatten. SQL vereist dat voor elke kolom die onderdeel is van een primaire sleutel NOT NULL gedefinieerd wordt.

19.2 Per tabel kan maximaal één primaire sleutel gedefinieerd worden, maar deze is niet verplicht.

19.3
```
CREATE TABLE WEDSTRIJDEN (
   WEDSTRIJDNR   INTEGER NOT NULL,
   TEAMNR        INTEGER NOT NULL,
   SPELERSNR     INTEGER NOT NULL,
   GEWONNEN      INTEGER NOT NULL,
   VERLOREN      INTEGER NOT NULL,
   PRIMARY KEY   (WEDSTRIJDNR))
```

of

```
CREATE TABLE WEDSTRIJDEN (
   WEDSTRIJDNR   INTEGER NOT NULL PRIMARY KEY,
   TEAMNR        INTEGER NOT NULL,
   SPELERSNR     INTEGER NOT NULL,
   GEWONNEN      INTEGER NOT NULL,
   VERLOREN      INTEGER NOT NULL)
```

19.4
1. Kolom K4 in de definitie van de primaire sleutel bestaat niet.
2. Kolom K1 is tweemaal als primaire sleutel gedefinieerd; dit is niet toegestaan.
3. De eerste alternatieve sleutel op de kolom K3 is een deelverzameling van de tweede op de kolommen K2 met K3.

19.5 Refererende sleutels worden gedefinieerd om SQL te laten bewaken dat er geen incorrecte gegevens in de tabellen ingevoerd kunnen worden.

19.6 De volgende mutaties zijn nu niet meer toegestaan:

- Het verwijderen van een speler uit de SPELERS-tabel is nu alleen toegestaan als de desbetreffende speler geen wedstrijd heeft gespeeld.
- Het wijzigen van het spelersnummer van een speler in de SPELERS-tabel is alleen toegestaan als de desbetreffende speler geen wedstrijd heeft gespeeld.
- Het verwijderen van een team uit de TEAMS-tabel is nu alleen toegestaan als voor dat team geen wedstrijden zijn gespeeld.
- Het wijzigen van het teamnummer van een team in de TEAMS-tabel is alleen toegestaan als voor het desbetreffende team geen wedstrijden zijn gespeeld.
- Voor het invoeren van nieuwe spelers in de SPELERS-tabel worden door de refererende sleutels geen beperkingen opgelegd.
- Voor het invoeren van nieuwe teams in de TEAMS-tabel worden door de refererende sleutels geen beperkingen opgelegd.
- Voor het verwijderen van wedstrijden uit de WEDSTRIJDEN-tabel worden door de refererende sleutels geen beperkingen opgelegd.
- Het wijzigen van het spelersnummer van een speler in de WEDSTRIJDEN-tabel is alleen toegestaan als het nieuwe spelersnummer reeds in de SPELERS-tabel voorkomt.
- Het wijzigen van het teamnummer van een team in de WEDSTRIJDEN-tabel is alleen toegestaan als het nieuwe teamnummer reeds in de TEAMS-tabel voorkomt.
- Het invoeren van nieuwe wedstrijden in de WEDSTRIJDEN-tabel is alleen toegestaan als het nieuwe spelersnummer reeds in de SPELERS-tabel voorkomt en het nieuwe teamnummer in de TEAMS-tabel.
-

19.7 Als de refererende tabel en de gerefereerde tabel van een en dezelfde refererende sleutel dezelfde zijn, spreken we van self referential integrity.

19.8 Ja.

19.9 Het niet specificeren van refererende acties is gelijkwaardig aan het specificeren van ON UPDATE RESTRICT en ON DELETE RESTRICT.

19.10 De volgende mutaties zijn nu niet meer toegestaan:

- Het wijzigen van een speler uit de SPELERS-tabel is nu alleen toegestaan als de desbetreffende speler geen wedstrijd heeft gespeeld: ON UPDATE RESTRICT.
- Het verwijderen van het spelersnummer van een speler in de SPELERS-tabel is toegestaan: ON DELETE CASCADE.
- Het verwijderen van een team uit de TEAMS-tabel is niet toegestaan: ON DELETE RESTRICT.
- Het wijzigen van het teamnummer van een team in de TEAMS-tabel is toegestaan: ON UPDATE CASCADE.
- Voor het invoeren van nieuwe spelers in de SPELERS-tabel worden door de refererende sleutels geen beperkingen opgelegd.
- Voor het invoeren van nieuwe teams in de TEAMS-tabel worden door de refererende sleutels geen beperkingen opgelegd.
- Voor het verwijderen van wedstrijden uit de WEDSTRIJDEN-tabel worden door de refererende sleutels geen beperkingen opgelegd.
- Het wijzigen van het spelersnummer van een speler in de WEDSTRIJDEN-tabel is alleen toegestaan als het nieuwe spelersnummer reeds in de SPELERS-tabel voorkomt.
- Het wijzigen van het teamnummer van een team in de WEDSTRIJDEN-tabel is alleen toegestaan als het nieuwe teamnummer reeds in de TEAMS-tabel voorkomt.
- Het invoeren van nieuwe wedstrijden in de WEDSTRIJDEN-tabel is alleen toegestaan als het nieuwe spelersnummer reeds in de SPELERS-tabel voorkomt en het nieuwe teamnummer in de TEAMS-tabel.

19.11
```
CHECK(BEDRAG > 0)
```

19.12
```
CHECK(GEWONNEN > VERLOREN AND GEWONNEN + VERLOREN < 6)
```

19.13
```
CHECK(BEGIN_DATUM BETWEEN '1990-01-01' AND
      COALESCE(EIND_DATUM, '9999-01-01'))
```

1.15 Antwoorden hoofdstuk 20

20.1 De interne bytecodes zijn dan niet gelijk.

20.2
```
SELECT   CHARACTER_SET_NAME, COUNT(*)
FROM     INFORMATION_SCHEMA.COLLATIONS
GROUP BY CHARACTER_SET_NAME
```

20.3
```
EXPRESSIE1 COLLATE utf8 = EXPRESSIE2 COLLATE utf8
```

20.4
```
SELECT CHARSET((SELECT MAX(PLAATS) FROM SPELERS)),
       COLLATION((SELECT MAX(PLAATS) FROM SPELERS))
```

20.5
```
SELECT   PLAATS
FROM     SPELERS
ORDER BY PLAATS COLLATE latin1_danish_ci
```

1.16 Antwoorden hoofdstuk 21

21.1
```
ALTER TABLE BESTUURSLEDEN
   CHANGE FUNCTIE BESTUURSFUNCTIE CHAR(20)
```

21.2
```
ALTER TABLE BESTUURSLEDEN
   MODIFY BESTUURSFUNCTIE CHAR(30)
```

21.3
```
ALTER TABLE SPELERS
   ALTER PLAATS SET DEFAULT 'Den Haag'
```

1.17 Antwoorden hoofdstuk 22

22.1 De TELEFOON-kolom bevat het netnummer en het abonneenummer. Het is daarom beter deze te vervangen door twee kolommen.

221.2 Een determinant van de NETNR-kolom is de PLAATS-kolom (bij elke plaats hoort maximaal één netnummer). Een aparte tabel moet opgezet worden met de kolommen PLAATS (primaire sleutel) en NETNR. De NETNR-kolom verdwijnt dan uit de SPELERS-tabel. De kolommen die we dan overhouden zijn: SPELERSNR, NAAM, VOORLETTERS, GEB_DATUM, GESLACHT, JAARTOE, STRAAT, HUISNR, PLAATS, ABONNEENR en BONDSNR.

22.3 De WEDSTRIJDEN-tabel moet uitgebreid worden met een kolom, genaamd DIVISIE, waarin de divisie van het team waarvoor de wedstrijd gespeeld is, geregistreerd wordt. De SELECT-instructie ziet er dan als volgt uit:

```
SELECT   WEDSTRIJDNR, TEAMNR, DIVISIE
FROM     WEDSTRIJDEN
```

22.4 De SPELERS-tabel moet uitgebreid worden met twee kolommen: GEWONNEN en GEMIDDELD. De eerste kolom bevat het totaal aantal gewonnen wedstrijden van de speler, de tweede kolom geeft het gemiddeld aantal gewonnen wedstrijden aan. De instructie krijgt dan de volgende vorm:

```
SELECT   SPELERSNR, NAAM
FROM     SPELERS
WHERE    GEWONNEN > GEMIDDELD
```

22.5
```
CREATE TABLE SHOWS (
   NAAM_SHOW      CHAR(20) NOT NULL,
   CONFERENCIER   CHAR(20) NOT NULL,
   PRIMARY KEY    (NAAM_SHOW))

CREATE TABLE UITVOERING
   NAAM_SHOW      CHAR(20) NOT NULL,
   LOCATIE        CHAR(20) NOT NULL,
   DATUM          DATE NOT NULL,
   PRIMARY KEY    (NAAM_SHOW, LOCATIE, DATUM),
   FOREIGN KEY    (NAAM_SHOW) REFERENCES SHOWS(NAAM_SHOW))

CREATE TABLE BEZETTING (
   NAAM_SHOW     CHAR(20) NOT NULL,
   MUZIKANT      CHAR(20) NOT NULL,
   INSTRUMENT    CHAR(20) NOT NULL,
   PRIMARY KEY   (NAAM_SHOW, MUZIKANT, INSTRUMENT),
   FOREIGN KEY   (NAAM_SHOW) REFERENCES SHOWS(NAAM_SHOW))
```

1.18 Antwoorden hoofdstuk 23

23.1

1. Basisstrategie:

```
RESULT := [];
FOR EACH T IN TEAMS DO
   IF (T.TEAMNR > 1)
   AND (T.DIVISIE = 'tweede') THEN
      RESULT :+ T;
ENDFOR;
```

Geoptimaliseerde strategie:

```
RESULT := [];
FOR EACH T IN TEAMS
WHERE DIVISIE = 'tweede' DO
   IF T.TEAMNR > 1 THEN
      RESULT :+ T;
ENDFOR;
```

2. Basisstrategie:

```
RESULT := [];
FOR EACH S IN SPELERS DO
   FOR EACH W IN WEDSTRIJDEN DO
      IF S.SPELERSNR = W.SPELERSNR AND
         S.GEB_DATUM > '1963-01-01' THEN
         RESULT :+ S;
   ENDFOR;
ENDFOR;
```

Geoptimaliseerde strategie:

```
RESULT := [];
FOR EACH S IN SPELERS
WHERE S.GEB_DATUM > '1963-01-01' DO
   FOR EACH W IN WEDSTRIJDEN DO
      IF W.SPELERSNR = S.SPELERSNR THEN
         RESULT :+ S;
   ENDFOR;
ENDFOR;
```

1.19 Antwoorden hoofdstuk 24

24.1

```
CREATE   VIEW AANTALSP (TEAMNR, AANTAL) AS
SELECT   TEAMNR, COUNT(*)
FROM     WEDSTRIJDEN
GROUP BY TEAMNR
```

24.2

```
CREATE   VIEW WINNAARS AS
SELECT   SPELERSNR, NAAM
FROM     SPELERS
WHERE    SPELERSNR IN
        (SELECT   SPELERSNR
         FROM     WEDSTRIJDEN
         WHERE    GEWONNEN > VERLOREN)
```

24.3

```
CREATE   VIEW TOTALEN (SPELERSNR, SOM_BOETES) AS
SELECT   SPELERSNR, SUM(BEDRAG)
FROM     BOETES
GROUP BY SPELERSNR
```

24.4

View	Update	Insert	Delete
WOONPLAATSEN	nee	nee	nee
WSPELERS	ja	nee	ja
SOMMIGEN	ja	nee	ja
CIJFERS	nee	nee	nee
HAGENEZEN	ja	nee	ja
INWONERS	nee	nee	nee
VETERANEN	ja	ja	ja
TOTALEN	nee	nee	nee
LEEFTIJDEN	ja	nee	ja

24.5

1.
```
SELECT   YEAR(GEBOORTE) – 1900 AS VERSCHIL, COUNT(*)
FROM    (SELECT   SPELERSNR, NAAM,
                  VOORLETTERS, GEB_DATUM AS GEBOORTE
         FROM     SPELERS
         WHERE    PLAATS = 'Den Haag') AS HAGENAAR
GROUP BY VERSCHIL
```

2.
```
SELECT   DUREN.SPELERSNR
FROM    (SELECT   *
         FROM     SPELERS
         WHERE    SPELERSNR IN
                 (SELECT   SPELERSNR
                  FROM     BOETES)) AS DUREN,
        (SELECT   SPELERSNR, NAAM,
                  VOORLETTERS, GEB_DATUM AS GEBOORTE
         FROM     SPELERS
         WHERE    PLAATS = 'Den Haag') AS HAGENAAR
WHERE    DUREN.SPELERSNR = HAGENAAR.SPELERSNR
```

3.
```
UPDATE   SPELERS
SET      GEB_DATUM = '1950-04-04'
WHERE    SPELERSNR = 7
```

24.6

1. Ja.
2. Ja, maar de view is alleen te raadplegen en niet te muteren, omdat de viewformule een join zal bevatten.

1.20 Antwoorden hoofdstuk 26

26.1

```
CREATE  USER RONALDO IDENTIFIED BY 'NIKE'
```

26.2

```
DROP    USER RONALDO
```

26.3

```
GRANT   SELECT, INSERT
ON      SPELERS
TO      RONALDO
```

26.4

```
GRANT   ALL
ON      BESTUURSLEDEN
TO      PUBLIC
```

26.5

```
GRANT   UPDATE(STRAAT, HUISNR, POSTCODE, PLAATS)
ON      SPELERS
TO      RONALDO
```

26.6
```
GRANT   INSERT
ON      TENNIS.*
TO      JACO, DIANE
```

26.7
```
GRANTOR GRANTEE TABLE_NAME S I D U R WITHGRANTOPTION
------- ------- ---------- - - - - - ---------------
BOEKSQL PUBLIC  SPELERS    Y N N N N NO
BOEKSQL RUDY    SPELERS    N Y N N N YES
RUDY    REGINE  SPELERS    N Y N N N NO
RUDY    SUSAN   SPELERS    N Y N N N YES
SUSAN   REGINE  SPELERS    N Y N N N NO
```

1.21 Antwoorden hoofdstuk 27

27.1

1.
```
CREATE SEQUENCE EVEN_GETALLEN
   START WITH 2
   INCREMENT BY 2
```

2.
```
CREATE SEQUENCE TIENTALLEN
   START WITH 80
   INCREMENT BY -10
```

3.
```
CREATE SEQUENCE VAN_1_TOT_4
   START WITH 1
   INCREMENT BY 1
   MINVALUE 1
   MAXVALUE 4
   NOCACHE
   CYCLE
```

4.
```
CREATE SEQUENCE BIT
   START WITH 0
   MINVALUE 0
   MAXVALUE 1
   NOCACHE
   CYCLE
```

1.22 Antwoorden hoofdstuk 28

28.1 Het meest essentiële verschil tussen een stored procedure en een trigger is dat triggers niet direct door programma's of andere stored procedures aangeroepen kunnen worden.

28.2
```
CREATE TRIGGER SOM_BOETES_250
   AFTER INSERT, UPDATE(BEDRAG) OF BOETES
      FOR EACH ROW
   BEGIN
      SELECT   COUNT(*)
      INTO     AANTAL
      FROM     BOETES
      WHERE    SPELERSNR IN
              (SELECT   SPELERSNR
               FROM     BOETES
               GROUP BY SPELERSNR
               HAVING   SUM(BEDRAG) > 250);
      IF AANTAL > 0 THEN
         ROLLBACK WORK;
      ENDIF;
   END
```

28.3

```
CREATE TRIGGER AANTAL_WEDSTRIJDEN_INSERT
   AFTER INSERT OF WEDSTRIJDEN FOR EACH ROW
   BEGIN
      UPDATE   TEAMS
      SET      AANTAL_WEDSTRIJDEN =
              (SELECT   COUNT(*)
               FROM     WEDSTRIJDEN
               WHERE    TEAMNR = NEW.TEAMNR)
      WHERE    TEAMNR = NEW.TEAMNR
END

CREATE TRIGGER AANTAL_WEDSTRIJDEN_DELETE
   AFTER DELETE, UPDATE OF WEDSTRIJDEN FOR EACH ROW
   BEGIN
      UPDATE   TEAMS
      SET      AANTAL_WEDSTRIJDEN =
              (SELECT   COUNT(*)
               FROM     WEDSTRIJDEN
               WHERE    TEAMNR = OLD.TEAMNR)
      WHERE    TEAMNR = OLD.TEAMNR
END
```

1.23 Antwoorden hoofdstuk 33

33.1

Regel 1 De SELECT-instructie wijzigt de inhoud van tabellen niet, maar start wel een transactie.

Regel 2 Een nog niet permanente mutatie.

Regel 3 Een COMMIT-instructie wordt uitgevoerd. Alle mutaties van de actuele transactie worden permanent gemaakt. Dit is de mutatie van regel 2.

Regel 4 Een ROLLBACK-instructie wordt uitgevoerd. Omdat dit de eerste SQL-instructie is na de vorige COMMIT, begint en eindigt hier een nieuwe transactie. Er zijn geen mutaties uitgevoerd, dus er hoeven ook geen mutaties teruggedraaid te worden.

Regels 5 – 6 Twee nog niet permanente mutaties.

Regel 7 Een ROLLBACK-instructie wordt uitgevoerd. Alle mutaties van de actuele transactie worden ongedaan gemaakt. Dit zijn de mutaties van regels 5 en 6.

Regel 8 Een nog niet permanente mutatie.

Regel 9 Een COMMIT-instructie wordt uitgevoerd. Alle mutaties van de actuele transactie worden permanent gemaakt. Dit is de mutatie van regel 8.

Regel 10 Het programma wordt hier beëindigd. Er is geen actuele transactie, dus het programma kan zonder problemen stoppen.

33.2

Regel 1 Een SELECT-instructie wijzigt de inhoud van tabellen niet, maar start wel een transactie.

Regel 2 Een savepoint wordt gedefinieerd met de naam S1.

Regel 3 Een nog niet permanente mutatie.

Regel 4 Een COMMIT-instructie wordt uitgevoerd. Alle mutaties van de actuele transactie worden permanent gemaakt. Dit is de mutatie van regel 3.

Regel 5 Een nog niet permanente mutatie.

Regel 6 Een savepoint wordt gedefinieerd met de naam S1.

Regel 7 Een nog niet permanente mutatie.

Regel 8 Een ROLLBACK-instructie wordt uitgevoerd. Alleen de mutatie van regel 7 wordt ongedaan gemaakt. De mutatie van regel 5 is nog niet permanent.

Regel 9 Een nog niet permanente mutatie.

Regel 10 Een savepoint wordt gedefinieerd met de naam S2.

Regel 11 Een nog niet permanente mutatie.

Regel 12 Een ROLLBACK-instructie wordt uitgevoerd. Alleen de mutaties van regels 7, 9 en 11 worden ongedaan gemaakt. De mutatie van regel 5 is nog (steeds) niet permanent.

Regel 13 Een COMMIT-instructie wordt uitgevoerd. Alle mutaties van de actuele transactie worden permanent gemaakt. Dit is de mutatie van regel 5.

Regel 14 Het programma wordt hier beëindigd. Er is geen actuele transactie, dus het programma kan zonder problemen stoppen.

1.24 Antwoorden hoofdstuk 34

34.1

1.
```
SELECT   *
FROM     SPELERS
WHERE    PLAATS = 'Den Haag'
AND      STRAAT = 'Erasmusweg'
UNION
SELECT   *
FROM     SPELERS
WHERE    GEB_DATUM < '1960-01-01'
```

2.
```
SELECT   *
FROM     SPELERS
```

3. Opmerking: De conditie `GEWONNEN * VERLOREN = GEWONNEN * 4` kan niet vereenvoudigd worden tot `VERLOREN = 4`, omdat beide kanten van de vergelijking gedeeld worden door GEWONNEN. GEWONNEN kan gelijk zijn aan 0 en dat zou betekenen dat we door 0 delen, hetgeen niet is toegestaan:

```
SELECT   DISTINCT TEAMS.*
FROM     TEAMS, WEDSTRIJDEN
WHERE    TEAMS.TEAMNR = WEDSTRIJDEN.TEAMNR
AND      GEWONNEN * VERLOREN = GEWONNEN * 4
```

4.
```
SELECT   DISTINCT T.TEAMNR
FROM     TEAMS AS T, WEDSTRIJDEN AS W
WHERE    T.TEAMNR = W.TEAMNR
AND      DIVISIE = 'tweede'
```

5.
```
SELECT   SPELERSNR
FROM     SPELERS
```

6.
```
SELECT   SPELERSNR,
         CASE GESLACHT
            WHEN 'V' THEN 'Vrouw'
            ELSE 'Man'
         END
FROM     SPELERS
```

7.
```
SELECT   GEB_DATUM, COUNT(*)
FROM     SPELERS
WHERE    GEB_DATUM >= '1970-01-01'
GROUP BY GEB_DATUM
```

34.2

1.
```
SELECT   SPELERSNR, NAAM, GEB_DATUM
FROM     SPELERS_XXL
WHERE    GEB_DATUM =
        (SELECT   MIN(GEB_DATUM)
         FROM     SPELERS_XXL)
```

2.
```
SELECT   *
FROM     SPELERS_XXL
```

3.
```
SELECT   SPELERSNR,
         CASE GESLACHT
            WHEN 'V' THEN 'Vrouw'
            ELSE 'Man'
         END
FROM     SPELERS_XXL
```

4.
```
SELECT   POSTCODE, COUNT(*)
FROM     SPELERS_XXL
WHERE    POSTCODE >= 'Y'
GROUP BY POSTCODE
```

5.
```
SELECT   *
FROM     SPELERS_XXL
WHERE    SPELERSNR <= 10
```

De Auteur

Rick F. van der Lans is auteur van vele boeken over SQL. Naast dit SQL Leerboek dat in diverse talen vertaald is, waaronder Engels, Duits, Chinees enItaliaans, heeft hij SQL boeken geschreven voor producten als MySQL, Oracle, SQLite, Ingres en Pervasive PSQL.

Hij is onafhankelijk adviesur, auteur en docent gespecialiseerd in databasetechnologie, datawarehousing en applicatie-integratie. Hij is oprichter en directeur van R20/Consultancy. Door de jaren heen heeft hij veel organisaties geadviseerd op het gebied van IT-architecturen.

Als spreker op conferenties en seminars wordt hij internationaal gerespecteerd. Al meer dan vijfentwintig jaar geeft hij over de gehele wereld lezingen, inclusief in de meeste Europese landen, Noord- en Zuid-Amerika en Australië. Hij is voorzitter van het jaarlijkse European Data Warehouse and Business Intelligence Conference. Hij schrijft een column voor Database Magazine en voor het internationale Beye-Network.com. Zeven jaar lang was hij lid van de Nederlandse ISO commissie verantwoordelijk voor ISO SQL Standaard.

Rick kan via de volgende kanalen bereikt worden:

Email: rick@r20.nl
Twitter: http://twitter.com/Rick_vanderlans
LinkedIn: http://www.linkedin.com/pub/rick-van-der-lans/9/207/223

Cursussen over de volgende onderwerpen kunnen door Rick F. van der Lans verzorgd worden

- Database-ontwerp en informatiemodellering
- De basis van SQL
- Het ontwikkelen van geavanceerde SQL queries
- Datawarehousing en business intelligence
- Data virtualisatie

Andere boeken geschreven door Rick F. van der Lans