Syntaxblad SQL

(dit blad krijg je bij de toets uitgereikt)

Volgorde SQL clausules bij SELECT

```
SELECT
  FROM
  JOIN
  WHERE
  GROUP BY
  HAVING
  ORDER BY
```

Voorbeeld:

```
SELECT * FROM Artikelen
  JOIN Orders
       ON Artikelen.order = Orders.id
  WHERE Artikel.prijs > 10
  GROUP BY Artikel.id
  HAVING COUNT(*) > 3
  ORDER BY Orders.datum
```

Commentaar

```
/* commentaarblok */
```

Informatie opvragen

```
SELECT * FROM {tabel};
SELECT {kolom1}, {kolom2} FROM {tabel};
```

Rijen filteren

```
SELECT * FROM {table}
  WHERE {conditie(s)};
```

Voorbeeld:

```
SELECT * FROM Klant
  WHERE leeftijd > 21;
SELECT * FROM Klant
  WHERE leeftijd > 21
  AND postcode = "3067MR";
```

Operators

```
rekenkundig: +, -, *, /
vergelijking: =, !=, <, >,  <=, >=
logisch: OR, AND, NOT
```

Condities met NULL

```
SELECT * FROM {table}
  WHERE {kolom} IS [NOT] NULL;
```

Voorbeeld:

```
SELECT * FROM Persoon
  WHERE partner IS NULL;
SELECT COUNT(*) FROM Persoon
  WHERE partner IS NOT NULL;
```

Condities met LIKE

Met LIKE kun je strings vergelijken met een patroon. Hiervoor maak je gebruik van wildcardsymbolen:

```
  %: 0 of meer tekens
  _: exact 1 teken

SELECT * FROM {table}
  WHERE {kolom} LIKE "{patroon}"
```

Voorbeeld:

```
  SELECT * FROM Werknemer
    WHERE woonplaats LIKE "%dam";
```

Condities met IN

```
SELECT * FROM {tabel}
  WHERE {kolomnaam} IN
  ({waarde1}, {waarde2}, ...);
```

Voorbeeld:

```
SELECT * FROM Werknemer
  WHERE woonplaats IN
  ("Rotterdam, "Amsterdam, "Den Haag");
```

Subqueries

```
SELECT * FROM {tabel}
  WHERE {kolomnaam} IN
    (
    SELECT {kolom} FROM etc...
    )
```

LET OP: als je een subquery met IN gebruikt, moet je verplicht maar één kolom opvragen!

Voorbeeld:

```
SELECT * FROM Werknemer
  WHERE woonplaats IN
  (
    SELECT naam FROM Plaats
    WHERE aantal_inwoners > 100000
  )
```

Resultaten rangschikken

```
SELECT * FROM {table}
  ORDER BY {kolom} [DESC];
```

Voorbeeld:

```
SELECT * FROM Klant
  ORDER BY leeftijd DESC;
```

Geaggregeerde / statistische functies

```
Laagste waarde: MIN(kolomnaam)
Hoogste waarde: MAX(kolomnaam)
Som: SUM(kolomnaam)
Gemiddelde: AVG(kolomnaam)
Aantal rijen met ingevulde
  waarde: COUNT(kolomnaam)
```

Voorbeeld:

```
SELECT MAX(leeftijd) FROM Klant;
SELECT COUNT(id) FROM Klant
  WHERE leeftijd >= 18;
```

Gegevens groeperen

```
SELECT {geaggregeerde functie}
  FROM {tabel}
  GROUP BY {kolom};
```

Voorbeeld:

```
SELECT COUNT(id) FROM Klant
  GROUP BY geslacht;
```

Groepen filteren

```
SELECT {geaggregeerde functie}
  FROM {tabel}
  GROUP BY {kolom}
  HAVING {conditie m.b.t
          geaggregeerde functie};
```

Voorbeeld:

```
SELECT provincie, COUNT(id) FROM Klant
  GROUP BY provincie
  HAVING COUNT(id) > 50;
```

Aangepaste kolomnaam in output

```
SELECT {kolom} AS {nieuwe_naam}
  FROM {tabel};
```

Voorbeeld:

```
SELECT COUNT(id) AS aantal_per_geslacht
  FROM Klant
  GROUP BY geslacht;
```

Tabel uitbreiden met andere tabel

```
SELECT * FROM {tabel1}
  [LEFT OUTER] JOIN {tabel2} ON
    {tabel1}.{verwijzende kolom}
       =
    {tabel2}.{verwezen_kolom};
```

Voorbeeld:

```
SELECT * FROM Klant
  JOIN Lidmaatschap ON
  Klant.lidmaatschap = Lidmaatschap.id;
```

Als je ook een uitbreiding wilt op rijen die bij Klant.lidmaatschap de waarde NULL hebben:

```
  SELECT * FROM Klant
  LEFT OUTER JOIN Lidmaatschap ON
  Klant.lidmaatschap = Lidmaatschap.id;
```

Alias voor tabel

```
SELECT * FROM {tabel} {tabelalias};
```

Let op: gebruik van aliassen zijn noodzakelijk bij wanneer je een tabel met zichzelf uitbreidt.

Voorbeeld:

```
  SELECT * FROM Klant K
    JOIN Lidmaatschap L ON
    K.lidmaatschap = L.id
```

Rijen toevoegen

```
INSERT INTO {tabel} (kolom1, kolom2, ...)
  VALUES ({waarde1}, {waarde2}, ...);
```

Voorbeeld:

```
  INSERT INTO Klant
    (naam, leeftijd, woonplaats)
    VALUES
    ("Jay", 45, "Hazerswoude");
```

Rijen verwijderen

```
DELETE FROM {tabel} [WHERE {conditie}];
```

LET OP: zonder conditie worden alle rijen uit de tabel verwijderd.

Voorbeeld:

```
DELETE FROM Klant WHERE id=443;
DELETE FROM Klant WHERE leeftijd < 18;
```

Veld wijzigen

```
UPDATE {tabel}
  SET {kolomnaam} = {nieuwe waarde}
  [WHERE {conditie}]
```

LET OP: zonder conditie wordt het veld in iedere rij gewijzigd. Ook is het mogelijk met behulp van een conditie voor meerdere rijen tegelijk het veld te wijzigen

Voorbeeld:

```
UPDATE Klant
  SET emailadres = "jan@yahoo.com"
  WHERE id = 59;
UPDATE Klant
  SET geslacht = "man"
  WHERE geslacht = "m";
```