Cheatsheet

Folgendes Cheatsheet ist basiert auf der Datenbank der Social-Media-Plattform InstaHub. In den ersten Befehlen wird insbesondere die Tabelle `users` verwendet. Die Tabelle enthält unter anderem folgende Informationen:

id	username	name	birthday	city	centimeters
1	niclas258	Niclas Schweizer	2001-01-31	Wremen	182
2	rafael54	Rafael Probst	2004-08-06	Leipzig	187
3	luis52	Luis Krüger	2004-12-15	Lautertal	173
...	...	...	...	...	...

Daneben enthält die Tabelle noch viele weitere Spalten mit Informationen zum Land, Geschlecht usw.

Spalten auswählen: SELECT

Auswahl aller Spalten der Tabelle:

```
SELECT *
FROM users
```

Auswahl der Spalten `city` und `gender` der Tabelle `users`:

```
SELECT city, gender
FROM users
```

Duplikate löschen: DISTINCT

Mit dem Zusatz `DISTINCT` werden Duplikate aus den Resultaten gelöscht.

```
SELECT DISTINCT gender
FROM users
```

Sortieren: ORDER BY

Mit dem Befehl `ORDER BY` können Resultate sortiert werden, nach einer oder mehreren Spalten. Mit folgendem Befehl erhalten wir die Namen aller Users, sortiert nach Stadt und nach Region:

```
SELECT name, city
FROM users
ORDER BY city ASC, name ASC
```

- `DESC` = absteigend (*descending*)
- `ASC` = aufsteigend (*ascending*)

Erste / Letzte Zeilen: LIMIT

Mit dem Befehl `LIMIT` können eine Tabelle auf die ersten n Zeilen beschränkt werden. Z.B. können die Namen der ersten drei Personen der Tabelle `users` wie folgt abgefragt werden:

```
SELECT name
FROM users
LIMIT 3
```

Zeilen Filtern: WHERE

Mit dem Befehl `WHERE` können die Resultate einer Abfrage nach eigenen Kriterien gefiltert werden:

```
SELECT name, city, gender
FROM users
WHERE gender='male'
```

Operatoren	Bedeutung
`=`	gleich (=)
`<>`	ungleich (≠)
`<`	kleiner als (<)
`<=`	kleiner oder gleich (≤)
`>`	grösser als (>)
`>=`	grösser oder gleich (≥)
`BETWEEN x AND y`	ein Wert zwischen x und y
`IN (wert1, wert2, ...)`	einer von mehreren Werten
`IS NULL`	Wert ist leer

Filter kombinieren: AND, OR

Mehrere Filter können mit `AND` („und“ → beides muss wahr sein) oder `OR` („oder“ → eine der beiden Bedingungen muss wahr sein) verbunden werden.

Users aus Leipzig, Berlin oder Hamburg, welche grösser als 170 sind:

```
SELECT name, city
FROM users
WHERE city IN ('Leipzig','Berlin',
    'Hamburg')
 AND centimeters > 170
```

Ungefähre Treffer: LIKE

Mit folgendem Befehl erhalten wir alle Städte, die mit „Be...“ beginnen:

```
SELECT DISTINCT city
FROM users
WHERE city LIKE 'Be%'
```

Das Prozentzeichen (%) ist ein *Platzhalter* und steht für „irgendetwas kommt hier (oder nicht)“. In diesem Beispiel bedeutet das, dass 0, 1, oder mehr Zeichen auf das „Be...“ folgen können.

Aggregatsfunktionen

Mit Aggregatsfunktionen werden Daten zusammengefasst, beispielsweise um die Anzahl Zeilen (`COUNT(*)`), die Anzahl nicht leerer Zellen in einer Spalte (`COUNT(spalte)`) oder die Anzahl unterschiedlicher Werte in einer Spalte (`COUNT(DISTINCT spalte)`) zu zählen oder um das Maximum, das Minimum, die Summe oder den Mittelwert einer Spalte zu berechnen (`MAX(spalte)` / `MIN(spalte)` / `SUM(spalte)` / `AVG(spalte)`). Z.B. berechnet folgender Ausdruck die Anzahl Users in Leipzig:

```
SELECT city, COUNT(*)
FROM users
WHERE city = 'Leipzig'
```

Rechnen und umbenennen: AS

Mit Spalten sowie Aggregatsfunktionen kann man rechnen, beispielsweise um ein Resultat durch eine andere Zahl zu dividieren. Zudem können Spalten-Titel mit dem Befehl `AS` umbenannt werden:

```
SELECT centimeters/100 AS 'Grösse
    in Metern'
FROM users
```

Gruppieren: GROUP BY

Aggregatsfunktionen können auch pro Gruppe verwendet werden, z.B. um die Anzahl Mitglieder in jeder Stadt zu berechnen:

```
SELECT city, COUNT(*) AS 'Users
    pro Stadt'
FROM users
GROUP BY city
```

Filtern *nach* Gruppieren: HAVING

Mit `HAVING` können Resultate *nach dem* Gruppieren gefiltert werden. `WHERE` filtert *vor* dem Gruppieren und steht demnach immer vor einem `GROUP BY`.

Durchschnittliche Körpergrösse aller männlichen Mitglieder in jeder Stadt berechnen, danach auf Städte beschränken, in denen die Menschen durchschnittlich zwischen 150 und 155 gross sind:

```
SELECT city, AVG(centimeters) AS '
    Körpergrösse'
FROM users
WHERE gender = 'male'
GROUP BY city
HAVING `Körpergrösse` BETWEEN 150
    AND 155
```

Texte werden immer innerhalb von Anführungszeichen (`''`) geschrieben. Spaltennamen, welche Spezialzeichen oder Abstände enthalten, werden innerhalb von *backticks* (`` ` ``) geschrieben.

Tabellen erstellen: CREATE TABLE

Eine Tabelle `meinetabelle` kann wie folgt erstellt werden:

```
CREATE TABLE meinetabelle(
  name_spalte1 spaltentyp1,
  name_spalte2 spaltentyp2,
  ... --etc.
)
```

Gängige Spalten-Typen sind:

`INTEGER`	Ganzzahl Z.B. 35
`REAL`	Kommazahl Z.B. 3.341
`DATE`	Datum Format: `'YYYY-MM-DD'`
`BOOLEAN`	Wahrheitswert Wahr (1) oder Falsch (0)
`VARCHAR(n)`	Text n = maximale Länge

Tabellen ändern: ALTER

Mit dem Befehl `ALTER tabellenname` können Tabellen verändert werden, beispielsweise um eine Spalte hinzuzufügen oder umzubenennen.

```
ALTER TABLE tabellenname
ADD neue_spalte eigenschaften
```

```
ALTER TABLE tabellenname
RENAME COLUMN name_vorher TO
    name_neu;
```

Tabellen löschen: DROP TABLE

```
DROP TABLE tabellenname
```

Überprüfen, ob existiert: IF EXISTS

Wenn man eine Tabelle erstellen will, welche es bereits gibt, kann eine Fehlermeldung erscheinen. Folgender Befehl wird nur ausgeführt, falls es noch keine Tabelle `testtabelle` gibt:

```
CREATE TABLE IF NOT EXISTS
    testtabelle(
  name_spalte1 spaltentyp1,
  name_spalte2 spaltentyp2,
  ... --etc.
)
```

Ähnlich kann vor man den Befehl `DROP TABLE` anpassen, so dass er nur ausgeführt wird, wenn es eine solche Tabelle wirklich gibt:

```
DROP TABLE IF EXISTS testtabelle
```

Daten einfügen: INSERT INTO

Neue Einträge (Zeilen) können mit dem Befehl `INSERT INTO` in eine Tabelle eingefügt werden. Beispielsweise können mit folgender Befehlsstruktur drei neue Zeilen eingefügt werden:

```
INSERT INTO tabellenname
    (spalte1, spalte2, ...)
VALUES
    (wert1, wert2, ...),
    (wert1, wert2, ...),
    (wert1, wert2, ...)
```

Zuerst werden also die Spalten angegeben, für welche Werte eingefügt werden, danach die Werte für jede neue Zeile. Werte in nicht angegebenen Spalten bleiben leer.

Daten verändern: UPDATE

Existierende Daten können wie folgt geändert werden:

```
UPDATE tabelle_name
SET spalte1 = wert1, ...
WHERE bedingung(en)
```

Beispielsweise können mit folgendem Befehl alle Users, die zur Stadt Berlin gehören, der Stadt Bern zugeordnet werden:

```
UPDATE users
SET city='Bern'
WHERE city='Berlin'
```

Einträge löschen: DELETE FROM

Einzelne Einträge (Zeilen) können mit einer `WHERE`-Bedingung und dem Befehl `DELETE FROM` gelöscht werden:

```
DELETE FROM users
WHERE username='guenther37'
```

Mehrere Tabellen verbinden: JOIN

Mehrere Tabellen können mit folgendem Befehl zu einer Tabelle verbunden werden:

```
SELECT t1.spalte_x, t2.spalte_y,
    ...
FROM tabelle1 AS t1
JOIN tabelle2 AS t2
ON t1.spalte_id1 = t2.spalte_id2
```

Falls Primär- und Fremdschlüssel in beiden Tabellen gleich heissen, kann man `USING` verwenden:

```
SELECT *
FROM tabelle1
JOIN tabelle2 USING (spalte_id)
```

Unterabfragen (Subqueries)

Unterabfragen (en. *subqueries*) können wie folgt geschrieben werden:

```
SELECT spalte11
FROM tabelle1
WHERE spaltenname IN
    (SELECT spalte2 FROM tabelle2
    WHERE ...);
```

Dabei können alle bekannten Vergleichsoperatoren wie `=`, `IN`, `>`, `<`, usw. verwendet werden.

Mehrere Ausdrücke verbinden

Mehrere SQL-Ausdrücke können mit Semikolon (`;`) verbunden werden:

```
-- Tabelle erstellen, danach
    wieder löschen
CREATE TABLE IF NOT EXISTS
    testtabelle(
  name_spalte1 spaltentyp1,
  name_spalte2 spaltentyp2,
  ... --etc.
);
DROP TABLE IF EXISTS testtabelle
```

Kommentare

Kommentare werden von SQL ignoriert und dienen der besseren Leserlichkeit des Codes. Kommentare werden durch zwei Bindestriche gekennzeichnet (siehe Box zu „Mehrere Ausdrücke verbinden“)

Weitere Befehle & Feedback

Weitere SQL-Befehle und Erklärungen finden Sie unter w3schools.com.

Verbesserungsvorschläge können gerne an Cyril Blum geschickt werden.