```sql
PRAGMA auto_vacuum = 1;
PRAGMA automatic_index = 1;
PRAGMA case_sensitive_like = 0;
PRAGMA defer_foreign_keys = 0;
PRAGMA encoding = "UTF-8";
PRAGMA foreign_keys = 1;
PRAGMA ignore_check_constraints = 0;
PRAGMA journal_mode = WAL;
PRAGMA query_only = 0;
PRAGMA recursive_triggers = 1;
PRAGMA reverse_unordered_selects = 0;
PRAGMA secure_delete = 0;
PRAGMA synchronous = NORMAL;
--.headers ON

/*=========================================
 *=============== TABLES =================
 *=========================================*/

SELECT '=========================================';
SELECT '=============== TABLES ==================';
SELECT '=========================================';
SELECT '';

CREATE TABLE Adresse (
    Strasse VARCHAR(70) NOT NULL CONSTRAINT Strassenname CHECK (
        Strasse NOT GLOB ('*[0-9]*') AND
        LENGTH(CAST(Strasse AS VARCHAR)) > 0),
    Hausnummer VARCHAR(10) NOT NULL CONSTRAINT Hausnummer CHECK(
        CAST (substr(Hausnummer, 1, LENGTH(Hausnummer) - 1) AS INTEGER)
        == substr(Hausnummer, 1, LENGTH(Hausnummer) - 1) AND
        CAST (substr(Hausnummer, LENGTH(Hausnummer)) AS TEXT)
        == substr(Hausnummer, LENGTH(Hausnummer)) OR
        CAST (Hausnummer AS INTEGER) == Hausnummer),
    PLZ DECIMAL(5,0) NOT NULL CONSTRAINT PLZ CHECK(
        LENGTH(CAST(PLZ AS VARCHAR)) == 5),
    Ort VARCHAR(40) NOT NULL CONSTRAINT Ort CHECK (
        Ort NOT GLOB ('*[0-9]*') AND
        LENGTH(CAST(Ort AS VARCHAR)) > 0),
    Adressen_ID INTEGER PRIMARY KEY NOT NULL
    );

CREATE TABLE Kunde (
    E_Mail_Adresse VARCHAR(320) NOT NULL CONSTRAINT E_Mail_Adresse CHECK (
        E_Mail_Adresse LIKE '%_@_%.__%'),
    Vorname VARCHAR(50) NOT NULL CONSTRAINT Vorname CHECK (
        Vorname NOT GLOB ('*[0-9]*') AND
        LENGTH(CAST(Vorname AS VARCHAR)) > 0),
    Nachname VARCHAR(30) NOT NULL CONSTRAINT Nachname CHECK (
        Nachname NOT GLOB ('*[0-9]*') AND
        LENGTH(CAST(Nachname AS VARCHAR)) > 0),
    Passwort VARCHAR(64) NOT NULL CONSTRAINT Passwort CHECK (
        LENGTH(CAST(Passwort AS VARCHAR)) >= 6),
    Adressen_ID INTEGER NOT NULL,
    PRIMARY KEY(E_Mail_Adresse),
    FOREIGN KEY (Adressen_ID) REFERENCES Adresse (Adressen_ID)
    );
```

```
58
59  CREATE TABLE Premiumkunde (
60      Ablaufdatum VARCHAR NOT NULL CONSTRAINT Ablaufdatum CHECK (
61          DATE(Ablaufdatum) IS NOT NULL),
62      Studierendenausweis BLOB,
63      E_Mail_Adresse VARCHAR(320) NOT NULL,
64      PRIMARY KEY (E_Mail_Adresse),
65      FOREIGN KEY (E_Mail_Adresse) REFERENCES Kunde (E_Mail_Adresse)
66      );
67
68  CREATE TABLE Angestellter (
69      Jobbezeichnung VARCHAR(30) NOT NULL CONSTRAINT Jobbezeichnung CHECK (
70          LENGTH(CAST(Jobbezeichnung AS VARCHAR)) > 0),
71      Gehalt INTEGER NOT NULL CONSTRAINT Gehalt CHECK (
72          Gehalt >= 0 AND (
73              TYPEOF(Gehalt) == 'real' OR
74              TYPEOF(Gehalt) == 'integer'
75          )
76      ),
77      E_Mail_Adresse VARCHAR(320) NOT NULL,
78      PRIMARY KEY (E_Mail_Adresse),
79      FOREIGN KEY (E_Mail_Adresse) REFERENCES Kunde (E_Mail_Adresse)
80      );
81
82  CREATE TABLE Lieferdienst(
83      Lieferdienst_Bezeichnung VARCHAR(40) NOT NULL
84          CONSTRAINT Lieferdienst_Bezeichnung CHECK (
85              LENGTH(CAST(Lieferdienst_Bezeichnung AS VARCHAR)) > 0),
86      Versandkosten DECIMAL(3,2) NOT NULL CONSTRAINT Versandkosten CHECK (
87          Versandkosten >= 0 AND (
88              TYPEOF(Versandkosten) == 'real' OR
89              TYPEOF(Versandkosten) == 'integer'
90          )
91      ),
92      PRIMARY KEY (Lieferdienst_Bezeichnung)
93      );
94
95  CREATE TABLE Warenkorb (
96      Bestelldatum VARCHAR
97          CONSTRAINT Bestelldatum CHECK (
98              (
99                  Bestelldatum IS NOT NULL AND
100                 DATE(Bestelldatum) IS NOT NULL
101             ) OR (
102                 Bestelldatum IS NULL
103             )
104         ),
105     Bestellstatus VARCHAR(20) NOT NULL CONSTRAINT Bestellstatus CHECK (
106         Bestellstatus IN (
107             'In Bearbeitung', 'Storniert', 'Bezahlt', 'Versandfertig',
108             'Versendet', 'Abgeschlossen'
109         )
110     ),
111     Warenkorb_ID INTEGER PRIMARY KEY NOT NULL,
112     E_Mail_Adresse VARCHAR(320) NOT NULL,
113     Lieferdienst_Bezeichnung VARCHAR(40) NOT NULL,
114     Lieferdatum VARCHAR
```

```
115         CONSTRAINT Lieferdatum CHECK (
116             (
117                 Lieferdatum IS NOT NULL AND
118                 DATE(Lieferdatum) IS NOT NULL AND
119                 Lieferdatum > CURRENT_DATE
120             ) OR (
121                 Lieferdatum IS NULL
122             )
123         ),
124     FOREIGN KEY (E_Mail_Adresse) REFERENCES Kunde (E_Mail_Adresse),
125     FOREIGN KEY (Lieferdienst_Bezeichnung)
126         REFERENCES Lieferdienst (Lieferdienst_Bezeichnung)
127     );
128
129 CREATE TABLE Lieferabo (
130     Intervall INTEGER NOT NULL CONSTRAINT Intervall CHECK (
131         TYPEOF(Intervall) == 'integer' AND
132         Intervall > 0),
133     Beginn VARCHAR NOT NULL CONSTRAINT Beginn_Datum CHECK (
134         DATE(Beginn) IS NOT NULL),
135     Ende VARCHAR NOT NULL CONSTRAINT Ende_Datum CHECK (
136         DATE(Ende) IS NOT NULL),
137     Warenkorb_ID INTEGER NOT NULL,
138     PRIMARY KEY (Warenkorb_ID),
139     FOREIGN KEY (Warenkorb_ID) REFERENCES Warenkorb (Warenkorb_ID),
140     check (Beginn < Ende)
141     );
142
143 CREATE TABLE Newsletter (
144     Betreff VARCHAR(40) NOT NULL CONSTRAINT Betreff CHECK (
145         LENGTH(CAST(Betreff AS VARCHAR)) > 0),
146     Text TEXT NOT NULL CONSTRAINT Text CHECK (
147         LENGTH(CAST(Text AS TEXT)) > 0),
148     Datum VARCHAR NOT NULL
149         DEFAULT CURRENT_DATE
150         --ON UPDATE CURRENT_DATE
151         CONSTRAINT Newsletter_Datum CHECK (
152             DATE(Datum) IS NOT NULL),
153     Newsletter_ID INTEGER PRIMARY KEY NOT NULL,
154     E_Mail_Adresse VARCHAR(320) NOT NULL,
155     FOREIGN KEY (E_Mail_Adresse) REFERENCES Angestellter (E_Mail_Adresse)
156     );
157
158 CREATE TABLE Newsletterabo (
159     E_Mail_Adresse VARCHAR(320) NOT NULL,
160     Newsletter_ID INTEGER NOT NULL,
161     PRIMARY KEY (E_Mail_Adresse, Newsletter_ID),
162     FOREIGN KEY (E_Mail_Adresse) REFERENCES Kunde (E_Mail_Adresse),
163     FOREIGN KEY (Newsletter_ID) REFERENCES Newsletter (Newsletter_ID)
164     );
165
166 CREATE TABLE Artikel (
167     Bezeichnung VARCHAR(40) NOT NULL CONSTRAINT Bezeichnung CHECK (
168         LENGTH(CAST(Bezeichnung AS VARCHAR)) > 0),
169     Beschreibung TEXT NOT NULL CONSTRAINT Beschreibung CHECK (
170         LENGTH(CAST(Beschreibung AS TEXT)) > 0),
171     Bild BLOB DEFAULT NULL,
```

```
172      Artikel_ID INTEGER PRIMARY KEY NOT NULL
173      );
174
175  CREATE TABLE Angebot (
176      Angebots_ID INTEGER PRIMARY KEY NOT NULL,
177      Artikel_ID INTEGER NOT NULL,
178      Preis DECIMAL(10,2) NOT NULL CONSTRAINT Preis CHECK (
179          Preis >= 0 AND (
180              TYPEOF(Preis) == 'real' OR
181              TYPEOF(Preis) == 'integer'
182          )
183      ),
184      FOREIGN KEY (Artikel_ID) REFERENCES Artikel (Artikel_ID)
185      );
186
187  CREATE TABLE Anbieter (
188      Anbieterbezeichnung VARCHAR(40) NOT NULL CONSTRAINT Anbieterbezeichnung CHECK (
189          LENGTH(CAST(Anbieterbezeichnung AS VARCHAR)) > 0),
190      PRIMARY KEY (Anbieterbezeichnung)
191      );
192
193  CREATE TABLE Angebot_im_Warenkorb (
194      Angebots_ID INTEGER NOT NULL,
195      Anbieterbezeichnung VARCHAR(40) NOT NULL,
196      Warenkorb_ID INTEGER NOT NULL,
197      Anzahl INTEGER NOT NULL CONSTRAINT Anzahl CHECK (
198          TYPEOF(Anzahl) == 'integer' AND
199          Anzahl >= 1),
200      PRIMARY KEY (Angebots_ID, Anbieterbezeichnung, Warenkorb_ID),
201      FOREIGN KEY (Angebots_ID) REFERENCES Angebot (Angebots_ID),
202      FOREIGN KEY (Anbieterbezeichnung) REFERENCES Anbieter (Anbieterbezeichnung),
203      FOREIGN KEY (Warenkorb_ID) REFERENCES Warenkorb (Warenkorb_ID)
204      );
205
206  CREATE TABLE Anbieter_bietet_an (
207      Anbieterbezeichnung VARCHAR(40) NOT NULL,
208      Angebots_ID INTEGER NOT NULL,
209      Bestand INTEGER NOT NULL CONSTRAINT Bestand CHECK (
210          TYPEOF(Bestand) == 'integer' AND
211          Bestand >= 0),
212      PRIMARY KEY (Anbieterbezeichnung, Angebots_ID),
213      FOREIGN KEY (Anbieterbezeichnung) REFERENCES Anbieter (Anbieterbezeichnung),
214      FOREIGN KEY (Angebots_ID) REFERENCES Angebot (Angebots_ID)
215      );
216
217  CREATE TABLE Artikel_im_Newsletter (
218      Newsletter_ID INTEGER NOT NULL,
219      Artikel_ID INTEGER NOT NULL,
220      PRIMARY KEY(Newsletter_ID, Artikel_ID),
221      FOREIGN KEY(Newsletter_ID) REFERENCES Newsletter (Newsletter_ID),
222      FOREIGN KEY(Artikel_ID) REFERENCES Artikel (Artikel_ID)
223      );
224
225  CREATE TABLE Artikel_empfiehlt_Artikel (
226      Artikel_ID1 INTEGER NOT NULL,
227      Artikel_ID2 INTEGER NOT NULL,
228      PRIMARY KEY (Artikel_ID1, Artikel_ID2),
```

```
229      FOREIGN KEY (Artikel_ID1) REFERENCES Artikel (Artikel_ID),
230      FOREIGN KEY (Artikel_ID2) REFERENCES Artikel (Artikel_ID)
231      );
232
233  CREATE TABLE Schlagwort (
234      Schlagwort VARCHAR(30) NOT NULL CONSTRAINT Schlagwort CHECK (
235          LENGTH(CAST(Schlagwort AS VARCHAR)) > 0),
236      PRIMARY KEY (Schlagwort),
237      UNIQUE (Schlagwort COLLATE NOCASE)
238      );
239
240  CREATE TABLE Artikel_gehoert_zu_Schlagwort (
241      Artikel_ID INTEGER NOT NULL,
242      Schlagwort VARCHAR(30) NOT NULL,
243      PRIMARY KEY (Artikel_ID, Schlagwort),
244      FOREIGN KEY (Artikel_ID) REFERENCES Artikel(Artikel_ID),
245      FOREIGN KEY (Schlagwort) REFERENCES Schlagwort(Schlagwort)
246      );
247
248  /*=========================================
249   *================ TRIGGER ================
250   *=========================================*/
251
252  SELECT '=========================================';
253  SELECT '================ TRIGGER ================';
254  SELECT '=========================================';
255  SELECT '';
256
257  CREATE TRIGGER update_newsletter_datum
258  AFTER UPDATE ON Newsletter
259  WHEN NEW.Datum < CURRENT_DATE
260  BEGIN
261      UPDATE Newsletter SET Datum = CURRENT_DATE
262          WHERE E_Mail_Adresse = NEW.E_Mail_Adresse;
263  END;
264
265  CREATE TRIGGER newsletter_zu_viele_artikel
266  BEFORE INSERT ON Artikel_im_Newsletter
267  WHEN EXISTS (
268      SELECT COUNT(*) FROM Artikel_im_Newsletter
269      GROUP BY Artikel_im_Newsletter.Newsletter_ID
270      HAVING COUNT(*) > 9
271  )
272  BEGIN
273      SELECT RAISE (ABORT, 'Maximal 10 Artikel im Newsletter!');
274  END;
275
276  /* TODO: newsletter_zu_wenig_artikel? Aber wie lässt sich das realisieren,
277   *       wenn die Artikel dem Newsletter nacheinander hinzugefügt werden,
278   *       außer mit COMMIT? Und das würde bedeuten, dass man die UI um-
279   *       programmieren müsste. -- Aussage des Korrektors: Muss nicht
280   *       implementiert werden.
281   */
282
283  /* Wenn ein Angebot in den Warenkorb gelegt werden soll, überprüft dieser
284   * Trigger, ob der gewählte Anbieter dieses Angebot noch in der
285   * gewünschten Anzahl anbietet.
```

```
 */

/* Wenn ein Angebot eines Anbieters in einen Warenkorb gelegt werden soll,
 * der schon das Angebot dieses Anbieters enthält, dann soll die Anzahl
 * dieses Angebotes in diesem Warenkorb erhöht werden, statt das Angebot
 * ein zweites Mal in den Warenkorb zu legen, da der Warenkorb nicht
 * zweimal dasselbe Angebot desselben Anbieters enthalten kann.
 */

/* Dieser Trigger kombiniert die Aufgaben der beiden oben beschriebenen
 * Trigger in einem, da für die Ausführung des zweiten Triggers das
 * RAISE(IGNORE)-Statement benötigt wird, um zu verhindern, dass der
 * ursprüngliche, fehlschlagende INSERT, der ja durch den Trigger ersetzt
 * wird, dennoch zu einem Fehler führt. Dies führt aber auch dazu, dass
 * andere Trigger ignoriert werden, wenn dieser Trigger ausgelöst wurde.
 * Konkret: Wurde also ein zweites Mal dasselbe Angebot desselben
 * Anbieters in denselben Warenkorb gelegt, wurde zwar die Anzahl dieses
 * Angebots dieses Anbieters in diesem Warenkorb erhöht, aber nicht mehr
 * überprüft, ob diese Anzahl dieses Angebotes bei diesem Anbieter
 * überhaupt vorhanden ist, da der dafür zuständige andere Trigger nicht
 * mehr ausgeführt wurde. Deshalb kombiniert dieser Trigger beide Trigger
 * in einem. Zusätzlich wurde noch ein dritter Trigger eingefügt, welcher
 * überprüft, ob das Angebot überhaupt von dem gewünschten Anbieter
 * angeboten wird.
 */
CREATE TRIGGER angebot_im_warenkorb
BEFORE INSERT ON Angebot_im_Warenkorb
/* Diese erste WHEN-Klausel überprüft, ob das Angebot noch in der ge-
 * wünschten Anzahl bei diesem Anbieter vorhanden ist.
 */
WHEN(
    SELECT (
        SELECT Bestand
        FROM Anbieter_bietet_an
        WHERE Angebots_ID = NEW.Angebots_ID
        AND Anbieterbezeichnung = NEW.Anbieterbezeichnung
    ) -
    (
        SELECT SUM(Anzahl)
        FROM Angebot_im_Warenkorb
        WHERE Angebots_ID = NEW.Angebots_ID
        AND Anbieterbezeichnung = NEW.Anbieterbezeichnung
    )
) <= NEW.Anzahl
/* Diese zweite WHEN-Klausel überprüft, ob der Warenkorb bereits eine
 * bestimmte Anzahl dieses Angebotes dieses Anbieters enthält.
 */
OR EXISTS (
    SELECT Anzahl
    FROM Angebot_im_Warenkorb
    WHERE Angebots_ID = NEW.Angebots_ID
    AND Anbieterbezeichnung = NEW.Anbieterbezeichnung
    AND Warenkorb_ID = NEW.Warenkorb_ID
)
/* Diese dritte WHEN-Klausel überprüft, ob das Angebot überhaupt von dem
 * gewünschten Anbieter angeboten wird.
 */
```

```
343  OR NOT EXISTS (
344      SELECT Bestand
345      FROM Anbieter_bietet_an
346      WHERE Angebots_ID = NEW.Angebots_ID
347      AND Anbieterbezeichnung = NEW.Anbieterbezeichnung
348  )
349  BEGIN
350      /* Dieses erste SELECT-Statement bricht die Transaktion ab, wenn
351       * das Angebot nicht mehr in der gewünschten Anzahl beim Anbieter
352       * vorhanden ist. Dazu muss erneut die erste WHEN-Bedingung als
353       * WHERE-Klausel dieses SELECT-Statements überprüft werden, damit
354       * dieses SELECT-Statement nicht ausgeführt wird, wenn der Trigger
355       * aufgrund der zweiten WHEN-Klausel ausgelöst wurde.
356       */
357      SELECT RAISE (ABORT, 'Gewünschte Anzahl dieses Angebots bei diesem Anbieter
         nicht verfügbar!')
358      WHERE(
359          SELECT (
360              SELECT Bestand
361              FROM Anbieter_bietet_an
362              WHERE Angebots_ID = NEW.Angebots_ID
363              AND Anbieterbezeichnung = NEW.Anbieterbezeichnung
364          ) -
365          (
366              SELECT SUM(Anzahl)
367              FROM Angebot_im_Warenkorb
368              WHERE Angebots_ID = NEW.Angebots_ID
369              AND Anbieterbezeichnung = NEW.Anbieterbezeichnung
370          )
371      ) <= NEW.Anzahl;
372      /* Dieses zweite UPDATE-Statement erhöht die Anzahl des Angebotes
373       * des Anbieters im Warenkorb um die gewünschte Anzahl, wenn dieses
374       * bereits im Warenkorb enthalten wird. Auch hier muss erneut die
375       * entsprechende WHEN-Bedingung als WHERE-Klausel geprüft werden,
376       * damit dies nicht geschieht, wenn der Trigger oben aufgrund der
377       * ersten WHEN-Klausel ausgelöst wurde.
378       */
379      UPDATE Angebot_im_Warenkorb
380      SET Anzahl = Anzahl + NEW.Anzahl
381      WHERE Angebots_ID = NEW.Angebots_ID
382      AND Anbieterbezeichnung = NEW.Anbieterbezeichnung
383      AND Warenkorb_ID = NEW.Warenkorb_ID
384      AND EXISTS (
385          SELECT Anzahl
386          FROM Angebot_im_Warenkorb
387          WHERE Angebots_ID = NEW.Angebots_ID
388          AND Anbieterbezeichnung = NEW.Anbieterbezeichnung
389          AND Warenkorb_ID = NEW.Warenkorb_ID
390      );
391      /* Gleiches gilt für dieses RAISE-Statement, welches ebenfalls nur
392       * ausgeführt werden soll, wenn die zweite WHEN-Klausel zur
393       * Auslösung des Triggers geführt hat.
394       */
395      SELECT RAISE (IGNORE)
396      WHERE EXISTS (
397          SELECT Anzahl
398          FROM Angebot_im_Warenkorb
```

```
399             WHERE Angebots_ID = NEW.Angebots_ID
400             AND Anbieterbezeichnung = NEW.Anbieterbezeichnung
401             AND Warenkorb_ID = NEW.Warenkorb_ID
402         );
403         /* Dieses vierte RAISE-Statement hat die Aufgabe, eine Fehlermeldung
404          * auszugeben und die Transaktion abzubrechen, wenn ein Angebot von
405          * einem Anbieter in den Warenkorb gelegt werden soll, der dieses
406          * Angebot nicht anbietet.
407          */
408         SELECT RAISE (ABORT, 'Anbieter bietet dieses Angebot nicht an!')
409         WHERE NOT EXISTS (
410             SELECT Bestand
411             FROM Anbieter_bietet_an
412             WHERE Angebots_ID = NEW.Angebots_ID
413             AND Anbieterbezeichnung = NEW.Anbieterbezeichnung
414         );
415     END;
416
417     /*=========================================
418      *================ INSERTS ================
419      *=========================================*/
420
421     SELECT '=========================================';
422     SELECT '================ INSERTS ================';
423     SELECT '=========================================';
424     SELECT '';
425
426     -----------------------------------------
427     ----------------- Adresse -----------------
428     -----------------------------------------
429
430     SELECT 'Testing: 3 funktionierende Tests: Adresse';
431     INSERT INTO Adresse (Strasse, Hausnummer, PLZ, Ort, Adressen_ID)
432         VALUES ('Musterstrasse', '1', 11111, 'Musterort', NULL);
433     INSERT INTO Adresse (Strasse, Hausnummer, PLZ, Ort, Adressen_ID)
434         VALUES ('Musterweg', '2A', 22222, 'Musterstadt', NULL);
435     INSERT INTO Adresse (Strasse, Hausnummer, PLZ, Ort, Adressen_ID)
436         VALUES ('Musterallee', '33B', 33333, 'Musterdorf', NULL);
437
438     /* Die folgenden acht Tests sollten fehlschlagen:
439      * - Der erste Test enthält mehr als einen Buchstaben am Ende der Hausnummer
440      * - Der zweite Test enthält eine Zahl im Straßennamen
441      * - Der dritte Test enthält eine Zahl im Ortsnamen
442      * - Der vierte und fünfte Test enthält eine nicht-fünfstellige Postleitzahl.
443      * - Der sechste Test enthält einen leeren Straßennamen.
444      * - Der siebte Test enthält einen leeren Ortsnamen.
445      * - Der achte Test enthält eine leere Hausnummer.
446      */
447
448     SELECT 'Testing: 8 fehlschlagende Tests: Adresse';
449     INSERT INTO Adresse (Strasse, Hausnummer, PLZ, Ort, Adressen_ID)
450         VALUES ('Musterallee', '33BC', 33333, 'Musterdorf', NULL);
451     INSERT INTO Adresse (Strasse, Hausnummer, PLZ, Ort, Adressen_ID)
452         VALUES ('Mustera4llee', '33B', 33333, 'Musterdorf', NULL);
453     INSERT INTO Adresse (Strasse, Hausnummer, PLZ, Ort, Adressen_ID)
454         VALUES ('Musterallee', '33B', 33333, 'Muster4dorf', NULL);
455     INSERT INTO Adresse (Strasse, Hausnummer, PLZ, Ort, Adressen_ID)
```

```
456     VALUES ('Musterallee', '33B', 3333, 'Musterdorf', NULL);
457 INSERT INTO Adresse (Strasse, Hausnummer, PLZ, Ort, Adressen_ID)
458     VALUES ('Musterallee', '33B', 333333, 'Musterdorf', NULL);
459 INSERT INTO Adresse (Strasse, Hausnummer, PLZ, Ort, Adressen_ID)
460     VALUES ('', '33B', 33333, 'Musterdorf', NULL);
461 INSERT INTO Adresse (Strasse, Hausnummer, PLZ, Ort, Adressen_ID)
462     VALUES ('Musterallee', '33B', 333333, '', NULL);
463 INSERT INTO Adresse (Strasse, Hausnummer, PLZ, Ort, Adressen_ID)
464     VALUES ('Musterallee', '', 333333, 'Musterdorf', NULL);
465
466 ---------------------------------------------
467 ------------------ Kunde ------------------
468 ---------------------------------------------
469
470 SELECT '';
471 SELECT 'Testing: 12 funktionierende Tests: Kunde';
472 INSERT INTO Kunde (E_Mail_Adresse, Vorname, Nachname, Passwort, Adressen_ID)
473     VALUES ('helge-schneider@helge-schneider.de', 'Helge', 'Schneider', '123456',
        1);     -- Kunde, Angestellter
474 INSERT INTO Kunde (E_Mail_Adresse, Vorname, Nachname, Passwort, Adressen_ID)
475     VALUES ('peter.thoms@helge-schneider.de', 'Peter', 'Thoms', '12345678',
        2);           -- Kunde
476 INSERT INTO Kunde (E_Mail_Adresse, Vorname, Nachname, Passwort, Adressen_ID)
477     VALUES ('peter.thoms@mail.com', 'Peter', 'Thoms', '!§"$!)§$(!)§$U§!"$KASFFH',
        2);     -- Kunde, Premiumkunde
478 INSERT INTO Kunde (E_Mail_Adresse, Vorname, Nachname, Passwort, Adressen_ID)
479     VALUES ('peter.thoms@mail.org', 'Peter', 'Thoms', '12345678',
        2);                     -- Kunde
480 INSERT INTO Kunde (E_Mail_Adresse, Vorname, Nachname, Passwort, Adressen_ID)
481     VALUES ('pete-york@helge-schneider.de', 'Pete', 'York', '123456',
        3);                 -- Kunde, Premiumkunde, Angestellter
482 INSERT INTO Kunde (E_Mail_Adresse, Vorname, Nachname, Passwort, Adressen_ID)
483     VALUES ('sergej_gleithmann@helge-schneider.de', 'Sergej', 'Gleithmann',
        '123456', 3); -- Kunde
484 INSERT INTO Kunde (E_Mail_Adresse, Vorname, Nachname, Passwort, Adressen_ID)
485     VALUES ('buddy.casino@helge-schneider.de', 'Buddy', 'Casino', '123456',
        3);           -- Kunde, Angestellter
486 INSERT INTO Kunde (E_Mail_Adresse, Vorname, Nachname, Passwort, Adressen_ID)
487     VALUES ('rainer-lipski@helge-schneider.de', 'Rainer', 'Lipski', '123456',
        2);         -- Kunde
488 INSERT INTO Kunde (E_Mail_Adresse, Vorname, Nachname, Passwort, Adressen_ID)
489     VALUES ('kaistruwe@helge-schneider.de', 'Kai', 'Struwe', '123456',
        3);               -- Kunde, Premiumkunde
490 INSERT INTO Kunde (E_Mail_Adresse, Vorname, Nachname, Passwort, Adressen_ID)
491     VALUES ('willy_ketzer@helge-schneider.de', 'Willy', 'Ketzer', '123456',
        2);           -- Kunde, Angestellter
492 INSERT INTO Kunde (E_Mail_Adresse, Vorname, Nachname, Passwort, Adressen_ID)
493     VALUES ('sandro-jampedro@helge-schneider.de', 'Sandro', 'Jampedro', '123456',
        2);     -- Kunde
494 INSERT INTO Kunde (E_Mail_Adresse, Vorname, Nachname, Passwort, Adressen_ID)
495     VALUES ('karlosboes@gmx.de', 'Karlos', 'Boes', '123456',
        3);                         -- Kunde
496
497 /* Die folgenden neun Tests sollten fehlschlagen:
498  * - Der erste Test enthält eine Zahl im Vornamen
499  * - Der zweite Test enthält eine Zahl im Nachnamen
500  * - Der dritte Test enthält eine E-Mail-Adresse ohne Top-Level-Domain
```

```
501   * - Der vierte Test enthält eine E-Mail-Adresse ohne Second-Level-Domain
502   * - Der fünfte Test enthält eine E-Mail-Adresse ohne Benutzernamen
503   * - Der sechste Test enthält ein zu kurzes Passwort.
504   * - Der siebte Test enthält eine leere E-Mail-Adresse.
505   * - Der achte Test enthält einen leeren Vornamen.
506   * - Der neunte Test enthält einen leeren Nachnamen.
507   */
508
509  SELECT 'Testing: 9 fehlschlagende Tests: Kunde';
510  INSERT INTO Kunde (E_Mail_Adresse, Vorname, Nachname, Passwort, Adressen_ID)
511      VALUES ('peter.thoms@mail.org', 'Pe3ter', 'Thoms', '123456', 3);
512  INSERT INTO Kunde (E_Mail_Adresse, Vorname, Nachname, Passwort, Adressen_ID)
513      VALUES ('peter.thoms@mail.org', 'Peter', 'Tho4ms', '123456', 3);
514  INSERT INTO Kunde (E_Mail_Adresse, Vorname, Nachname, Passwort, Adressen_ID)
515      VALUES ('peter.thoms@mail.', 'Pe3ter', 'Thoms', '123456', 3);
516  INSERT INTO Kunde (E_Mail_Adresse, Vorname, Nachname, Passwort, Adressen_ID)
517      VALUES ('peter.thoms@.org', 'Pe3ter', 'Thoms', '123456', 3);
518  INSERT INTO Kunde (E_Mail_Adresse, Vorname, Nachname, Passwort, Adressen_ID)
519      VALUES ('@mail.org', 'Pe3ter', 'Thoms', '123456', 3);
520  INSERT INTO Kunde (E_Mail_Adresse, Vorname, Nachname, Passwort, Adressen_ID)
521      VALUES ('peter.thoms@mail.org', 'Peter', 'Thoms', '12345', 3);
522  INSERT INTO Kunde (E_Mail_Adresse, Vorname, Nachname, Passwort, Adressen_ID)
523      VALUES ('', 'Peter', 'Thoms', '12345', 3);
524  INSERT INTO Kunde (E_Mail_Adresse, Vorname, Nachname, Passwort, Adressen_ID)
525      VALUES ('peter.thoms@mail.org', '', 'Thoms', '12345', 3);
526  INSERT INTO Kunde (E_Mail_Adresse, Vorname, Nachname, Passwort, Adressen_ID)
527      VALUES ('peter.thoms@mail.org', 'Peter', '', '12345', 3);
528
529  ------------------------------------------
530  --------------- Premiumkunde -------------
531  ------------------------------------------
532
533  SELECT '';
534  SELECT 'Testing: 3 funktionierende Tests: Premiumkunde';
535  INSERT INTO Premiumkunde (Ablaufdatum, Studierendenausweis, E_Mail_Adresse)
536      VALUES ('2300-11-02', NULL, 'peter.thoms@mail.com');
537  INSERT INTO Premiumkunde (Ablaufdatum, Studierendenausweis, E_Mail_Adresse)
538      VALUES ('2300-11-02',
539          readfile('201px-Helge_Schneider_auf_der_Frankfurter_Buchmesse_2015.JPG'),
540          'kaistruwe@helge-schneider.de');
541  INSERT INTO Premiumkunde(Ablaufdatum, Studierendenausweis, E_Mail_Adresse)
542      VALUES ('2300-11-02',
543          readfile('201px-Helge_Schneider_auf_der_Frankfurter_Buchmesse_2015.JPG'),
544          'pete-york@helge-schneider.de');
545
546  /* Die folgenden drei Tests sollten fehlschlagen:
547   * - Der erste Test enthält ein Ablaufdatum, welches nicht in der Zukunft liegt
548   * - Der zweite Test enthält ein Datum im falschen Format
549   * - Der dritte Test referenziert einen Kunden, den es nicht gibt
550   */
551
552  SELECT 'Testing: 3 fehlschlagende Tests: Premiumkunde';
553  INSERT INTO Premiumkunde (Ablaufdatum, Studierendenausweis, E_Mail_Adresse)
554      VALUES ('2017-11-02', NULL, 'peter.thoms@mail.org');
555  INSERT INTO Premiumkunde (Ablaufdatum, Studierendenausweis, E_Mail_Adresse)
556      VALUES ('02.11.2300', NULL, 'peter.thoms@mail.org');
557  INSERT INTO Premiumkunde (Ablaufdatum, Studierendenausweis, E_Mail_Adresse)
```

```
558      VALUES ('2300-11-02', NULL, 'pete-york@mail.com');
559
560  ---------------------------------------------
561  --------------- Angestellter ---------------
562  ---------------------------------------------
563
564  SELECT '';
565  SELECT 'Testing: 4 funktionierende Tests: Angestellter';
566  INSERT INTO Angestellter (Jobbezeichnung, Gehalt, E_Mail_Adresse)
567      VALUES ('Systemadministrator', 75001, 'pete-york@helge-schneider.de');
568  INSERT INTO Angestellter (Jobbezeichnung, Gehalt, E_Mail_Adresse)
569      VALUES ('Systemadministrator', 90000, 'helge-schneider@helge-schneider.de');
570  INSERT INTO Angestellter (Jobbezeichnung, Gehalt, E_Mail_Adresse)
571      VALUES ('Organist', 85000, 'buddy.casino@helge-schneider.de');
572  INSERT INTO Angestellter (Jobbezeichnung, Gehalt, E_Mail_Adresse)
573      VALUES ('Systemadministrator', 69000, 'willy_ketzer@helge-schneider.de');
574
575  -- Der folgende Test sollte fehlschlagen, da die Jobbezeichnung leer ist.
576  SELECT 'Testing: 1 fehlschlagender Test: Angestellter';
577  INSERT INTO Angestellter (Jobbezeichnung, Gehalt, E_Mail_Adresse)
578      VALUES ('', 69000, 'willy_ketzer@helge-schneider.de');
579
580  ---------------------------------------------
581  --------------- Lieferdienst ---------------
582  ---------------------------------------------
583
584  SELECT '';
585  SELECT 'Testing: 5 funktionierende Tests: Lieferdienst';
586  INSERT INTO Lieferdienst (Lieferdienst_Bezeichnung, Versandkosten)
587      VALUES ('Deutsche Post', 2.50);
588  INSERT INTO Lieferdienst (Lieferdienst_Bezeichnung, Versandkosten)
589      VALUES ('DHL', 3.50);
590  INSERT INTO Lieferdienst (Lieferdienst_Bezeichnung, Versandkosten)
591      VALUES ('UPS', 4.00);
592  INSERT INTO Lieferdienst (Lieferdienst_Bezeichnung, Versandkosten)
593      VALUES ('DHL mit Nachnahme', 5.50);
594  INSERT INTO Lieferdienst (Lieferdienst_Bezeichnung, Versandkosten)
595      VALUES ('DHL Gefahrengutversand', 5.90);
596
597  /* Die folgenden drei Tests sollten fehlschlagen:
598   * - Der erste Test enthält eine leere Lieferdienst-Bezeichnung.
599   * - Der zweite Test enthält negative Versandkosten.
600   * - Der dritte Test enthält nicht-numerische Versandkosten.
601   */
602
603  SELECT 'Testing: 3 fehlschlagende Tests: Lieferdienst';
604  INSERT INTO Lieferdienst (Lieferdienst_Bezeichnung, Versandkosten)
605      VALUES ('', 5.90);
606  INSERT INTO Lieferdienst (Lieferdienst_Bezeichnung, Versandkosten)
607      VALUES ('DHL Gefahrengutversand', -5.90);
608  INSERT INTO Lieferdienst (Lieferdienst_Bezeichnung, Versandkosten)
609      VALUES ('DHL Gefahrengutversand', 'aaa');
610
611  ---------------------------------------------
612  ---------------- Warenkorb ----------------
613  ---------------------------------------------
614
```

```
615  SELECT '';
616  SELECT 'Testing: 9 funktionierende Tests: Warenkorb';
617  INSERT INTO Warenkorb (Bestelldatum, Bestellstatus, Warenkorb_ID, E_Mail_Adresse,
618      Lieferdienst_Bezeichnung, Lieferdatum)
619      VALUES ('2017-12-11', 'Versendet', NULL, 'peter.thoms@mail.com',
620          'DHL', '2027-12-15');
621  INSERT INTO Warenkorb (Bestelldatum, Bestellstatus, Warenkorb_ID, E_Mail_Adresse,
622      Lieferdienst_Bezeichnung, Lieferdatum)
623      VALUES ('2017-12-13', 'Versandfertig', NULL, 'peter.thoms@mail.com',
624          'UPS', NULL);
625  INSERT INTO Warenkorb (Bestelldatum, Bestellstatus, Warenkorb_ID, E_Mail_Adresse,
626      Lieferdienst_Bezeichnung, Lieferdatum)
627      VALUES ('2017-12-09', 'Abgeschlossen', NULL, 'peter.thoms@mail.com',
628          'DHL Gefahrengutversand', '2027-12-13');
629  INSERT INTO Warenkorb (Bestelldatum, Bestellstatus, Warenkorb_ID, E_Mail_Adresse,
630      Lieferdienst_Bezeichnung, Lieferdatum)
631      VALUES ('2017-12-13', 'Versandfertig', NULL, 'kaistruwe@helge-schneider.de',
632          'DHL mit Nachnahme', NULL);
633  INSERT INTO Warenkorb (Bestelldatum, Bestellstatus, Warenkorb_ID, E_Mail_Adresse,
634      Lieferdienst_Bezeichnung, Lieferdatum)
635      VALUES ('2017-12-09', 'Abgeschlossen', NULL, 'kaistruwe@helge-schneider.de',
636          'Deutsche Post', '2027-12-13');
637  INSERT INTO Warenkorb (Bestelldatum, Bestellstatus, Warenkorb_ID, E_Mail_Adresse,
638      Lieferdienst_Bezeichnung, Lieferdatum)
639      VALUES ('2017-12-09', 'Abgeschlossen', NULL, 'pete-york@helge-schneider.de',
640          'DHL', '2027-12-13');
641  INSERT INTO Warenkorb (Bestelldatum, Bestellstatus, Warenkorb_ID, E_Mail_Adresse,
642      Lieferdienst_Bezeichnung, Lieferdatum)
643      VALUES ('2017-12-11', 'Versendet', NULL, 'rainer-lipski@helge-schneider.de',
644          'Deutsche Post', '2027-12-15');
645  INSERT INTO Warenkorb (Bestelldatum, Bestellstatus, Warenkorb_ID, E_Mail_Adresse,
646      Lieferdienst_Bezeichnung, Lieferdatum)
647      VALUES (NULL, 'Versandfertig', NULL, 'rainer-lipski@helge-schneider.de',
648          'UPS', NULL);
649  INSERT INTO Warenkorb (Bestelldatum, Bestellstatus, Warenkorb_ID, E_Mail_Adresse,
650      Lieferdienst_Bezeichnung, Lieferdatum)
651      VALUES (NULL, 'Abgeschlossen', NULL, 'sandro-jampedro@helge-schneider.de',
652          'DHL Gefahrengutversand', '2027-12-13');
653
654  /* Die folgenden sechs Tests sollten fehlschlagen:
655   * - Der erste Test enthält ein Bestelldatum im falschen Datumsformat,
656   * - Der zweite Test enthält ein Lieferdatum im falschen Datumsformat,
657   * - Der dritte Test enthält einen nicht bekannten Kunden,
658   * - Der vierte Test enthält einen nicht bekannten Lieferdienst,
659   * - Der fünfte Test enthält einen nicht bekannten Bestellstatus,
660   * - Beim sechsten Test ist das Lieferdatum nicht größer gleich dem
661   *   aktuellen Datum.
662   */
663
664  SELECT 'Testing: 6 fehlschlagende Tests: Warenkorb';
665  INSERT INTO Warenkorb (Bestelldatum, Bestellstatus, Warenkorb_ID, E_Mail_Adresse,
666      Lieferdienst_Bezeichnung, Lieferdatum)
667      VALUES ('2017-12.11', 'Versendet', NULL, 'rainer-lipski@helge-schneider.de',
668          'Deutsche Post', '2027-12-15');
669  INSERT INTO Warenkorb (Bestelldatum, Bestellstatus, Warenkorb_ID, E_Mail_Adresse,
670      Lieferdienst_Bezeichnung, Lieferdatum)
671      VALUES ('2017-12-11', 'Versendet', NULL, 'rainer-lipski@helge-schneider.de',
```

```
672             'Deutsche Post', '2027-12.15');
673     INSERT INTO Warenkorb (Bestelldatum, Bestellstatus, Warenkorb_ID, E_Mail_Adresse,
674         Lieferdienst_Bezeichnung, Lieferdatum)
675         VALUES ('2017-12-11', 'Versendet', NULL, 'rainer-lips2ki@helge-schneider.de',
676             'Deutsche Post', '2027-12-15');
677     INSERT INTO Warenkorb (Bestelldatum, Bestellstatus, Warenkorb_ID, E_Mail_Adresse,
678         Lieferdienst_Bezeichnung, Lieferdatum)
679         VALUES ('2017-12-11', 'Versendet', NULL, 'rainer-lipski@helge-schneider.de',
680             'Deuts2che Post', '2027-12-15');
681     INSERT INTO Warenkorb (Bestelldatum, Bestellstatus, Warenkorb_ID, E_Mail_Adresse,
682         Lieferdienst_Bezeichnung, Lieferdatum)
683         VALUES ('2017-12-11', 'Vers2endet', NULL, 'rainer-lipski@helge-schneider.de',
684             'Deutsche Post', '2027-12-15');
685     INSERT INTO Warenkorb (Bestelldatum, Bestellstatus, Warenkorb_ID, E_Mail_Adresse,
686         Lieferdienst_Bezeichnung, Lieferdatum)
687         VALUES (NULL, 'Abgeschlossen', NULL, 'sandro-jampedro@helge-schneider.de',
688             'DHL Gefahrengutversand', '2017-12-13');
689
690     ------------------------------------------
691     ---------------- Lieferabo ----------------
692     ------------------------------------------
693
694     SELECT '';
695     SELECT 'Testing: 6 funktionierende Tests: Lieferabo';
696     INSERT INTO Lieferabo (Intervall, Beginn, Ende, Warenkorb_ID)
697         VALUES (30, '2017-11-02', '2018-11-02', 1);
698     INSERT INTO Lieferabo (Intervall, Beginn, Ende, Warenkorb_ID)
699         VALUES (30, '2017-11-02', '2018-11-02', 2);
700     INSERT INTO Lieferabo (Intervall, Beginn, Ende, Warenkorb_ID)
701         VALUES (30, '2017-11-02', '2018-11-02', 3);
702     INSERT INTO Lieferabo (Intervall, Beginn, Ende, Warenkorb_ID)
703         VALUES (30, '2017-11-02', '2018-11-02', 4);
704     INSERT INTO Lieferabo (Intervall, Beginn, Ende, Warenkorb_ID)
705         VALUES (30, '2017-11-02', '2018-11-02', 5);
706     INSERT INTO Lieferabo (Intervall, Beginn, Ende, Warenkorb_ID)
707         VALUES (30, '2017-11-02', '2018-11-02', 6);
708
709     /* Die folgenden sechs Tests sollten fehlschlagen:
710      * - Der erste Test enthält eine nichtpositive Intervallangabe
711      * - Der zweite Test enthält eine nichtganzzahlige Intervallangabe
712      * - Der dritte Test enthält ein Beginn-Datum im falschen Datumsformat
713      * - Der vierte Test enthält ein Ende-Datum im falschen Datumsformat
714      * - Der fünfte Test enthält ein Ende-Datum vor dem Beginn-Datum
715      * - Der sechste Test enthält eine ungültige Warenkorb_ID
716      */
717
718     SELECT 'Testing: 6 fehlschlagende Tests: Lieferabo';
719     INSERT INTO Lieferabo (Intervall, Beginn, Ende, Warenkorb_ID)
720         VALUES (-30, '2017-11-02', '2018-11-02', 6);
721     INSERT INTO Lieferabo (Intervall, Beginn, Ende, Warenkorb_ID)
722         VALUES (3.0, '2017-11-02', '2018-11-02', 6);
723     INSERT INTO Lieferabo (Intervall, Beginn, Ende, Warenkorb_ID)
724         VALUES (30, '2017.11-02', '2018-11-02', 6);
725     INSERT INTO Lieferabo (Intervall, Beginn, Ende, Warenkorb_ID)
726         VALUES (30, '2017-11-02', '2018-11.02', 6);
727     INSERT INTO Lieferabo (Intervall, Beginn, Ende, Warenkorb_ID)
728         VALUES (30, '2017-11-02', '2016-11-02', 6);
```

```
729  INSERT INTO Lieferabo (Intervall, Beginn, Ende, Warenkorb_ID)
730      VALUES (30, '2017-11-02', '2018-11-02', 99999);
731
732  ------------------------------------------
733  ---------------- Newsletter ---------------
734  ------------------------------------------
735
736  SELECT '';
737  SELECT 'Testing: 2 funktionierende Tests: Newsletter';
738  /* - Der erste Test testet, ob das Datum entsprechend gesetzt wird,
739   *   wenn ein neuer Eintrag vorgenommen wurde.
740   * - Der zweite Test testet, ob das Datum entsprechend aktualisiert wird,
741   *   wenn ein Eintrag aktualisiert wurde.
742   * WICHTIG: Auf der Kommandozeile sollte anschließend einmal das aktuelle
743   *          Datum, dann ein altes Datum, und dann wieder das aktuelle
744   *          Datum ausgegeben werden, um zu zeigen, dass der erste Eintrag
745   *          mit dem aktuellen Datum erstellt wird, und der zweite Eintrag,
746   *          der mit dem alten Datum erstellt wird, bei einem Update mit
747   *          dem neuen Datum aktualisiert wird.
748   */
749
750  INSERT INTO Newsletter (Betreff, Text, Newsletter_ID, E_Mail_Adresse)
751      VALUES ('Supertolle Neuheiten', 'Das sind die supertollen Neuheiten',
752          NULL, 'pete-york@helge-schneider.de');
753  SELECT Datum FROM Newsletter WHERE Newsletter_ID == 1;
754  INSERT INTO Newsletter (Betreff, Text, Datum, Newsletter_ID, E_Mail_Adresse)
755      VALUES ('Supertolle Neuheiten', 'Das sind die supertollen Neuheiten',
756          '2017-11-02', NULL, 'pete-york@helge-schneider.de');
757  SELECT Datum FROM Newsletter WHERE Newsletter_ID == 2;
758  UPDATE Newsletter SET Betreff = 'Noch bessere Neuheiten',
759      Text = 'Das sind die noch besseren Neuheiten!'
760      WHERE Newsletter_ID == 2;
761  SELECT Datum FROM Newsletter WHERE Newsletter_ID == 2;
762
763  /* Die folgenden 4 Tests sollten fehlschlagen:
764   * - Der erste Test enthält ein fehlerhaftes Datumsformat.
765   * - Der zweite Test enthält eine E-Mail-Adresse, die nicht zu einem
766   *   Angestellten gehört.
767   * - Der dritte Test enthält einen leeren Betreff.
768   * - Der vierte Test enthält einen leeren Text.
769   */
770
771  SELECT 'Testing: 4 fehlschlagende Tests: Newsletter';
772  INSERT INTO Newsletter (Betreff, Text, Datum, Newsletter_ID, E_Mail_Adresse)
773      VALUES ('Supertolle Neuheiten', 'Das sind die supertollen Neuheiten',
774          '2017-11.02', NULL, 'pete-york@helge-schneider.de');
775  INSERT INTO Newsletter (Betreff, Text, Datum, Newsletter_ID, E_Mail_Adresse)
776      VALUES ('Supertolle Neuheiten', 'Das sind die supertollen Neuheiten',
777          '2017-11-02', NULL, 'sergej_gleithmann@helge-schneider.de');
778  INSERT INTO Newsletter (Betreff, Text, Datum, Newsletter_ID, E_Mail_Adresse)
779      VALUES ('', 'Das sind die supertollen Neuheiten',
780          '2017-11.02', NULL, 'pete-york@helge-schneider.de');
781  INSERT INTO Newsletter (Betreff, Text, Datum, Newsletter_ID, E_Mail_Adresse)
782      VALUES ('Supertolle Neuheiten', '',
783          '2017-11.02', NULL, 'pete-york@helge-schneider.de');
784
785  ------------------------------------------
```

```
786  -------------- Newsletterabo --------------
787  -------------------------------------------
788
789  SELECT '';
790  SELECT 'Testing: 2 funktionierende Tests: Newsletterabo';
791  INSERT INTO Newsletterabo (E_Mail_Adresse, Newsletter_ID)
792      VALUES ('sergej_gleithmann@helge-schneider.de', 1);
793  INSERT INTO Newsletterabo (E_Mail_Adresse, Newsletter_ID)
794      VALUES ('sergej_gleithmann@helge-schneider.de', 2);
795  INSERT INTO Newsletterabo (E_Mail_Adresse, Newsletter_ID)
796      VALUES ('karlosboes@gmx.de', 2);
797
798  -------------------------------------------
799  ----------------- Artikel -----------------
800  -------------------------------------------
801
802  SELECT '';
803  SELECT 'Testing: 11 funktionierende Tests: Artikel';
804  INSERT INTO Artikel (Bezeichnung, Beschreibung, Bild, Artikel_ID)
805      VALUES ('Artikel', 'Ein Artikel', NULL, NULL);
806  INSERT INTO Artikel (Bezeichnung, Beschreibung, Bild, Artikel_ID)
807      VALUES ('Super Artikel', 'Ein super Artikel',
808      readfile('210px-HelgeSchneider_2009_2_amk.jpg'), NULL);
809  INSERT INTO Artikel (Bezeichnung, Beschreibung, Bild, Artikel_ID)
810      VALUES ('Toller Artikel', 'Ein toller Artikel',
811      readfile('210px-HelgeSchneider_2009_2_amk.jpg'), NULL);
812  INSERT INTO Artikel (Bezeichnung, Beschreibung, Bild, Artikel_ID)
813      VALUES ('Großartiger Artikel', 'Ein großartiger Artikel',
814      readfile('210px-HelgeSchneider_2009_2_amk.jpg'), NULL);
815  INSERT INTO Artikel (Bezeichnung, Beschreibung, Bild, Artikel_ID)
816      VALUES ('Mega Artikel', 'Ein mega-Artikel', NULL, NULL);
817  INSERT INTO Artikel (Bezeichnung, Beschreibung, Bild, Artikel_ID)
818      VALUES ('Hammer Artikel', 'Ein Hammer Artikel', NULL, NULL);
819  INSERT INTO Artikel (Bezeichnung, Beschreibung, Bild, Artikel_ID)
820      VALUES ('Krasser Artikel', 'Ein krasser Artikel', NULL, NULL);
821  INSERT INTO Artikel (Bezeichnung, Beschreibung, Bild, Artikel_ID)
822      VALUES ('Heftiger Artikel', 'Ein heftiger Artikel', NULL, NULL);
823  INSERT INTO Artikel (Bezeichnung, Beschreibung, Bild, Artikel_ID)
824      VALUES ('Langweiliger Artikel', 'Ein langweiliger Artikel', NULL, NULL);
825  INSERT INTO Artikel (Bezeichnung, Beschreibung, Bild, Artikel_ID)
826      VALUES ('Banane', 'Eine Banane.', NULL, NULL);
827  INSERT INTO Artikel (Bezeichnung, Beschreibung, Bild, Artikel_ID)
828      VALUES ('Ottifant', 'Ein Ottifant.', NULL, NULL);
829
830  /* Die folgenden zwei Tests sollten fehlschlagen:
831   * - Der erste Test enthält eine leere Artikelbezeichnung.
832   * - Der zweite Test enthält eine leere Artikelbeschreibung.
833   */
834
835  SELECT 'Testing: 2 fehlschlagende Tests: Artikel';
836  INSERT INTO Artikel (Bezeichnung, Beschreibung, Bild, Artikel_ID)
837      VALUES ('', 'Ein Ottifant.', NULL, NULL);
838  INSERT INTO Artikel (Bezeichnung, Beschreibung, Bild, Artikel_ID)
839      VALUES ('Ottifant', '', NULL, NULL);
840
841  -------------------------------------------
842  ----------------- Angebot -----------------
```

```
843  ---------------------------------------------
844
845  SELECT '';
846  SELECT 'Testing: 6 funktionierende Tests: Angebot';
847  INSERT INTO Angebot (Angebots_ID, Artikel_ID, Preis)
848      VALUES (NULL, 1, 1.50);
849  INSERT INTO Angebot (Angebots_ID, Artikel_ID, Preis)
850      VALUES (NULL, 1, 2.50);
851  INSERT INTO Angebot (Angebots_ID, Artikel_ID, Preis)
852      VALUES (NULL, 2, 3.50);
853  INSERT INTO Angebot (Angebots_ID, Artikel_ID, Preis)
854      VALUES (NULL, 3, 4000);
855  INSERT INTO Angebot (Angebots_ID, Artikel_ID, Preis)
856      VALUES (NULL, 5, 16.50);
857  INSERT INTO Angebot (Angebots_ID, Artikel_ID, Preis)
858      VALUES (NULL, 5, 50);
859
860  ---------------------------------------------
861  ----------------- Anbieter ----------------
862  ---------------------------------------------
863
864  SELECT '';
865  SELECT 'Testing: 6 funktionierende Tests: Anbieter';
866  INSERT INTO Anbieter (Anbieterbezeichnung)
867      VALUES ('Helge Schneider Fanshop');
868  INSERT INTO Anbieter (Anbieterbezeichnung)
869      VALUES ('Petes tolle Sachen');
870  INSERT INTO Anbieter (Anbieterbezeichnung)
871      VALUES ('Peters Müllhalde');
872  INSERT INTO Anbieter (Anbieterbezeichnung)
873      VALUES ('Sergejs langsamer Versandhandel');
874  INSERT INTO Anbieter (Anbieterbezeichnung)
875      VALUES ('Super-beschissen-Export, Inc.');
876  INSERT INTO Anbieter (Anbieterbezeichnung)
877      VALUES ('Teurer-gehts-nicht-Express GmbH');
878
879  -- Der folgende Test sollte fehlschlagen, da der Anbieter leer ist.
880
881  SELECT 'Testing: 1 fehlschlagender Test: Anbieter';
882  INSERT INTO Anbieter (Anbieterbezeichnung)
883      VALUES ('');
884
885  ---------------------------------------------
886  ------------ Anbieter_bietet_an ------------
887  ---------------------------------------------
888
889  SELECT '';
890  SELECT 'Testing: 16 funktionierende Tests: Anbieter_bietet_an';
891  INSERT INTO Anbieter_bietet_an (Anbieterbezeichnung, Angebots_ID, Bestand)
892      VALUES ('Helge Schneider Fanshop', 1, 100);
893  INSERT INTO Anbieter_bietet_an (Anbieterbezeichnung, Angebots_ID, Bestand)
894      VALUES ('Helge Schneider Fanshop', 2, 100);
895  INSERT INTO Anbieter_bietet_an (Anbieterbezeichnung, Angebots_ID, Bestand)
896      VALUES ('Helge Schneider Fanshop', 5, 100);
897  INSERT INTO Anbieter_bietet_an (Anbieterbezeichnung, Angebots_ID, Bestand)
898      VALUES ('Helge Schneider Fanshop', 3, 100);
899  INSERT INTO Anbieter_bietet_an (Anbieterbezeichnung, Angebots_ID, Bestand)
```

```
900      VALUES ('Super-beschissen-Export, Inc.', 1, 100);
901  INSERT INTO Anbieter_bietet_an (Anbieterbezeichnung, Angebots_ID, Bestand)
902      VALUES ('Super-beschissen-Export, Inc.', 2, 100);
903  INSERT INTO Anbieter_bietet_an (Anbieterbezeichnung, Angebots_ID, Bestand)
904      VALUES ('Super-beschissen-Export, Inc.', 3, 100);
905  INSERT INTO Anbieter_bietet_an (Anbieterbezeichnung, Angebots_ID, Bestand)
906      VALUES ('Super-beschissen-Export, Inc.', 5, 100);
907  INSERT INTO Anbieter_bietet_an (Anbieterbezeichnung, Angebots_ID, Bestand)
908      VALUES ('Teurer-gehts-nicht-Express GmbH', 1, 100);
909  INSERT INTO Anbieter_bietet_an (Anbieterbezeichnung, Angebots_ID, Bestand)
910      VALUES ('Teurer-gehts-nicht-Express GmbH', 5, 100);
911  INSERT INTO Anbieter_bietet_an (Anbieterbezeichnung, Angebots_ID, Bestand)
912      VALUES ('Petes tolle Sachen', 1, 100);
913  INSERT INTO Anbieter_bietet_an (Anbieterbezeichnung, Angebots_ID, Bestand)
914      VALUES ('Petes tolle Sachen', 2, 100);
915  INSERT INTO Anbieter_bietet_an (Anbieterbezeichnung, Angebots_ID, Bestand)
916      VALUES ('Petes tolle Sachen', 3, 100);
917  INSERT INTO Anbieter_bietet_an (Anbieterbezeichnung, Angebots_ID, Bestand)
918      VALUES ('Petes tolle Sachen', 5, 100);
919  INSERT INTO Anbieter_bietet_an (Anbieterbezeichnung, Angebots_ID, Bestand)
920      VALUES ('Peters Müllhalde', 2, 100);
921  INSERT INTO Anbieter_bietet_an (Anbieterbezeichnung, Angebots_ID, Bestand)
922      VALUES ('Peters Müllhalde', 3, 100);
923
924  /* Die folgenden Tests sollten fehlschlagen:
925   * - Der erste Test lässt einen Anbieter ein Angebot anbieten, welches
926   *   er bereits anbietet.
927   * - Der zweite Test enthält einen negativen Bestand.
928   */
929
930  SELECT 'Testing: 2 fehlschlagende Tests: Anbieter_bietet_an';
931  INSERT INTO Anbieter_bietet_an (Anbieterbezeichnung, Angebots_ID, Bestand)
932      VALUES ('Peters Müllhalde', 3, 100);
933  INSERT INTO Anbieter_bietet_an (Anbieterbezeichnung, Angebots_ID, Bestand)
934      VALUES ('Peters Müllhalde', 5, -100);
935
936  ----------------------------------------------
937  ------------ Angebot_im_Warenkorb ------------
938  ----------------------------------------------
939
940  SELECT '';
941  SELECT 'Testing: 6 funktionierende Tests: Angebot_im_Warenkorb';
942  INSERT INTO Angebot_im_Warenkorb (Angebots_ID, Anbieterbezeichnung,
943      Warenkorb_ID, Anzahl)
944          VALUES (1, 'Helge Schneider Fanshop', 1, 5);
945  INSERT INTO Angebot_im_Warenkorb (Angebots_ID, Anbieterbezeichnung,
946      Warenkorb_ID, Anzahl)
947          VALUES (1, 'Super-beschissen-Export, Inc.', 1, 5);
948  INSERT INTO Angebot_im_Warenkorb (Angebots_ID, Anbieterbezeichnung,
949      Warenkorb_ID, Anzahl)
950          VALUES (1, 'Super-beschissen-Export, Inc.', 2, 5);
951  INSERT INTO Angebot_im_Warenkorb (Angebots_ID, Anbieterbezeichnung,
952      Warenkorb_ID, Anzahl)
953          VALUES (2, 'Helge Schneider Fanshop', 3, 5);
954  INSERT INTO Angebot_im_Warenkorb (Angebots_ID, Anbieterbezeichnung,
955      Warenkorb_ID, Anzahl)
956          VALUES (2, 'Super-beschissen-Export, Inc.', 3, 5);
```

```
957  INSERT INTO Angebot_im_Warenkorb (Angebots_ID, Anbieterbezeichnung,
958      Warenkorb_ID, Anzahl)
959          VALUES (3, 'Super-beschissen-Export, Inc.', 5, 5);
960
961  /* Die folgenden vier Tests sollten fehlschlagen:
962   * - Der erste Test enthält eine negative Anzahl
963   * - Der zweite Test versucht einem Warenkorb Angebote eines Anbieters
964   *   hinzuzufügen, der bereits dasselbe Angebot dieses Anbieters enthält.
965   * - Der dritte Test versucht, ein Angebot in den Warenkorb zu legen,
966   *   welches von keinem Anbieter angeboten wird.
967   * - Der vierte Test versucht, ein Angebot in einer höheren Stückzahl
968   *   in den Warenkorb zu legen, als der Anbieter es anbietet.
969   */
970
971  SELECT 'Testing: 4 fehlschlagende Tests: Angebot_im_Warenkorb';
972  INSERT INTO Angebot_im_Warenkorb (Angebots_ID, Anbieterbezeichnung,
973      Warenkorb_ID, Anzahl)
974          VALUES (1, 'Super-beschissen-Export, Inc.', 5, -5);
975  INSERT INTO Angebot_im_Warenkorb (Angebots_ID, Anbieterbezeichnung,
976      Warenkorb_ID, Anzahl)
977          VALUES (3, 'Super-beschissen-Export, Inc.', 5, 5);
978  INSERT INTO Angebot_im_Warenkorb (Angebots_ID, Anbieterbezeichnung,
979      Warenkorb_ID, Anzahl)
980          VALUES (4, 'Helge Schneider Fanshop', 5, 5);
981  INSERT INTO Angebot_im_Warenkorb (Angebots_ID, Anbieterbezeichnung,
982      Warenkorb_ID, Anzahl)
983          VALUES (1, 'Helge Schneider Fanshop', 5, 900);
984
985  ---------------------------------------------
986  ----------- Artikel_im_Newsletter -----------
987  ---------------------------------------------
988
989  SELECT '';
990  SELECT 'Testing: 8 funktionierende Tests: Artikel_im_Newsletter';
991  INSERT INTO Artikel_im_Newsletter (Artikel_ID, Newsletter_ID) VALUES (1, 1);
992  INSERT INTO Artikel_im_Newsletter (Artikel_ID, Newsletter_ID) VALUES (1, 2);
993  INSERT INTO Artikel_im_Newsletter (Artikel_ID, Newsletter_ID) VALUES (2, 1);
994  INSERT INTO Artikel_im_Newsletter (Artikel_ID, Newsletter_ID) VALUES (2, 2);
995  INSERT INTO Artikel_im_Newsletter (Artikel_ID, Newsletter_ID) VALUES (3, 1);
996  INSERT INTO Artikel_im_Newsletter (Artikel_ID, Newsletter_ID) VALUES (3, 2);
997  INSERT INTO Artikel_im_Newsletter (Artikel_ID, Newsletter_ID) VALUES (4, 1);
998  INSERT INTO Artikel_im_Newsletter (Artikel_ID, Newsletter_ID) VALUES (5, 2);
999
1000 /* Der folgende Test sollte fehlschlagen, da mehr als 10 Artikel im
1001  * selben Newsletter referenziert werden sollen.*/
1002 SELECT 'Testing: 1 fehlschlagender Test: Artikel_im_Newsletter';
1003 INSERT INTO Artikel_im_Newsletter (Artikel_ID, Newsletter_ID) VALUES (5, 1);
1004 INSERT INTO Artikel_im_Newsletter (Artikel_ID, Newsletter_ID) VALUES (6, 1);
1005 INSERT INTO Artikel_im_Newsletter (Artikel_ID, Newsletter_ID) VALUES (7, 1);
1006 INSERT INTO Artikel_im_Newsletter (Artikel_ID, Newsletter_ID) VALUES (8, 1);
1007 INSERT INTO Artikel_im_Newsletter (Artikel_ID, Newsletter_ID) VALUES (9, 1);
1008 INSERT INTO Artikel_im_Newsletter (Artikel_ID, Newsletter_ID) VALUES (10, 1);
1009 INSERT INTO Artikel_im_Newsletter (Artikel_ID, Newsletter_ID) VALUES (11, 1);
1010
1011 -----------------------------------------------
1012 --------- Artikel_empfiehlt_Artikel ---------
1013 -----------------------------------------------
```

```
1014
1015  SELECT '';
1016  SELECT 'Testing: 5 funktionierende Tests: Artikel_empfiehlt_Artikel';
1017  INSERT INTO Artikel_empfiehlt_Artikel (Artikel_ID1, Artikel_ID2) VALUES (1, 2);
1018  INSERT INTO Artikel_empfiehlt_Artikel (Artikel_ID1, Artikel_ID2) VALUES (2, 1);
1019  INSERT INTO Artikel_empfiehlt_Artikel (Artikel_ID1, Artikel_ID2) VALUES (3, 2);
1020  INSERT INTO Artikel_empfiehlt_Artikel (Artikel_ID1, Artikel_ID2) VALUES (4, 2);
1021  INSERT INTO Artikel_empfiehlt_Artikel (Artikel_ID1, Artikel_ID2) VALUES (5, 1);
1022
1023  ------------------------------------------
1024  --------------- Schlagwort ---------------
1025  ------------------------------------------
1026
1027  SELECT '';
1028  SELECT 'Testing: 5 funktionierende Tests: Schlagwort';
1029  INSERT INTO Schlagwort (Schlagwort) VALUES ('Werkzeug');
1030  INSERT INTO Schlagwort (Schlagwort) VALUES ('Malerzubehör');
1031  INSERT INTO Schlagwort (Schlagwort) VALUES ('Unnützes Zeug');
1032  INSERT INTO Schlagwort (Schlagwort) VALUES ('Müll');
1033  INSERT INTO Schlagwort (Schlagwort) VALUES ('Obst');
1034
1035  /* Die folgenden zwei Tests sollten fehlschlagen:
1036   * - Der erste Test versucht, ein neues Schlagwort anzulegen, obwohl es
1037   *   bereits dasselbe Schlagwort mit einer anderen Groß- und Kleinschreibung
1038   *   gibt.
1039   * - Der zweite Test hat ein leeres Schlagwort.
1040   */
1041
1042  SELECT '';
1043  SELECT 'Testing: 2 fehlschlagender Test: Schlagwort';
1044  INSERT INTO Schlagwort (Schlagwort) VALUES ('unnützes Zeug');
1045  INSERT INTO Schlagwort (Schlagwort) VALUES ('');
1046
1047  ------------------------------------------
1048  ------- Artikel_gehoert_zu_Schlagwort -------
1049  ------------------------------------------
1050
1051  SELECT '';
1052  SELECT 'Testing: 12 funktionierende Tests: Artikel_gehoert_zu_Schlagwort';
1053  INSERT INTO Artikel_gehoert_zu_Schlagwort (Artikel_ID, Schlagwort)
1054      VALUES (1, 'Werkzeug');
1055  INSERT INTO Artikel_gehoert_zu_Schlagwort (Artikel_ID, Schlagwort)
1056      VALUES (1, 'Malerzubehör');
1057  INSERT INTO Artikel_gehoert_zu_Schlagwort (Artikel_ID, Schlagwort)
1058      VALUES (1, 'Unnützes Zeug');
1059  INSERT INTO Artikel_gehoert_zu_Schlagwort (Artikel_ID, Schlagwort)
1060      VALUES (2, 'Werkzeug');
1061  INSERT INTO Artikel_gehoert_zu_Schlagwort (Artikel_ID, Schlagwort)
1062      VALUES (2, 'Obst');
1063  INSERT INTO Artikel_gehoert_zu_Schlagwort (Artikel_ID, Schlagwort)
1064      VALUES (3, 'Werkzeug');
1065  INSERT INTO Artikel_gehoert_zu_Schlagwort (Artikel_ID, Schlagwort)
1066      VALUES (3, 'Malerzubehör');
1067  INSERT INTO Artikel_gehoert_zu_Schlagwort (Artikel_ID, Schlagwort)
1068      VALUES (4, 'Werkzeug');
1069  INSERT INTO Artikel_gehoert_zu_Schlagwort (Artikel_ID, Schlagwort)
1070      VALUES (4, 'Malerzubehör');
```

```
1071  INSERT INTO Artikel_gehoert_zu_Schlagwort (Artikel_ID, Schlagwort)
1072      VALUES (4, 'Müll');
1073  INSERT INTO Artikel_gehoert_zu_Schlagwort (Artikel_ID, Schlagwort)
1074      VALUES (4, 'Obst');
1075  INSERT INTO Artikel_gehoert_zu_Schlagwort (Artikel_ID, Schlagwort)
1076      VALUES (5, 'Werkzeug');
1077  INSERT INTO Artikel_gehoert_zu_Schlagwort (Artikel_ID, Schlagwort)
1078      VALUES (5, 'Malerzubehör');
1079
1080  /* Der folgende Test sollte fehlschlagen, da er versucht, einen Artikel
1081   * zu einem Schlagwort hinzuzufügen, das dieser Artikel bereits hat.
1082   */
1083
1084  SELECT 'Testing: 1 fehlschlagender Test: Artikel_gehoert_zu_Schlagwort';
1085  INSERT INTO Artikel_gehoert_zu_Schlagwort (Artikel_ID, Schlagwort)
1086      VALUES (5, 'Malerzubehör');
1087
1088  /*=========================================
1089   *============= TEST-ANFRAGEN ==============
1090   *=========================================*/
1091
1092  SELECT '=========================================';
1093  SELECT '============= TEST-ANFRAGEN ==============';
1094  SELECT '=========================================';
1095  SELECT '';
1096
1097  SELECT '';
1098  SELECT 'Lieferdienste, deren Versandkosten zwischen 5 und 6 Euro liegen:';
1099  SELECT * FROM Lieferdienst
1100  WHERE Versandkosten > 5 AND Versandkosten < 6;
1101
1102  SELECT '';
1103  SELECT 'Angestellte mit Jobbezeichnung "Sytemadministrator" und Jahres-';
1104  SELECT '  gehalt über 75000 Euro:';
1105  SELECT * FROM Angestellter
1106  WHERE Jobbezeichnung = 'Systemadministrator' AND Gehalt > 75000;
1107
1108  SELECT '';
1109  SELECT 'Premiumkunden mit der höchsten Anzahl an Lieferabos:';
1110  SELECT MAX(Anzahl), E_Mail_Adresse FROM (
1111      SELECT COUNT(*) AS Anzahl, Premiumkunde.E_Mail_Adresse FROM Lieferabo
1112      JOIN Warenkorb on Lieferabo.Warenkorb_ID == Warenkorb.Warenkorb_ID
1113      JOIN Premiumkunde on Premiumkunde.E_Mail_Adresse == Warenkorb.E_Mail_Adresse
1114      GROUP BY Premiumkunde.E_Mail_Adresse
1115  );
1116
1117  SELECT '';
1118  SELECT 'Artikel aus mindestens zwei Newslettern mit Kategorie "Werkzeug"';
1119  SELECT '  und Kategorie "Malerzubehör":';
1120
1121  SELECT AnzahlInNewslettern, NewsletterArtikelID, Schlagwort1, Schlagwort2
1122  FROM (
1123      SELECT * FROM (
1124          SELECT COUNT(*) AS AnzahlInNewslettern,
1125          Artikel_im_Newsletter.Artikel_ID AS NewsletterArtikelID
1126          FROM Artikel_im_Newsletter
1127          GROUP BY Artikel_im_Newsletter.Artikel_ID
```

```
1128            HAVING COUNT(*) >= 2
1129        ) JOIN (
1130            SELECT Artikel_ID AS SchlagwortArtikelID1, Schlagwort AS Schlagwort1
1131            FROM Artikel_gehoert_zu_Schlagwort
1132            WHERE Schlagwort = 'Malerzubehör'
1133        ) ON SchlagwortArtikelID1 == NewsletterArtikelID
1134        JOIN (
1135            SELECT Artikel_ID AS SchlagwortArtikelID2, Schlagwort AS Schlagwort2
1136            FROM Artikel_gehoert_zu_Schlagwort
1137            WHERE Schlagwort = 'Werkzeug'
1138        )
1139        ON SchlagwortArtikelID2 == NewsletterArtikelID
1140    )
1141    ;--ON SchlagwortArtiklelID;
1142
1143    SELECT '';
1144    SELECT 'Artikel, zu denen ein Bild vorliegt, aber kein Angebot:';
1145    SELECT * FROM Artikel
1146    LEFT JOIN Angebot on Angebot.Artikel_ID == Artikel.Artikel_ID
1147    WHERE Artikel.Bild IS NOT NULL AND Angebot.Angebots_ID IS NULL;
1148
```