```
/* ************************************************************************
PL/SQL mit dem SQL-Developer entwickeln –
Wie der Einstieg und die Fehlersuche gelingt

Autor: Jens Lambert, Hochschule Düsseldorf
Version: 3, im Juli 2022
Lizenz:  CC BY-SA 4.0
Lizenzbedingungen: https://creativecommons.org/licenses/by-sa/4.0/deed.de

Der SQL-Developer unterliegt den Lizenzbedingungen der Oracle
Corporation https://www.oracle.com, Stand Juli 2022.
************************************************************************ */

-- Der Möbelkatalog
-- Eingaben auch in Skriptausgabe anzeigen
SET ECHO ON
-- PL/SQL Server Meldungen in Skriptausgabe anzeigen
SET SERVEROUTPUT ON

-- SQL Formatierung, optional
SET LINESIZE 95
SET PAGESIZE 50000
SET SQLFORMAT ANSICONSOLE

-- Alte Tabellen entfernen
DROP FUNCTION verkaufspreis;
DROP PROCEDURE verkaufen;
DROP TABLE verkaeufe;
DROP SEQUENCE belegnummer_seq;
DROP TABLE artikel;

-- ########################################################################
-- PL/SQL ausführen

EXECUTE dbms_output.put_line('Hallo Welt!');

-- PL/SQL Server Meldungen in Skriptausgabe anzeigen
SET SERVEROUTPUT ON

BEGIN
   dbms_output.put('Hallo');
   dbms_output.put(' Welt');
   dbms_output.put_line('!');
   dbms_output.put_line('-----------');
END;

-- ########################################################################
-- Tabellen für das Möbelkaufhaus vorbereiten

-- Die Artikel werden in einem Katalog gespeichert
CREATE TABLE artikel (
   artikelnummer NUMBER(8)
      CONSTRAINT artikel_pk PRIMARY KEY,
   bezeichnung VARCHAR2(25) NOT NULL,
   nettoeinkaufspreis NUMBER(8,2) NOT NULL,
   nettoverkaufspreis NUMBER(8,2) NOT NULL
);

-- Es darf nicht unter Einkaufspreis verkauft werden.
ALTER TABLE artikel
   ADD CONSTRAINT artikel_nettoverkaufspreis_ck
      CHECK (nettoverkaufspreis > nettoeinkaufspreis);


-- Der Artikel Katalog wird gefüllt
INSERT INTO artikel VALUES(1, 'Sofa - William', 420.00, 835.45);
INSERT INTO artikel VALUES(2, 'Sessel - Mia', 140.00, 275.45);
INSERT INTO artikel VALUES(3, 'Tisch - Isabella', 210.00, 415.45);
INSERT INTO artikel VALUES(4, 'Stuhl - Lucas', 65.00, 125.45);
INSERT INTO artikel VALUES(5, 'Bett - Finn', 310.00, 615.45);
INSERT INTO artikel VALUES(6, 'Design Liege - James', 460.00, 1200.45);
COMMIT;

```

```sql
-- Ausgabe aller gespeicherten Artikel
SELECT *
FROM artikel;


-- Automatische Erzeugung von eindeutigen Belegnummern für Verkäufe
CREATE SEQUENCE belegnummer_seq
   START WITH     1001
   INCREMENT BY   1;


-- Jeder Verkauf wird dokumentiert
CREATE TABLE verkaeufe (
   belegnummer NUMBER(8)
      CONSTRAINT verkaeufe_pk PRIMARY KEY,
   artikelnummer NUMBER(8)
      CONSTRAINT verkaeufe_artikel_fk REFERENCES artikel (artikelnummer),
   anzahl NUMBER(8) NOT NULL,
   verkaufspreis NUMBER(8,2) NOT NULL
);

-- ###########################################################################
-- anonymer Block MIT FEHLERN

-- Ziel: Ausgabe des Verkaufspreises:
DECLARE
    l_nettoverkaufspreis NUMBER(8,2):= 0;
    l_rabatt NUMBER(8,2) := ((100 - 10)/100); -- hier: -10%
    co_mwst CONSTANT NUMBER(8,2) := 1.19;
BEGIN

    -- Verkaufspreis netto aus Artikel-Tabelle lesen
    SELECT nettoverkaufspreis
    INTO nettoverkaufspreis
    FROM artikel
    WHERE artikel.artikelnummer = 100;

   -- berechnen als Produkt aus Preis, Anzahl und MwSt.
  dbms_output.put_line('Verkaufspreis: '|| ROUND(l_nettoverkaufspreis * l_rabatt *
  co_mwst, 2) ||' €');

END;


-- Fehler:
-- 1) falscher Variablen-Bezeichner INTO -> l_nettoverkaufspreis
-- 2) WHERE falsche Aritkelnummer -> 1

-- Fehler korrigiert:
DECLARE
    l_nettoverkaufspreis NUMBER(8,2):= 0;
    l_rabatt NUMBER(8,2) := ((100 - 10)/100); -- hier: -10%
    co_mwst CONSTANT NUMBER(8,2) := 1.19;
BEGIN

    -- Verkaufspreis netto aus Artikel-Tabelle lesen
    SELECT nettoverkaufspreis
    INTO l_nettoverkaufspreis
    FROM artikel
    WHERE artikel.artikelnummer = 1;

   -- berechnen als Produkt aus Preis, Anzahl und MwSt.
    dbms_output.put_line('Verkaufspreis: '|| ROUND(l_nettoverkaufspreis * l_rabatt *
    co_mwst, 2) );

END;

-- ###########################################################################
-- FUNKTION MIT FEHLERN
/**
Berechnung des Verkaufspreises
@param in_artikelnummer Artikelnummer des verkauften Artikels
@param in_rabatt Rabatt des verkauften Artikels (0-100%)
```

```
145  @return aktueller Verkaufspreis des Artikels mit MwSt. und Rabatt
146  */
147  CREATE OR REPLACE FUNCTION verkaufspreis(in_artikelnummer IN NUMBER,
148     in_rabatt IN NUMBER)
149      l_nettoverkaufspreis NUMBER(8,2):= 0;
150      l_rabatt NUMBER(8,2) := ((100 - in_rabatt)/100);
151      co_mwst CONSTANT NUMBER(8,2) := 1.19;
152  BEGIN
153
154      -- Verkaufspreis netto aus Artikel-Tabelle lesen
155      SELECT nettoverkaufspreis
156      INTO l_nettoverkaufspreis
157      FROM artikel
158      WHERE artikel.artikelnummer = in_artikelnummer;
159
160     -- berechnen als Produkt aus Preis, Anzahl und MwSt.
161     dbms_output.put_line('Verkaufspreis: '|| ROUND(l_nettoverkaufspreis * l_rabatt *
        co_mwst, 2) );
162
163      -- EXCEPTION
164      -- hier können Ausnahmen behandelt werden
165
166  END verkaufspreis;
167
168  -- SHOW ERRORS
169
170  -- Fehler:
171  -- 1) Rückgabewert mit richtigem Datentyp fehlt:
172  -- Für Debug kompilieren, Zeile ansehen, im Skript korrigieren, erneut kompilieren
173  -- 2) Ausgabe zu RETURN abändern.
174
175  -- ######################################################################
176  -- Fehler korrigiert:
177  /**
178  Berechnung des Verkaufspreises
179  @param in_artikelnummer Artikelnummer des verkauften Artikels
180  @param in_rabatt Rabatt des verkauften Artikels (0-100%)
181  @return aktueller Verkaufspreis des Artikels mit MwSt. und Rabatt
182  */
183  CREATE OR REPLACE FUNCTION verkaufspreis(in_artikelnummer IN NUMBER, in_rabatt IN
     NUMBER)
184  RETURN NUMBER AS
185     l_nettoverkaufspreis NUMBER(8,2):= 0;
186     l_rabatt NUMBER(8,2) := (100 - in_rabatt)/100;
187     co_mwst CONSTANT NUMBER(8,2) := 1.19;
188  BEGIN
189
190     -- Verkaufspreis netto aus Artikel-Tabelle lesen
191     SELECT nettoverkaufspreis
192     INTO l_nettoverkaufspreis
193     FROM artikel
194     WHERE artikel.artikelnummer = in_artikelnummer;
195
196     -- berechnen als Produkt aus Preis, Anzahl und MwSt.
197     RETURN ROUND(l_nettoverkaufspreis * l_rabatt * co_mwst , 2);
198
199      -- EXCEPTION
200      -- hier können Ausnahmen behandelt werden
201
202  END verkaufspreis;
203
204  -- ######################################################################
205
206  -- Test der Funktion mit anonymen Block
207  BEGIN
208     DBMS_OUTPUT.PUT_LINE('Verkauft für: '|| verkaufspreis(1,10)||' € inkl. 19% MwSt.');
209  END;
210
211
212  -- Test der Funktion als Query
213  SELECT verkaufspreis(1,10)||' € inkl. 19% MwSt.' AS "Verkauft für"
214  FROM DUAL;
215
```

```
-- ########################################################################

/**
Prozedur zum persistieren von Verkäufen.
@param in_bezeichnung Bezeichnung des verkauften Artikels
@param in_anzahl Anzahl des verkauften Artikels
*/
CREATE OR REPLACE PROCEDURE verkaufen(in_bezeichnung IN VARCHAR2, in_anzahl IN NUMBER
, in_rabatt IN NUMBER) AS
   l_belegnummer NUMBER(8) := belegnummer_seq.nextval;
   l_artikelnummer NUMBER;
BEGIN
   SELECT artikelnummer INTO l_artikelnummer
   FROM artikel
   WHERE bezeichnung = in_bezeichnung;

   -- Verkauf in der Relation eintragen
   INSERT INTO verkaeufe VALUES (l_belegnummer, l_artikelnummer, in_anzahl,
      verkaufspreis(l_artikelnummer, in_rabatt) * in_anzahl);

    -- EXCEPTION
    -- hier können Ausnahmen behandelt werden

END verkaufen;


-- Nun wird etwas verkauft....
EXECUTE verkaufen('Bett - Finn', 1,0);
EXECUTE verkaufen('Sessel - Mia', 2,0);
EXECUTE verkaufen('Sofa - William', 1,0);
EXECUTE verkaufen('Tisch - Isabella', 1,0);
EXECUTE verkaufen('Stuhl - Lucas', 4,10);

COMMIT;

SELECT belegnummer AS "Beleg",
   artikelnummer AS "Art. Nr.",
   bezeichnung AS "Bezeichnung",
   TO_CHAR(nettoeinkaufspreis,'L99G999D99MI') AS "Einkauf",
   TO_CHAR(nettoverkaufspreis,'L99G999D99MI') AS "Verkauf",
   anzahl AS "Anzahl",
   TO_CHAR(verkaufspreis, 'L99G999D99MI') AS "Verkauf Summe"
FROM artikel NATURAL JOIN verkaeufe
ORDER BY belegnummer;
```