

## Funktionen mit komplexeren Schnittstellen

In den vorhergehenden Beispielen waren Parameter bzw. Rückgabewerte jeweils nur ein Wert (eines einfachen Datentyps wie `INTEGER`, `NUMERIC`, `TEXT`, etc.). Die Schnittstellen können aber auch komplexere Strukturen aufweisen:

- eine Zeile einer Tabelle (mit mehreren Spalten)
- eine Spalte einer Tabelle (mit mehreren Zeilen)
- eine Tabelle (mit mehreren Zeilen und mehreren Spalten)

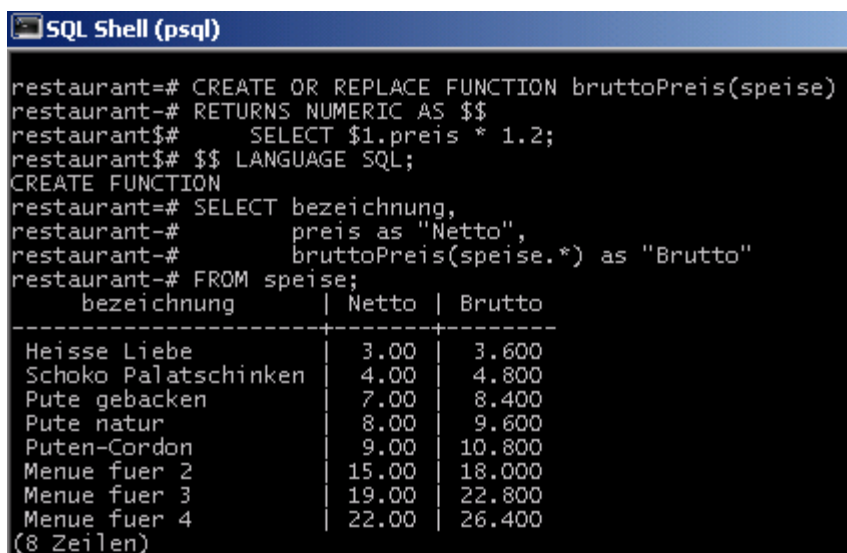
### Funktionen mit einer Tabellenzeile als Parameter

#### Beispiel:

Zu allen Speisen soll der Brutto-Preis berechnet und als `brutto` angezeigt werden:

```
CREATE OR REPLACE FUNCTION bruttoPreis(speise)
RETURNS NUMERIC AS $$
    SELECT $1.preis * 1.2;
$$ LANGUAGE SQL;
```

- Als Funktionsparameter wird der Tabellenname `speise` definiert, d.h. beim Aufruf muss jeweils ein Datensatz vom Typ `speise` übergeben werden.
- Im Funktionskörper kann man sich auf die Spalte `preis` des Datensatzes beziehen, indem man den Parameter mit dem Spaltennamen qualifiziert: `$1.preis`.
- Beim Funktionsaufruf werden mittels `speise.*` alle Spalten des Datensatzes übergeben. Die Funktion wird für jeden Datensatz, welcher durch den äußeren `SELECT` gelesen wird, einzeln aufgerufen.



```
SQL Shell (psql)
restaurant=# CREATE OR REPLACE FUNCTION bruttoPreis(speise)
restaurant=# RETURNS NUMERIC AS $$
restaurant$#     SELECT $1.preis * 1.2;
restaurant$# $$ LANGUAGE SQL;
CREATE FUNCTION
restaurant=# SELECT bezeichnung,
restaurant=#           preis as "Netto",
restaurant=#           bruttoPreis(speise.*) as "Brutto"
restaurant=# FROM speise;
 bezeichnung | Netto | Brutto
-----+-----+-----
 Heisse Liebe | 3.00  | 3.600
 Schoko Palatschinken | 4.00  | 4.800
 Pute gebacken | 7.00  | 8.400
 Pute natur | 8.00  | 9.600
 Puten-Cordon | 9.00  | 10.800
 Menue fuer 2 | 15.00 | 18.000
 Menue fuer 3 | 19.00 | 22.800
 Menue fuer 4 | 22.00 | 26.400
(8 Zeilen)
```

#### Übung:

Erstelle eine weitere Funktion zur Berechnung der MWSt. Zeige von allen Speisen den Brutto-Preis als `Brutto` und die darin enthaltene MWSt. als Spalte `MWSt` an.

Zusatz: Die Anzeige bei Brutto/MWSt soll auf zwei Nachkommastellen beschränkt werden.

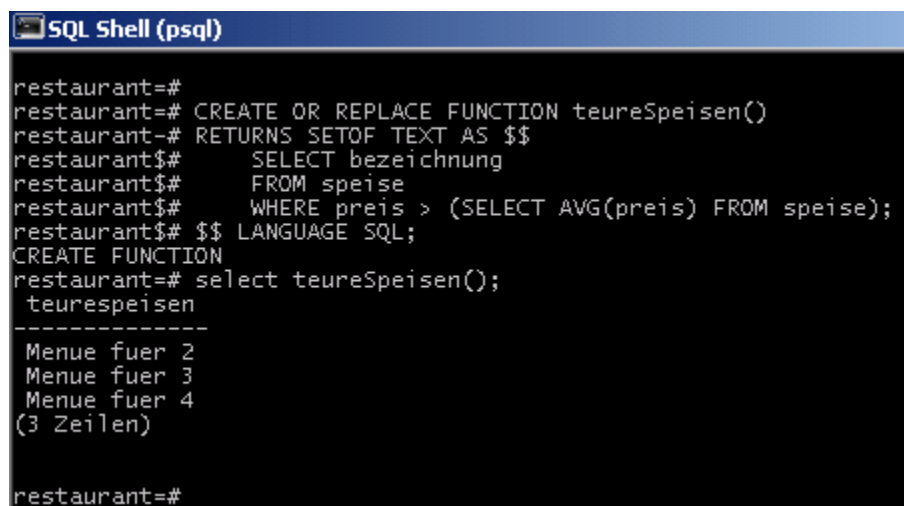
## Funktionen mit einer Tabellenspalte als Rückgabewert

### Beispiel:

Es sollen die Speisebezeichnungen jener Speisen, die teurer als der Durchschnittspreis sind, als Menge an den Aufrufer der Funktion zurückgeliefert werden.

```
CREATE OR REPLACE FUNCTION teureSpeisen()  
RETURNS SETOF TEXT AS $$  
    SELECT bezeichnung  
    FROM speise  
    WHERE preis > (SELECT AVG(preis) FROM speise);  
$$ LANGUAGE SQL;
```

- Durch die Angabe `SETOF` in Verbindung mit einem Datentyp wird festgelegt, dass die Funktion eine Menge (von Informationen desselben Datentyps) zurückliefert.



```
SQL Shell (psql)  
restaurant=#  
restaurant=# CREATE OR REPLACE FUNCTION teureSpeisen()  
restaurant=# RETURNS SETOF TEXT AS $$  
restaurant$#     SELECT bezeichnung  
restaurant$#     FROM speise  
restaurant$#     WHERE preis > (SELECT AVG(preis) FROM speise);  
restaurant$# $$ LANGUAGE SQL;  
CREATE FUNCTION  
restaurant=# select teureSpeisen();  
teurespeisen  
-----  
Menue fuer 2  
Menue fuer 3  
Menue fuer 4  
(3 Zeilen)  
restaurant=#
```

### Übung:

Erstelle eine Funktion zur Anzeige der Bezeichnungen der noch nie bestellten Speisen.

Zusatz: Erweitere die aufrufende `SELECT`-Anweisung so, dass das Ergebnis als Tabelle mit den Spaltenüberschriften "Bezeichnung" und "Nettopreis" angezeigt wird.

## Funktionen mit Tabellen als Rückgabewert

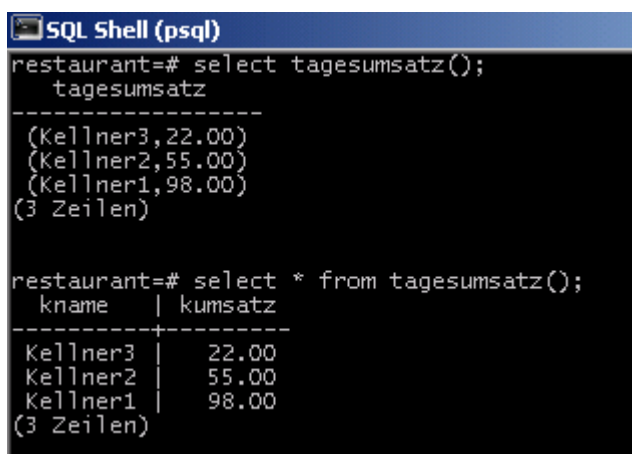
```
CREATE TABLE tablename (
    spalte1    datentyp1,
    spalte2    datentyp2
);

CREATE OR REPLACE FUNCTION funktionsname()
RETURNS SETOF tablename AS $$
    SELECT spalteX AS "spalte1", spalteY AS "spalte2"
    FROM ...
    WHERE ...;
$$ LANGUAGE SQL;

SELECT * FROM funktionsname();

DROP FUNCTION funktionsname();    // sonst Abhängigkeit ...
DROP TABLE tablename;
```

- Wenn der "Rückgabewert" einer Funktion mehrere Datensätze mit jeweils mehreren Spalten umfassen soll, muss die Struktur dieser Datensätze durch eine Tabellendefinition beschrieben werden.
- Der Name der Tabelle wird durch die Angabe `SETOF` definiert.
- Die Struktur dieser Tabelle muss vor der Funktionsdefinition mittels `CREATE TABLE` definiert worden sein.
- Diese Tabelle muss keine PK oder FK-Definitionen enthalten.
- Es können aber auch bereits in der DB bestehende Tabellen-Strukturen genutzt werden, unabhängig davon, ob sie Daten enthalten. Diese Daten werden beim Funktionsaufruf nicht überschrieben oder verändert!
- Das Ergebnis des Funktionsaufrufes kann wie eine "normale" Tabelle in der `FROM`-Klausel der äußeren `SELECT`-Anweisung eingesetzt werden.
- Die äußere `SELECT`-Anweisung zeigt die Tabellenstruktur nur dann, wenn eine `FROM`-Klausel angegeben wird. Ohne `FROM`-Klausel erfolgt die Darstellung als Menge:



```
SQL Shell (psql)
restaurant=# select tagesumsatz();
 tagesumsatz
-----
(Kellner3,22.00)
(Kellner2,55.00)
(Kellner1,98.00)
(3 Zeilen)

restaurant=# select * from tagesumsatz();
 kname | kumsatz
-----+-----
 Kellner3 | 22.00
 Kellner2 | 55.00
 Kellner1 | 98.00
(3 Zeilen)
```

### Übung:

Es soll eine Liste der Kellner und deren jeweiliger Tagesumsatz ausgegeben werden.  
Korrekte Lösung mit der Original-DB: nur eine Zeile mit `Kellner1` und `71.00`