Die folgenden Übungen sollten Sie mit der MySQL-Datenbank *SQL_Demo* durchführen. Diese Datenbank haben Sie bereits übernommen oder sie liegt auf dem BSCW.

Führen Sie folgende Statements aus (ACHTUNG: kein Leerschritt zwischen COUNT und der Klammer) und überlegen Sie sich, welche Frage sie beantwortet:

1. SELECT COUNT (Preis) FROM tbl_artikel
   Suchen Sie den Unterschied zwischen COUNT (*) und COUNT(*attribut*)!
   Mit Stern werden alle Datensätze gezählt bei Attributangabe nur die NOT NULL-Werte _____

2. SELECT name, COUNT (vorname) FROM tbl_personen group by name
   zählt die Anzahl Personen mit gleichem Nachnamen _____

3. SELECT FS_Rechnung, COUNT (FS_Artikel) FROM tbl_trans GROUP BY FS_Rechnung
   wie viele Positionen eine Rechnung enthält; verschiedene Artikel – nicht Menge! _____

Die folgenden Statements führen Sie mit der gleichen Datenbank schrittweise an die Verwendung der Joins heran – hoffe ich ☺

SELECT * FROM tbl_personen, tbl_rechnungen
Beispiel für einen Full-Join (kartesisches Produkt)

Eigentlich sinnlos, da es einfach alle Datensätze kombiniert

```
SELECT * FROM tbl_personen, tbl_rechnungen
WHERE tbl_personen.Person_ID = tbl_rechnungen.Person_FS
```

Schon besser, da es die Rechnungen mit den zugehörigen Personen anzeigt

```
SELECT * FROM tbl_personen INNER JOIN tbl_rechnungen
  ON tbl_personen.Person_ID = tbl_rechnungen.Person_FS
```

Macht das gleiche, wie das vorherige statement, nur mit JOIN (normalerweise effizienter)

```
SELECT * FROM tbl_personen LEFT JOIN tbl_rechnungen
  ON tbl_personen.Person_ID = tbl_rechnungen.Person_FS
```

Gibt alle Personen aus und wenn eine Rechnung dazugehört, die Rechnungsdaten, sonst NULL

```
SELECT * FROM tbl_personen RIGHT JOIN tbl_rechnungen
  ON tbl_personen.Person_ID = tbl_rechnungen.Person_FS
```

Gibt alle Rechnungen aus und wenn eine Person dazugehört, die Personendaten, sonst NULL

```
SELECT * FROM tbl_personen AS p INNER JOIN tbl_rechnungen AS r
  ON p.Person_ID = r.Person_FS
  INNER JOIN tbl_trans AS t ON r.Rechnungs_ID = t.FS_Rechnung
ORDER BY Person_ID
```

Zeigt nur die Rechnungen, die eine Position/einen Artikel enthalten

```
SELECT * FROM tbl_personen AS p INNER JOIN tbl_rechnungen AS r
  ON p.Person_ID = r.Person_FS
  INNER JOIN tbl_trans AS t ON r.Rechnungs_ID = t.FS_Rechnung
    INNER JOIN tbl_artikel AS a ON t.FS_Artikel = a.Artikel_ID
ORDER BY Person_ID
```

Zeigt nur die Rechnungen, die eine Position/einen Artikel enthalten + der Artikeldaten

```
SELECT * FROM tbl_personen AS p INNER JOIN tbl_rechnungen AS r
  ON p.Person_ID = r.Person_FS
  INNER JOIN tbl_trans AS t ON r.Rechnungs_ID = t.FS_Rechnung
    LEFT OUTER JOIN tbl_artikel AS a ON t.FS_Artikel = a.Artikel_ID
ORDER BY Person_ID
```

Gleiche Ausgabe wie vorher, da in diesem Fall alle inner-Einträge durch den left outer join erfasst werden

```
SELECT * FROM tbl_personen AS p INNER JOIN tbl_rechnungen AS r
  ON p.Person_ID = r.Person_FS
  LEFT OUTER JOIN tbl_trans AS t ON r.Rechnungs_ID = t.FS_Rechnung
    LEFT OUTER JOIN tbl_artikel AS a ON t.FS_Artikel = a.Artikel_ID
ORDER BY Person_ID
```

Jetzt erscheint auch Lars Peronik, da sich left outer jetzt auch auf die Rechnungstabelle bezieht

```
SELECT * FROM tbl_personen AS p LEFT OUTER JOIN tbl_rechnungen AS r
  ON p.Person_ID = r.Person_FS
  LEFT OUTER JOIN tbl_trans AS t ON r.Rechnungs_ID = t.FS_Rechnung
    LEFT OUTER JOIN tbl_artikel AS a ON t.FS_Artikel = a.Artikel_ID
ORDER BY Person_ID
```

Jetzt erscheinen alle Personen, da sich left outer jetzt auch auf die tbl_personen bezieht