Aufgabe 1: SQL

Gegeben sind folgende Relationen aus einer Personalverwaltung:

```
Mitarbeiter : {[ MitarbeiterID, Vorname, Nachname, Vorgesetzter[Mitarbeiter], Abtei
lungsID[Abteilung], Telefonnummer, Gehalt ]}

Abteilung : {[ AbteilungsID, Bezeichnung ]}
```

```sql
CREATE TABLE Abteilung(
  AbteilungsID INTEGER PRIMARY KEY,
  Bezeichnung VARCHAR(30)
);

CREATE TABLE Mitarbeiter(
  MitarbeiterID INTEGER PRIMARY KEY,
  Vorname VARCHAR(30),
  Nachname VARCHAR(30),
  Vorgesetzter INTEGER REFERENCES Mitarbeiter(MitarbeiterID),
  AbteilungsID INTEGER REFERENCES Abteilung(AbteilungsID),
  Telefonnummer VARCHAR(50),
  Gehalt DOUBLE PRECISION
);

INSERT INTO Abteilung VALUES
  (1, 'Buchhaltung'),
  (2, 'Vertrieb'),
  (42, 'Managment'),
  (4, 'Qualitätskontrolle'),
  (5, 'Produktion');

INSERT INTO Mitarbeiter
  (MitarbeiterID, Vorname, Nachname, Vorgesetzter, AbteilungsID, Telefonnummer,
  ↪ Gehalt)
VALUES
  (1,  'Hans',  'Meier',    11,   4, '023/13432', 2335),
  (2,  'Fred',  'Wolitz',   11,   2, '0233/413432', 1233),
  (11, 'Lea',   'Müller',   NULL, 42, '0343/3452', 5875),
  (3,  'Till',  'Fuchs',    2,    1, '023/13344', 2345),
  (4,  'Fred',  'Hase',     11,   4, '04/453432', 1334),
  (12, 'Gerd',  'Navratil', NULL, 42, '0345/552', 7154),
  (6,  'Jürgen', 'Schmidt', 12,   5, '097/dfg854', 654);
```

(a) Schreiben Sie eine SQL-Anfrage, die *Vor-* und *Nachnamen* der *Mitarbeiter* aller *Abteilungen* mit der Bezeichnung *„Buchhaltung“* ausgibt, absteigend sortiert nach *Mitarbeiter-ID*.

```sql
SELECT Vorname, Nachname
FROM Mitarbeiter m, Abteilung a
WHERE
  m.AbteilungsID = a.AbteilungsID AND
  a.Bezeichnung = 'Buchhaltung'
ORDER BY m.MitarbeiterID DESC;
```

```
 vorname | nachname
---------+----------
 Till    | Fuchs
```

```
 Hans     | Meier
(2 rows)
```

(b) Schreiben Sie eine SQL-Anfrage, die die Nachnamen aller Mitarbeiter mit dem Nachnamen ihres jeweiligen direkten Vorgesetzten ausgibt. Mitarbeiter ohne Vorgesetzten sollen in der Ausgabe ebenfalls enthalten sein. In diesem Fall soll der Nachname des Vorgesetzten NULL sein.

```
SELECT m.Nachname AS Mitarbeiter, v.Nachname AS Vorgesetzter
FROM Mitarbeiter m LEFT OUTER JOIN Mitarbeiter v
ON m.Vorgesetzter = v.MitarbeiterID;
```

```
 mitarbeiter | vorgesetzter
-------------+--------------
 Meier       | Müller
 Wolitz      | Müller
 Müller      |
 Fuchs       | Wolitz
 Hase        | Müller
 Navratil    |
 Schmidt     | Navratil
(7 rows)
```

(c) Schreiben Sie eine SQL-Anfrage, die die 10 Abteilungen ausgibt, deren Mitarbeiter das höchste Durchschnittsgehalt haben. Ausgegeben werden sollen der Rang (1 = höchstes Durchschnittsgehalt bis 10 = niedrigstes Durchschnittsgehalt), die Bezeichnung sowie das Durchschnittsgehalt der Abteilung. Gehen Sie davon dass es keine zwei Abteilungen mit gleichem Durchschnittsgehalt gibt. Sie können der Übersichtlichkeit halber Views oder With-Anweisungen verwenden. Verwenden Sie jedoch keine datenbanksystemspezifischen Erweiterungen wie `limit` oder `rownum`.

```
CREATE VIEW Durchschnittsgehälter AS
SELECT Abteilung.AbteilungsID, Bezeichnung,
  AVG (Gehalt) AS Durchschnittsgehalt
FROM Mitarbeiter, Abteilung
WHERE Mitarbeiter.AbteilungsID = Abteilung.AbteilungsID
GROUP BY Abteilung.AbteilungsID, Bezeichnung;

SELECT a.Bezeichnung, a.Durchschnittsgehalt, COUNT (*) AS Rang
FROM Durchschnittsgehälter a, Durchschnittsgehälter b
WHERE a.Durchschnittsgehalt <= b.Durchschnittsgehalt
GROUP BY a.AbteilungsID, a.Bezeichnung, a.Durchschnittsgehalt
HAVING COUNT(*) <= 10
ORDER BY Rang ASC;
```

```
 bezeichnung | durchschnittsgehalt | rang
-------------+---------------------+------
 Managment   |              6514.5 |    1
 Buchhaltung |                2340 |    2
 Vertrieb    |              1283.5 |    3
 Produktion  |                 654 |    4
```

```
(4 rows)
```

(d) Schreiben Sie eine SQL-Anfrage, die das Gehalt aller Mitarbeiter aus der Abteilung mit der AbteilungsID 42 um 5% erhöht.

```
 vorname | nachname | gehalt
---------+----------+--------
 Lea     | Müller   |   5875
 Gerd    | Navratil |   7154
(2 rows)
```

```sql
SELECT Vorname, Nachname, Gehalt
FROM MITARBEITER
WHERE AbteilungsId = 42
ORDER BY Gehalt;

UPDATE Mitarbeiter
SET Gehalt = 1.05 * Gehalt
WHERE AbteilungsID = 42;

SELECT Vorname, Nachname, Gehalt
FROM MITARBEITER
WHERE AbteilungsId = 42
ORDER BY Gehalt;
```

```
 vorname | nachname |      gehalt
---------+----------+-------------------
 Lea     | Müller   |           6168.75
 Gerd    | Navratil | 7511.700000000001
(2 rows)
```

(e) Alle *Abteilungen* mit Bezeichnung *„Qualitätskontrolle“* sollen zusammen mit den Datensätzen ihrer *Mitarbeiter* gelöscht werden. `ON DELETE CASCADE` ist für keine der Tabellen gesetzt. Schreiben Sie die zum Löschen notwendigen SQL-Anfragen.

```
 vorname | nachname
---------+----------
 Hans    | Meier
 Fred    | Wolitz
 Lea     | Müller
 Till    | Fuchs
 Fred    | Hase
 Gerd    | Navratil
 Jürgen  | Schmidt
(7 rows)

 abteilungsid |    bezeichnung
--------------+--------------------
            1 | Buchhaltung
            2 | Vertrieb
           42 | Managment
```

```
             4 | Qualitätskontrolle
             5 | Produktion
(5 rows)
```

```sql
SELECT Vorname, Nachname FROM Mitarbeiter;
SELECT * FROM Abteilung;

DELETE FROM Mitarbeiter
WHERE AbteilungsID IN (
  SELECT a.AbteilungsID
  FROM Abteilung a
  WHERE a.Bezeichnung = 'Qualitätskontrolle'
);

DELETE FROM Abteilung
WHERE Bezeichnung = 'Qualitätskontrolle';

SELECT Vorname, Nachname FROM Mitarbeiter;
SELECT * FROM Abteilung;
```

```
 vorname | nachname
---------+----------
 Fred    | Wolitz
 Lea     | Müller
 Till    | Fuchs
 Gerd    | Navratil
 Jürgen  | Schmidt
(5 rows)

 abteilungsid | bezeichnung
--------------+-------------
            1 | Buchhaltung
            2 | Vertrieb
           42 | Managment
            5 | Produktion
(4 rows)
```

(f) Alle Mitarbeiter sollen mit SQL-Anfragen nach den Telefonnummern anderer Mitarbeiter suchen können. Sie dürfen jedoch das Gehalt der Mitarbeiter nicht sehen können. Erläutern Sie in zwei bis drei Sätzen eine Möglichkeit, wie dies in einem Datenbanksystem realisiert werden kann, ohne die gegebenen Relationen, die Tabellen als abgelegt sind, zu verändern. Sie brauchen hierzu keinen SQL-Code schreiben.

Wir könnten eine VIEW erstellen, die zwar Namen und ID der anderen Mitarbeiter, sowie ihre Telefonnummern enthält (evtl. auch Abteilungsbezeichnung und ID), aber eben nicht das Gehalt: Mitarbeiter arbeiten auf eingeschränkter Sicht.

Alternativ mit `GRANT`:

explizit mit `SELECT` die Spalten auswählen, die man lesen können soll (auf nicht angegebene Spalten ist kein Zugriff möglich)

```
GRANT SELECT (Vorname, Nachname, Telefonnummer)
ON Mitarbeiter TO postgres;
```