Aufgabe 3

Formulieren Sie folgende Anfragen in SQL gegen die angegebene Datenbank aus einer imaginären Serie.

```
Figur : {[ Id, Name, Schwertkunst, Lebendig, Titel ]}
gehört_zu : {[ Id, Familie, FK (Id) references Figur(Id), FK (Familie) references Familie(Id)
]}
Familie : {[ Id, Name, Reichtum, Anführer ]}
Drache : {[ Name, Lebendig ]}
besitzt : {[ Id, Name, FK (Id) references Figur(Id), FK (Name) references Drache(Name)
]}
Festung : {[ Name, Ort, Ruine ]}
besetzt : {[ Familie, Festung, FK (Familie) references Familie(Id), FK (Festung) references
Festung(Name) ]}
lebt : {[ Id, Name, FK (Id) references Figur(Id), FK (Name) references Festung(Name)
]}
```

```sql
CREATE TABLE Figur (
  Id integer PRIMARY KEY,
  Name varchar(20),
  Schwertkunst integer,
  Lebendig boolean,
  Titel varchar(50)
);

CREATE TABLE Familie (
  Id integer PRIMARY KEY,
  Name varchar(20),
  Reichtum numeric(11,2),
  Anführer varchar(20)
);

CREATE TABLE gehört_zu (
  Id integer REFERENCES Figur(id),
  Familie integer REFERENCES Familie(id)
);

CREATE TABLE Drache (
  Name varchar(20) PRIMARY KEY,
  Lebendig boolean
);

CREATE TABLE besitzt (
  Id integer REFERENCES Figur(Id),
  Name varchar(20) REFERENCES Drache(Name)
);

CREATE TABLE Festung (
  Name varchar(20) PRIMARY KEY,
  Ort varchar(30),
  Ruine boolean
);

CREATE TABLE besetzt (
  Familie integer REFERENCES Familie(Id),
  Festung varchar(20) REFERENCES Festung(Name)
);

CREATE TABLE lebt (
  Id integer REFERENCES Figur(Id),
```

```
  Name varchar(20) REFERENCES Festung(Name)
);

INSERT INTO Figur VALUES
  (1, 'Eddard Stark', 5, FALSE, 'Lord von Winterfell'),
  (2, 'Rodd Stark', 4, FALSE, 'Lord von Winterfell'),
  (3, 'Tywin Lennister', 5, FALSE, 'Lord von Casterlystein'),
  (4, 'Cersei Lennister', 2, TRUE, 'Lady von Casterlystein'),
  (5, 'Brandon Stark', 0, TRUE, 'König der Andalen'),
  (6, 'Jon Schnee', 5, TRUE, 'König-jenseits-der-Mauer');

INSERT INTO Familie VALUES
  (1, 'Haus Stark', 76873.12, 'Eddard Stark'),
  (2, 'Haus Lennister', 82345.43, 'Tywin Lennister');

INSERT INTO gehört_zu VALUES
  (1, 1),
  (2, 1),
  (3, 2),
  (4, 2),
  (5, 1),
  (6, 1);

INSERT INTO Festung VALUES
  ('Roter Bergfried', 'Westeros', FALSE),
  ('Casterlystein', 'Westeros', FALSE),
  ('Winterfell', 'Westeros', FALSE);

INSERT INTO besetzt VALUES
  (1, 'Winterfell'),
  (2, 'Roter Bergfried'),
  (2, 'Casterlystein');

INSERT INTO lebt VALUES
  (1, 'Winterfell'),
  (2, 'Winterfell'),
  (3, 'Casterlystein'),
  (4, 'Roter Bergfried'),
  (5, 'Winterfell'),
  (6, 'Winterfell');
```

(a) Geben Sie für alle Figuren an, wie oft alle vorhandenen Titel vorkommen.

```
SELECT
  Name,
  Titel,
  (SELECT COUNT(*) FROM Figur g WHERE g.Titel = f.Titel) AS
  ↪ Anzahl_Titel
FROM Figur f;
```

(b) Welche Figuren (Name ist gesucht) kommen aus „Kings Landing“?

```
SELECT
  f.Name
FROM
  Figur f,
  Festung b,
  lebt l
WHERE
  b.Name = l.Name AND
```

```
    f.Id = l.Id AND
    b.Ort = 'Königsmund';
```

(c) Geben Sie für jede Familie (Name) die Anzahl der zugehörigen Charaktere und Festungen an.

```
SELECT
  f.Name,
  (SELECT COUNT(*) FROM Figur fi, gehört_zu ge WHERE fi.id = ge.id AND
   ↪  ge.Familie = f.id) AS Anzahl_Charaktere,
  (SELECT COUNT(*) FROM Festung fe, besetzt be WHERE fe.Name =
   ↪  be.Festung AND be.Familie = f.id) AS Anzahl_Festungen
FROM
  Familie f;
```

(d) Gesucht sind die besten fünf Schwertkämpfer aus Festungen aus dem Ort „Westeros“. Es soll der Name, die Schwertkunst und die Platzierung ausgegeben werden. Die Ausgabe soll nach der Platzierung sortiert erfolgen.

```
-- Problem: Es gibt 3 mal 3. Platz und nicht 3 mal 1. Platz
SELECT f1.Name, f1.Schwertkunst, COUNT(*) FROM Figur f1, Figur f2
WHERE f1.Schwertkunst <= f2.Schwertkunst
GROUP BY f1.Name, f1.Schwertkunst
ORDER BY COUNT(*)
LIMIT 5;
```

(e) Schreiben Sie eine Anfrage, die alle Figuren löscht, die tot sind. Das Attribut *Lebendig* kann dabei die Optionen „ja“ und „nein“ annehmen.

```
-- Nur zu Testzwecken auflisten:
SELECT * FROM Figur;
SELECT * FROM gehört_zu;

-- PostgreSQL unterstützt kein DELETE JOIN
-- DELETE f, g, b, l FROM Figur AS f
--  JOIN gehört_zu AS g ON f.id = g.id
--  JOIN besitzt AS b ON f.id = b.id
--  JOIN lebt AS l ON f.id = l.id
-- WHERE f.Lebendig = FALSE;

DELETE FROM gehört_zu WHERE id IN (SELECT id FROM Figur WHERE Lebendig
 ↪  = FALSE);
DELETE FROM besitzt WHERE id IN (SELECT id FROM Figur WHERE Lebendig =
 ↪  FALSE);
DELETE FROM lebt WHERE id IN (SELECT id FROM Figur WHERE Lebendig =
 ↪  FALSE);
DELETE FROM Figur WHERE Lebendig = FALSE;

-- Nur zu Testzwecken auflisten:
SELECT * FROM Figur;
SELECT * FROM gehört_zu;
```

(f) Löschen Sie die Spalten „Lebendig“ aus der Datenbank.

```sql
ALTER TABLE Figur DROP COLUMN Lebendig;
ALTER TABLE Drache DROP COLUMN Lebendig;
```

(g) Erstellen Sie eine weitere Tabelle mit dem Namen *Waffen*, welche genau diese auflistet. Eine Waffe ist genau einer Figur zugeordnet, hat einen eindeutigen Namen und eine Stärke zwischen 0 und 5. Wählen Sie sinnvolle Typen für die Attribute.

```sql
CREATE TABLE Waffen (
  Name varchar(20) PRIMARY KEY,
  Figur integer NOT NULL REFERENCES Figur(Id),
  Stärke integer NOT NULL CHECK(Stärke BETWEEN 0 AND 5)
);

-- Sollte funktionieren:
INSERT INTO Waffen VALUES
  ('Axt', 1, 5);

-- Sollte Fehler ausgeben:
-- INSERT INTO Waffen VALUES
--  ('Schleuder', 1, 6);
```