Aufgabe 3

Gegeben sei das folgende Datenbank-Schema, das für die Speicherung der Daten einer Schule entworfen wurde, zusammen mit einem Teil seiner Ausprägung. Die Primärschlüssel-Attribute sind jeweils unterstrichen.

Die Relation *Schüler* enthält allgemeine Daten zu den Schülerinnen und Schülern. Schülerinnen und Schüler nehmen an Prüfungen in verschiedenen Unterrichtsfächern teil und erhalten dadurch Noten. Diese werden in der Relation *Noten* abgespeichert. Prüfungen haben ein unterschiedliches Gewicht. Beispielsweise hat ein mündliches Ausfragen oder eine Extemporale das Gewicht 1, während eine Schulaufgabe das Gewicht 2 hat.

Schüler:

SchülerID	Vorname	Nachname	Klasse
1	Laura	Müller	4A
2	Linus	Schmidt	4A
3	Jonas	Schneider	4A
4	Liam	Fischer	4B
5	Tim	Weber	4B
6	Lea	Becker	4B
7	Emilia	Klein	4C
8	Julia	Wolf	4C

Noten:

SchülerID [Schüler]	Schulfach	Note	Gewicht	Datum
1	Mathematik	3	2	23.09.2017
1	Mathematik	1	1	03.10.2017
1	Mathematik	2	2	15.10.2017
1	Mathematik	4	1	11.11.2017

```sql
CREATE TABLE Schüler (
  SchülerID INTEGER PRIMARY KEY NOT NULL,
  Vorname VARCHAR(20),
  Nachname VARCHAR(20),
  Klasse VARCHAR(5)
);

CREATE TABLE Noten (
  SchülerID INTEGER NOT NULL,
  Schulfach VARCHAR(20),
  Note INTEGER,
  Gewicht INTEGER,
  Datum DATE,
  PRIMARY KEY (SchülerID, Schulfach, Datum),
  FOREIGN KEY (SchülerID) REFERENCES Schüler(SchülerID)
);

INSERT INTO Schüler VALUES
  (1, 'Laura', 'Müller', '4A'),
  (2, 'Linus', 'Schmidt', '4A'),
  (3, 'Jonas', 'Schneider', '4A'),
  (4, 'Liam', 'Fischer', '4B'),
  (5, 'Tim', 'Weber', '4B'),
  (6, 'Lea', 'Becker', '4B'),
  (7, 'Emilia', 'Klein', '4C'),
  (8, 'Julia', 'Wolf', '4C');

INSERT INTO Noten VALUES
```

```
(1, 'Mathematik', 3, 2, '23.09.2017'),
(1, 'Mathematik', 1, 1, '03.10.2017'),
(1, 'Mathematik', 2, 2, '15.10.2017'),
(1, 'Mathematik', 4, 1, '11.11.2017');
```

(a) Geben Sie die SQL-Befehle an, die notwendig sind, um die oben dargestellten Tabellen in einer SQL-Datenbank anzulegen.

```sql
CREATE TABLE IF NOT EXISTS Schüler (
  SchülerID INTEGER PRIMARY KEY NOT NULL,
  Vorname VARCHAR(20),
  Nachname VARCHAR(20),
  Klasse VARCHAR(5)
);

CREATE TABLE IF NOT EXISTS Noten (
  SchülerID INTEGER NOT NULL,
  Schulfach VARCHAR(20),
  Note INTEGER,
  Gewicht INTEGER,
  Datum DATE,
  PRIMARY KEY (SchülerID, Schulfach, Datum),
  FOREIGN KEY (SchülerID) REFERENCES Schüler(SchülerID)
);
```

(b) Entscheiden Sie jeweils, ob folgende Einfügeoperationen vom gegebenen Datenbanksystem (mit der angegebenen Ausprägungen) erfolgreich verarbeitet werden können und begründen Sie Ihre Antwort kurz.

```sql
INSERT INTO
Schüler (SchülerID, Vorname, Nachname, Klasse)
VALUES (9, 'Johannes', 'Schmied', '4C');
```

richtig:

Nein, denn die Spalte, die Primärschlüssel heißt Schüler und nicht SchülerID. Außerdem existiert bereits ein Schuüler mit der ID 6.

Musterlösung:

nein, SchülerID muss als Primärschlüssel eindeutig sein und ist bereits vergeben

```sql
INSERT INTO Noten VALUES (6, 'Chemie', 1, 2, '1.4.2020');
```

richtig:

Nein, Datum ist zwingend nötig, da es im Primärschlüssel enthalten ist. Es gibt keine Schüler mit der ID 9. Der müsste vorher angelegt sein, da die Spalte SchülerID von Noten auf den Fremdschlüssel Schüler aus der Schülertabelle verweist.

Musterlösung:

nein, Schüler mit der ID 9 existiert noch nicht, sodass diese Noten nicht eingetragen werden können, da Integritätsbedingungen nicht erfüllt (SchülerID ist Fremdschlüssel), zudem fehlt das Datum. Da es sich hier um ein Schlüsselattribut handelt, kann es nicht NULL sein.

(c) Geben Sie die Befehle für die folgenden Aktionen in SQL an. Beachten Sie dabei, dass die Befehle auch noch bei Änderungen des oben gegebenen Datenbank- zustandes korrekte Ergebnisse zurückliefern müssen.

- Die Schule möchte verhindern, dass in die Datenbank mehrere Kinder mit dem selben Vornamen in die gleiche Klasse kommen. Dies soll bereits auf Datenbankebene verhindert werden. Dabei sollen die Primärschlüssel nicht verändert werden. Geben Sie den Befehl an, der diese Änderung durchführt.

  falsch:

```sql
ALTER TABLE Schüler ALTER COLUMN Vorname ADD UNIQUE;
```

  Musterlösung:

```sql
ALTER TABLE Schüler
ADD CONSTRAINT Vorname_eindeutig UNIQUE (Vorname, Klasse);
```

- Der Schüler Tim Weber (SchülerID: 5) wechselt die Klasse. Geben Sie den SQL-Befehl an, der den genannten Schüler in die Klasse "4C" überführt.

  richtig:

```sql
UPDATE Schüler
SET Klasse = '4C'
WHERE Vorname = 'Tim' AND Nachname = 'Weber' AND SchülerID = 5;
```

  Musterlösung:

```sql
UPDATE Schüler
SET Klasse = '4C' WHERE SchülerID = 5;
```

- Die Schülerin Laura Müller (SchülerID: 1) zieht um und wechselt die Schule. Löschen Sie die Schülerin aus der Datenbank. Nennen Sie einen möglichen Effekt, welcher bei der Verwendung von Primär- und Fremdschlüsseln auftreten kann.

  richtig:

```sql
DELETE FROM Noten WHERE SchülerID = 1;
DELETE FROM Schüler WHERE Schüler = 1
```

  Es könnte passieren, dass man vergisst die Einträge in Noten zu löschen, denn der Fremdschlüssel SchülerID verweist auf den Primärschlüssel SchülerID in Schüler.

  Musterlösung:

Löschen aller Noten von Laura Müller, falls ON DELETE CASCADE gesetzt. Oder es müssen erst alle Fremdschlüsselverweise auf diese SchülerID in Noten gelöscht werden

- Erstellen Sie eine View „DurchschnittsNoten“, die die folgenden Spalten beinhaltet: Klasse, Schulfach, Durchschnittsnote Hinweis: Beachten Sie die Gewichte der Noten.

  Musterlösung:

```sql
CREATE VIEW DurchschnittsNoten AS (
  (SELECT s.Klasse, n.Schulfach, (SUM(n.Note * n.Gewicht) /
  ↪  SUM(n.Gesicht)) AS Durchschnittsnote
  FROM Noten n, Schüler s
  WHERE s.SchülerID = n.SchülerID AND n.Gewicht = 1
  GROUP BY s.Klasse, n.Schulfach)
);
```

- Geben Sie den Befehl an, der die komplette Tabelle „Noten“ löscht.

  Musterlösung:

```sql
DROP TABLE Noten;
```

(d) Formulieren Sie die folgenden Anfragen in SQL. Beachten Sie dabei, dass sie SQL-Befehle auch noch bei Änderungen der Ausprägung die korrekten Anfrageergebnisse zurückgeben sollen.

- Gesucht ist die durchschnittliche Note, die im Fach Mathematik vergeben wird.

  Hinweis: Das Gewicht ist bei dieser Anfrage nicht relevant

```sql
SELECT AVG(Note)
FROM Noten
WHERE Schulfach = 'Mathematik';
```

  Musterlösung nimmt View zu hand.

  Musterlösung:

```sql
SELECT AVG(Note) AS durchschnittlicheNote
FROM DurchschnittsNoten
WHERE Schulfach = 'Mathematik';
```

- Berechnen Sie die Anzahl der Schüler, die im Fach Mathematik am 23.09.2017 eine Schulaufgabe (d.h. Gewicht=2) geschrieben haben.

  Musterlösung:

  Ich habe in der WHERE Klauses das Schulfach vergessen. Sonst richtig. Außerdem habe ich das Datum einfach übernommen.

```sql
SELECT COUNT(SchülerID) AS AnzahlSchüler
FROM Noten
WHERE Datum = 2017-09-23 AND Gewichtung = 2 AND Schulfach =
↪  'Mathematik';
```

- Geben Sie die SchülerID aller Schüler zurück, die im Fach Mathematik mindestens drei mal die Schulnote 6 geschrieben haben.
  Musterlösung:

```
SELECT SchülerID
FROM Noten
WHERE Schulfach = 'Mathematik' AND Note = 6
GROUP BY SchülerID
HAVING COUNT(*) >= 3;
```

- Gesucht ist der Notendurchschnitt bezüglich jedes Fachs der Klasse „4A“.
  Musterlösung:

```
SELECT n.Schulfach, AVG (n.Note)
FROM Schüler s, Noten n
WHERE s.SchülerID = n.SchülerID AND s.Klasse = '4 A '
GROUP BY n.Schulfach
```

  Hier wäre Gewicht unberücksichtigt, also möglicherweise besser, auf die oben erstellte View zurückgreifen:

```
SELECT Schulfach, durchschnittlicheNote,
FROM DurchschnittsNoten
WHERE Klasse = '4a';
```

(e) Geben Sie jeweils an, welchen Ergebniswert die folgenden SQL-Befehle für die gegebene Ausprägung zurückliefern.

```
SELECT COUNT(DISTINCT Klasse)
FROM
Schüler NATURAL JOIN Noten;
```

Musterlösung:

1 → 4A von Laura Müller

```
SELECT COUNT(all Klasse)
FROM
Noten, Schüler;
```

Musterlösung:

32 → Kreuzprodukt, zählt alle Einträge in Klasse

```
SELECT COUNT(Note)
FROM
Schüler NATURAL LEFT OUTER JOIN Noten;
```

Musterlösung:

4 → null-Werte nicht mitgezählt, 4 Noten von Laura

```
SELECT COUNT(*)
FROM
Schüler NATURAL LEFT OUTER JOIN Noten;
```

Musterlösung:

11 $\rightarrow$ alle Schüler, Laura dabei 4-mal, weil 4 Noten