1 SQL

1.1 Probleme

1. Auswahl im Select, wo ColumnName mit ColumnName direkt verglichen wird

```
{sql}
      SELECT *
FROM Besucher
WHERE EXISTS (SELECT 1
  FROM Besucher
  WHERE Vorname=Vorname AND
  NOT (Name=Name));
```

1.2 Subqueries (in WHERE) mit EXISTS

```
{sql}
SELECT *
FROM Besucher AS x
WHERE EXISTS (SELECT 1
  FROM Besucher AS y
  WHERE x.Vorname=y.Vorname AND
  NOT (x.Name=y.Name));
```

1.3 Joins

1.3.1 Equijoin

Kreuzprodukt + Selektionsprädikat für gemeinsame Attrivute + autom. Umbenennung/Projektion von Attributen

```
SELECT R.A, R.B FROM R, S WHERE R.B = S.B;

SELECT R.A, R.B FROM R CROSS JOIN S WHERE R.B = S.B;

SELECT A,B FROM R NATURAL JOIN S;

SELECT R.A, R.B FROM R JOIN S ON R.B = S.B;
```

2 Aggregatfuntionen ohne Gruppierung

COUNT: zählt die Anzahl der Elemente
MAX: maximaler Wert der Elemente
MIN: minimaler Wert der Elemente
SUM: Summe der Werte der Elemente
AVG: Durchschnitt der Werte der Elemente

SUM und AVG müssen zählbar sein

2.1 COUNT

COUNT(¡AttributName¿) zählt nur diejenigen Tupel, bei denen der Wert des Attributs nicht NULL ist.
COUNT(*) zählt alle Tupel, es gibt kein Tupel, bei dem alle Attribute gleichzeigit NULL sein können.
COUNT(DISTINCT NAME): zählt die Anzahl verschiedener Namen

2.2 Bestellumsatz pro Kunde, unabhängig von der einzelnen Bestellung

```
SELECT kdNR, SUM(menge * preis) AS UMSATZ
FROM Bestellposition JOIN Kaufhistorei ON bestNr = bNr
GROUP BY kdNr;
```

2.3 Grösster Bestellumsatz

```
  SELECT MAX(Umsatz) AS GroessterUmsatz
  FROM (SELECT kdNR, SUM(menge * preis) AS UMSATZ
  FROM Bestellposition JOIN Kaufhistorei ON bestNr = bNr
  GROUP BY kdNr) AS x;
```

3 Aggregatsfunktionen mit Gruppierung

3.1 Reihenfolge der SQL Abfragen

1. FROM 2. WHERE 3. GORUP BY 4. HAVING 5. SELECT 6. ORDER BY

3.1.1 HAVING

HAVING COUNT(*) ¿ 2: Alle Sätze, wo alle numerischen Werte grösser als 2 sind.