Frage 1:	Welche Städtenamen sind in der Tabelle *City* gespeichert? **SELECT** name **FROM** city
Frage 2:	Wie heißt die Hauptstadt von *Bangladesh*? **SELECT** capital **FROM** country **WHERE** name='Bangladesh' Die Anfrage **SELECT** capital **FROM** country **WHERE** name **<** 'Bangladesh' liefert die Hauptstädte der Länder, die in der **lexikalischen Ordnung** vor Bangladesh stehen.
Frage 3:	Welche Städte haben mehr als 100000 Einwohner? **SELECT** name **FROM** city **WHERE** population>100000
Frage 4:	Welche Länder sind Mitglieder der *WHO*? **SELECT** country **FROM** is_member **WHERE** organization='WHO' **AND** type='member'
Frage 5:	Gesucht sind die Namen der Länder, die den Anfangsbuchstaben '**J**' haben. **SELECT** name **FROM** country **WHERE** name **LIKE 'J%'** **Achtung: LIKE** ist **Case Sensitive**. D. h., es unterscheidet zwischen Groß- und Kleinschreibung.
Frage 6:	Gesucht sind die Länder, deren Namen den Buchstaben '**j**' enthalten. **SELECT** name **FROM** country **WHERE** name **LIKE '%j%'**

Frage 7:	Gesucht ist die Liste der Länder, deren Namen **nicht** den Buchstaben **'b'** enthalten! **SELECT** name **FROM** country **WHERE** name **NOT LIKE '%b%'**
Frage 8:	Gesucht ist die Liste der Länder, deren **dritter Buchstabe** des Namens **'y'** ist! **SELECT** name **FROM** country **WHERE** name **LIKE '__y%'** Jedes Unterstrich „_“ ist Platzhalter für **genau ein** beliebiges Zeichen. %ist Platzhalter für beliebige Zeichen. D.h. % schließt auch **keine** Zeichen mit ein!
Frage 9:	Gesucht sind die Städte, deren Breitengrade zwischen **-5** und **5** liegen. **SELECT** name, latitude **FROM** city **WHERE** latitude **BETWEEN** -5 **AND** 5 Alternative Lösung: **SELECT** name, latitude **FROM** city **WHERE** latitude **>=** -5 **AND** latitude **>=** -5
Frage 10:	Gesucht sind die Länder, deren Flächen (Area) **nicht** zwischen 50000 und 60000 liegen. **SELECT** name **FROM** country **WHERE** area **NOT BETWEEN** 50000 **AND** 60000
Frage 11:	Welche Länder haben **keine** Hauptstadt? **SELECT** name **FROM** country **WHERE** capital **IS NULL** Das Schlüsselwort **IS** wird **nur** bei der Überprüfung von NULL-Werten verwendet.
Frage 12:	Von welchen Ländern ist die **Hauptstadt bekannt**? **SELECT** name **FROM** country **WHERE** capital **IS NOT NULL**

Frage 13: Gesucht sind die Länder, deren Einwohnerzahlen 69865 oder 68320 oder 57902 sind.

```
SELECT name, population
FROM country
WHERE population= 69865      OR
      population= 68320      OR
      population= 57902
```

Alternative Lösung:

```
SELECT name,population
FROM country
WHERE population IN (69865,68320,57902)
```

Frage 14: Gesucht sind die Länder, die Mitglieder der *EU* **oder** der *WHO* **oder** der *UNESCO* sind.

```
SELECT country,organization
FROM is_member
WHERE organization='EU'      OR
      organization='WHO'     OR
      organization='UNESCO'
```

Alternative Lösung:

```
SELECT country,organization
FROM is_member
WHERE organization IN ('EU','WHO','UNESCO')
```

Frage 15: Gesucht sind zum einen **alle** deutschen Städte **und** beliebige andere Städte**,** deren Einwohnerzahlen kleiner als 100000 sind.

```
SELECT name,country,population
FROM city
WHERE country='de' OR population<100000
```

Frage 16: Gesucht sind die deutschen Millionenstädte!

```
SELECT name, population
FROM city
WHERE country = 'de' AND
      population >= 1000000
```

Frage 17: Gesucht sind **alle** deutschen Städte und nur die französischen Städte, deren Einwohnerzahlen größer als 400000 sind.

```
SELECT name, population
FROM city
WHERE ountry = 'de'      OR
      country = 'fr'     AND
      population > 400000
```

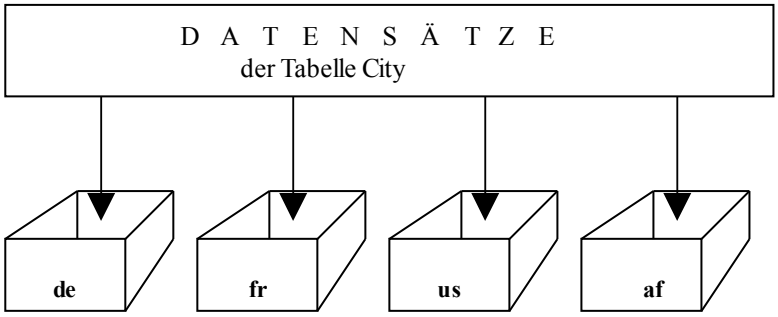
	Beachte die Auswertungsreihenfolge **NOT** vor **AND** vor **OR.** **Beispiel**: A **OR** B **AND** C Es werden zunächst B und C **AND**-verknüpft. Das Ergebnis wird dann mit A **OR**-verknüpft. (Das ist wie in der Mathemathik „·“ vor „-“: 3·4-5). Möchte man die Verknüpfungs-reihenfolge ändern, so kann dies durch Klammernsetzen erreicht werden, wie z.B: **(**A **OR** B**) AND** C Es werden jetzt zunächst A und B **OR**-verknüpft, das Ergebnis anschließend mit C **AND**-verknüpft.
Frage 18:	Gesucht sind alle Städte, die in der peruanischen Provinz Amazonas oder in der peruanischen Provinz La Libertad liegen. **SELECT** name **FROM** city **WHERE** country = 'pe' **AND** **(**province = 'Amazonas' **OR** province = 'La Libertad'**)** Würde man in der obigen Anfrage auf Klammern verzichten, so könnten in der Ausgabe auch Städte vorkommen, die zwar in einer Provinz namens La Libertad liegen, wobei das zugehörige Land, aber nicht notwendigerweise Peru sein muss. Die Provinz La Libertad liegt tatsächlich in mehreren Ländern. Alternativ könnte man die obige Anfrage wie folgt ohne Klammern lösen: **SELECT** name **FROM** city **WHERE** country = 'pe' **AND** province = 'Amazonas' **OR** country = 'pe' **AND** province = 'La Libertad'
Frage 19:	Gesucht sind die Städte, die **nicht** auf dem Breitengrad von 48,41 liegen. **SELECT** name, latitude **FROM** city **WHERE NOT** latitude **=** 48.41 **bzw.** **SELECT** name, latitude **FROM** city **WHERE** latitude **!=** 48.41

	bzw. **SELECT** name, latitude **FROM** city **WHERE** latitude **<>** 48.41
Frage 20:	Gesucht sind die deutschen Städte, die nicht auf dem Längengrad von 10,28 liegen. **SELECT** name, longitude **FROM** city **WHERE** country = 'de' **AND** **NOT** longitude **=** 48.41
Frage 21:	Gesucht sind die Städte, die auf dem Längengrad von 7,85 und auf dem Breitengrad von 48,41 liegen. **SELECT** name, longitude, latitude **FROM** city **WHERE** latitude = 48.41 **AND** longitude = 7.85
Frage 22:	Gesucht sind die Städte, die **nicht** gleichzeitig auf dem Längengrad von 7,85 und auf dem Breitengrad von 47,99 liegen. **SELECT** name, longitude, latitude **FROM** city **WHERE NOT** latitude = 48.41 **OR** **NOT** longitude = 7.85 bzw. **SELECT** name, longitude, latitude **FROM** city **WHERE NOT (**latitude = 48.41 **AND** longitude = 7.85**)** Die obige Frage hätte auch wie folgt gestellt werden können: Gesucht sind die Städte, die **nicht** auf dem Längengrad von 7,85 oder **nicht** auf dem Breitengrad von 47,99 liegen. Die zweite Variante der Lösung entsteht durch die Anwendung der **De-Morganschen-Regel:** • **NOT** (**A AND** B) = **NOT** A **OR NOT** B • **NOT (A OR B) = NOT A AND NOT B**

Frage 23:	Gesucht sind alle Städtenamen. Beachten Sie, dass die Ausgabe jede Stadt **nur einmal** enthält. **SELECT DISTINCT** name **FROM** city Der Primärschlüssel der Tabelle *City* besteht aus den Spalten *Name*, *Country* und *Province*. Somit kann es sein, dass zwei Städte denselben Namen haben, aber in unterschiedlichen Provinzen oder Ländern liegen. Duplikate werden durch die Verwendung von SELECT DISTINCT aus dem Ergebnis ausgeschlossen. Es wird aber empfohlen, DISTINCT nur zu verwenden, wenn es unbedingt notwendig ist, da die Eliminierung von Duplikaten eine vorherige Sortierung erfordert. Sortieren ist aber im Allgem. zeit- und rechaufwändig.
Frage 24:	Geben Sie alle Länder und ihre zugehörigen Einwohnerzahlen an. Dabei sollen die Einwohnerzahlen **aufsteigend sortiert** sein. **SELECT** name, population **FROM** country **ORDER BY** population **ASC** **bzw.** **SELECT** name, population **FROM** country **ORDER BY** 2 Mit dem ORDER BY <attribut> [ASC, DESC] Befehl können Ergebnisse nach bestimmten Attributen sortiert werden. ASC bedeutet, dass die Ergebnisse in *aufsteigender* Reihenfolge sortiert werden und DESC, dass die Ergebnisse in *absteigender* Reihenfolge sortiert werden. Verwendet man nur ORDER BY, wird das Ergebnis standardmäßig aufsteigend sortiert. Die **2** ist ebenfalls eine Abkürzung und bezeichnet im obigen Beispiel die Position von *Population* in der SELECT-Zeile.
Frage 25:	Erstellen Sie eine Liste aller Städte, **aufsteigend** bezüglich Land und **absteigend** bezüglich Breitengrad. **SELECT** * **FROM** city **ORDER BY** country **ASC**, latitude **DESC** Wird - wie in diesem Beispiel - nach zwei Attributen gleichzeitig sortiert, so hat das **erste Attribut eine höhere Priorität** als das zweite. Im obigen Beispiel wird bezüglich *Country* und *Latitude* gleichzeitig sortiert. Also hat *Country* die höhere Priorität. Das heißt, dass nach Länder komplett aufsteigend sortiert wird. Die Breitengrade werden dagegen nur bezüglich der Städte des gleichen Landes absteigend sortiert.

	Testen Sie, wie das Ergebnis der folgenden Anfrage sortiert ist! **SELECT** * **FROM** city **ORDER BY** latitude **DESC**, country
Frage 26:	Wie viele Länder gibt es in der Tabelle *Country?* **SELECT COUNT(*)** **FROM** country SQL bietet die folgenden **Aggregierungsfunktionen** an: • **COUNT():** zur Zählung von Zeilen • **MAX()** und **MIN()**: zur Bestimmung des kleinsten, bzw. des größten Elementes • **SUM():** zur Bildung der Summe • **AVG():** zur Bestimmung des Durchschnitts einer Menge von Zahlen Im obigen Beispiel liefert COUNT(*) liefert die Anzahl der Datensätze aus der Tabelle *Country*. Alternativ würde COUNT(Name) die Anzahl der Werte, die unter der Spalte *Name* gespeichert sind, ausgeben. Der Unterschied zwischen den beiden besteht aber darin, dass COUNT(Name) Null-Werte ignoriert und nicht mit-zählt, während COUNT(*) auch Zeilen, in denen manche Spaltenwerte Null sind, mitzählt. Aggregierungsfunktionen operieren immer auf eine **Menge** von Datensätzen (d.h. einer Menge von Zeilen) und fassen diese zu einem **einzelnen** Wert in der Ausgabe zusammen! Diese Eigenschaft verbietet es wiederum, in der SELECT-Zeile neben Aggregierungsfunktionen noch weitere Attribute aufzulisten (siehe nächste Frage).
Frage 27:	Wie groß ist der maximale und der minimale Flächenwert in der Tabelle *Country*? **SELECT MAX(**area**)**, **MIN(**area**)** **FROM** country Eine Anfrage der Form **SELECT** name, **MAX(**area**)** **FROM** country ist **nicht zulässig.** In einer SELECT-Zeile, in der mindestens eine Aggregierungsfunktion vorkommt, dürfen **keine** Attribute aufgelistet werden. Eine Ausnahme bilden aber SELECT-Befehle unter Verwendung der GROUP BY-Klausel (siehe Frage 33). Frage 61 zeigt aber einige Lösungsmöglichkeiten, auch den Namen der der Länder auszugeben.

Frage 28:	Wie groß ist die durchschnittliche Fläche aller Länder? **SELECT AVG(**area**)** **FROM** country
Frage 29:	Wie viele Länder sind Mitglied der EU? **SELECT COUNT(*)** **FROM** is_member **WHERE** organization = 'EU'
Frage 30:	Wie viele Einwohner haben alle Länder der Tabelle *Country* zusammen? **SELECT SUM(**population**)** **FROM** country
Frage 31:	Geben Sie das Gründungsdatum der jüngsten Organisation an: **SELECT MAX(**established**)** **FROM** organization Die obige Anfrage liefert nur das Gründungsdatum der jüngsten Organidsation. Es ist in der obigen Anfrage **nicht erlaubt** auch den Namen der Organisation in der SELECT-Zeile auszugeben! **Falsche Lösung!** **SELECT** name, **MAX(**established**)** **FROM** organization Frage 62 zeigt einige Lösungsmöglichkeiten, auch den Namen der gesuchten Organisation auszugeben.
Frage 32:	Wie viele unterschiedliche Organisationen sind in *Is_member* aufgeführt? **SELECT COUNT(DISTINCT** organization**)** **FROM** is_member
Frage 33:	Wie hoch sind die durchschnittlichen Einwohnerzahlen in den Städten **der einzelnen** Länder? **SELECT** country, **AVG(**population**)** **FROM** city **GROUP BY** country



In der obigen Anfrage werden alle Zeilen der Tabelle City, die den gleichen Wert im Attribut *Country* haben (d.h. alle Städte, die zu einem Land gehören), mittels GROUP BY Country zu einer Gruppe zusammengefasst.
Für jede der so entstandenen Gruppe wird die Summe der Einwohnerzahlen ihrer zugehörigen Städte gebildet. Somit gibt die **AVG()**-Funktion für **jede** dieser Gruppen genau **einen** Wert aus.

In den vorherigen Anfragen war jeweils die **ganze Tabelle** die **eine Gruppe.** Deshalb haben dort die Aggregierungsfunktionen jeweils auch nur einen Wert pro Abfrage geliefert.

Frage 34: Wie hoch sind die durchschnittlichen Einwohnerzahlen in den Provinzen **der einzelnen** Länder?

```
SELECT country, AVG(population)
FROM province
GROUP BY country
```

Frage 35: In welchen Ländern ist die durchschnittliche Einwohnerzahl in den Städten kleiner als 100000?

Falsche Lösung:

```
SELECT country, AVG(population)
FROM city
WHERE AVG(population) < 100000
GROUP BY country
```

Richtige Lösung:

```
SELECT country, AVG(population)
FROM city
GROUP BY country
HAVING AVG(population) < 100000
```

Mittels der WHERE-Zeile kann die gesamte Tabelle zunächst vorgefiltert werden. Um aber zusätzliche Bedingungen an die einzelnen Gruppen zu definieren, muss die HAVING-Klausel verwendet werden. D. h. nur mit **HAVING** können **zusätzliche** Bedingungen für die einzelnen **Gruppen** formuliert werden.

Beachte Sie:

Es ist **nicht erlaubt** Aggregierungsfunktionen in der WHERE-Zeile zu verwenden. Sie dürfen **nur** in der SELECT- und/oder in der HAVING-Zeile vorkommen.

Frage 36: Bestimme jede Organisation, *außer die EU*, deren Anzahl von Mitgliedern größer als 2 ist!

Das Ergebnis dieser Anfrage verdeutlicht den **Unterschied** zwischen **WHERE** und **HAVING**:

```
SELECT organization, COUNT(*)
FROM is_member
WHERE NOT organization = 'EU'
GROUP BY organization
HAVING COUNT (*) > 2
```

In der obigen Anfrage werden durch die WHERE-Bedingung zunächst alle EU-Zeilen aus der weiteren Betrachtung ausgeschlossen. Anschließend werden die jeweiligen Organisationen gruppiert. Das geschieht dadurch, alle Zeilen, die für die Spalte Organisation den gleichen Wert haben, zu einer Gruppe zusammengefasst werden. Nach dieser Aktion werden dann nur solche Organisationen ausgegeben, deren Gesamtvorkommen in der Tabelle größer als 2 ist. Die folgende Sequenz von Tabellen verdeutlicht nochmal den Ablauf:

Tabelle: *Is_member:*

Oranization	Country	Type
EU	de	member
WHO	de	member
UNESCO	at	member
UNESCO	af	member
NATO	us	member
WHO	af	member
UNSECO	de	member
UNESCO	fr	member

Durch die WHERE-Zeile werden zunächst alle EU-Zeilen ignoriert. Anschließend werden die Zeilen, die in der Spalte Organisation den gleichen Wert haben, mit GROUP BY Organization zu einer Gruppe zusammengefasst:

Oranization	Country	Type
WHO	de	member
WHO	af	member
NATO	us	member
UNSECO	de	member
UNESCO	fr	member
UNESCO	at	member
UNESCO	af	member
...	...	...

Die Ausgabe für die obige Beispiel Tabelle wäre dann UNESCO, da nur diese Gruppe die HAVING-Bedingung erfüllt (UNESCO hat offensichtlich vier Mitgliedsländer).
Das Beispiel hat deutlich gemacht, dass die WHERE-Bedingung zunächst ein Filter auf die **gesamte Tabelle** ist, während mit HAVING Bedingungen für die **einzelnen Gruppen** gestellt werden können.

Lösen Sie nun die Anfrage ohne die Verwendung der WHERE-Zeile!

Frage 37: Sortiere die Ausgabe von Aufgabe 36 nach Organisationen.

```
SELECT organization, COUNT(*)
FROM is_member
WHERE NOT organization = 'EU'
GROUP BY organization
HAVING COUNT (*) > 2
ORDER BY 1
```

Die letzte Anfrage verdeutlicht die komplette **allgemeine Struktur von SELECT-Befehlen:**

SELECT $A_1, A_2, \ldots$	Attribute der Antworttabelle
FROM $T_1, T_2, \ldots$	benötigte Tabellen
WHERE B	Auswahlbedingung
GROUP BY $B_1, \ldots$	Gruppierungsattribute
HAVING B	Gruppierungsbedingung
ORDER BY H	Sortierordnung

Für die **Auswertungsreihenfolge** gilt:

FROM-Klausel vor **WHERE**-Klausel vor **GROUP**-Klausel vor **HAVING**-Klausel vor **ORDER**-Klausel vor **SELECT**-Klausel.

Frage 38:	Geben Sie für jedes Meer die Anzahl der Länder an, die an diesem Meer liegen. **SELECT** sea, **COUNT**(**DISTINCT** country) **FROM** geo_sea **GROUP BY** sea Hier ist ein **DISTINCT** notwendig, da die Tabelle geo_sea eine Beziehungstabelle zwischen den Tabellen Province und Sea ist und mehrere Provinzen desselben Landes am gleichem Meer liegen können!
Frage 39:	Geben Sie von jedem Kontinent den Namen sowie die Anzahl der Länder an, die auf diesem Kontinent liegen. Die benötigte Tabelle dazu ist *Encompasses*. **SELECT** continent, **COUNT(*)** **FROM** encompasses **GROUP BY** continent
Frage 40:	Wie viele Länder sind Mitglied der EU (Löse diese Aufgabe ohne Verwendung einer WHERE-Bedingung)? **SELECT COUNT(*)** **FROM** is_member **GROUP BY** organization **HAVING o**rganization = 'EU' Variante 2: Siehe Aufgabe 30
Frage 41:	Geben Sie die vollständigen Namen (enthalten in der Tabelle *Country*) aller Länder an, die Mitglieder der EU sind! Um diese Anfrage lösen zu können, müssen nun, anders als bisher, **zwei Tabellen** in der FROM-Zeile angegeben werden. Die Informationen über die EU-Mitgliedschaften sind in der Tabelle *Is_member* vorhanden. Leider sind die Länder in der Tabelle *Is_member* aber nur mit ihrem Länderkürzel dargestellt. Dies hängt damit zusammen, dass die Spalte *country* der Tabelle *Is_member* ein Fremdschlüssel ist, der auf die Spalte *code* der Tabelle *Country* zeigt. Die Spalte *code* enthält nur Länderkürzel. Zwischen den Tabellen *Is_member* und *Country* liegt eine 1:n-Beziehung vor. Um den Namen der Länder zu finden, müssen die Tabellen geeignet verknüpft werden.

Country:

Name	Code
Germany	de
France	fr
Afghanistan	af

Is_member:

Country	Orgaization
de	EU
fr	EU
af	WHO

`Country` **referenziert** `Code`

Ein erster Versuch, die Aufgabe zu lösen könnte wie folgt aussehen:

SELECT name, organization
FROM country**,** is_member
WHERE is_member.organization = 'EU'

(Wegen der Eindeutigkeit verwenden wir die *Tabelle.Spalte*-Notation.)

Das Datenbankmanagementsystem erzeugt aufgrund der angegebenen FROM-Zeile aus den beiden ursprünglichen Tabellen die folgende *temporäre Zwischentabelle*:

Zwischentabelle

	Name	Code	Country	Organization
1.	Germany	de	de	EU
2.	Germany	de	fr	EU
3.	Germany	de	af	WHO
4.	France	fr	de	EU
5.	France	fr	fr	EU
6.	France	fr	af	WHO
7.	Afghanistan	af	de	EU
8.	Afghanistan	af	fr	EU
9.	Afghanistan	af	af	WHO

In der *Zwischentabelle* ist **jede** Zeile der Tabelle *Country* mit **jeder** Zeile der Tabelle *Is_member* verbunden. Da jede Tabelle 3 Zeilen hat, besitzt die *Zwischentabelle* entsprechend 9 (3 x 3) Zeilen.

Abgesehen davon, dass die *Zwischentabelle* bei mehreren Zeilen in den ursprünglichen Tabellen **sehr groß** werden kann, enthalten manche Zeilen von ihr auch **„unsinnige“** Informationen (siehe z. B. Zeile 2).

Interessant sind aber die Zeilen 1, 5 und 9, da aus ihnen hervorgeht, **welches Land mit welchem Namen in welcher Organisation Mitglied ist**. Das sind aber genau die Zeilen, in denen die Spalten *code* und *country*

	dieselben Werte haben. Für die interessierenden Datensätze gilt also die Bedingung: **country.code = is_member.country**. (sog. **JOIN-Bedingung**) Diese JOIN-Bedingung fehlte aber in unserer obigen ersten Lösungsversuch. Damit das Datenbankmanagementsystem (DBMS) nicht eine riesige Zwischentabelle generiert, müssen wir ihm also noch in einer Art und Weise die obige JOIN-Bedingung mitteilen. D. h. das DBM muss vorher wissen, wie es die zwei in der FROM-Zeile angegebenen Tabellen miteinander verknüpfen soll, damit nur die interessierenden Zeilen in der Zwischentabelle abgelegt werden. Hierzu formulieren wir die **JOIN-Bedingung** in der WHERE-Zeile: **SELECT** name, organization **FROM** country, is_member **WHERE** is_member .organization = 'EU' **AND** **country.code = is_member .country** Die folgende Anfrage liefert dasselbe Ergebnis: **SELECT** name, organization **FROM** country **JOIN** is_member **ON** country.code = is_member.country **WHERE** is_member.organization = 'EU' In der zweiten Lösung wurde die JOIN-Bedingung in die FROM-Zeile verlagert. In diesem Fall ist die Verwendung von **JOIN** und **ON** notwendig!
Frage 42:	Geben Sie zu jeder Stadt den Namen des zugehörigen Landes an! **SELECT** city.name **AS** Stadt, country.name **AS** Land **FROM** city, country **WHERE** city.country = country.code oder: **SELECT** city.name **AS** Stadt, country.name **AS** Land **FROM** city **JOIN** country **ON** city.country = country.code Mittels **AS** in der SELECT-Zeile können Spalten der Ergebnistabelle umbenannt werden.
Frage 43:	Geben Sie die Ländernamen und die zu ihnen gehörenden Provinznamen an! **SELECT** country.name **AS** Land, province.name **AS** Prov **FROM** province **AS** p, country **AS** c **WHERE** province.country = country.code oder:

	SELECT country.name **AS** Land, province.name **AS** Prov **FROM** province **JOIN** country **ON** province.country = country.code Mittels AS in der FROM-Zeile können Aliase für Tabellen definiert werden. Bei manchen DBMS (z. B. Oracle) kann der Aliasname auch direkt hinter dem Tabellennamen stehen. In den folgenden Beispielen werden die Aliase der FROM-Zeile ohne AS geschrieben.
Frage 44:	Geben Sie zu jedem **Land in Asien** den Namen des Landes sowie die Namen aller in dem Land liegenden Berge an. Die Informationen über die Berge sind in der Tabelle *Geo_Mountain* gespeichert! **SELECT** c.name, g.mountain **FROM** country c, geo_mountain g, encompasses e **WHERE** g.country = c.code **AND** e.country = c.code **AND** e.continent = 'Asia'
Frage 45:	Geben Sie zu jedem Land in Asien den Namen des Landes, die Namen aller in dem Land liegenden Berge und deren Höhen an. Die Informationen über die Berge sind in der Tabelle *Geo_Mountain* gespeichert! Die Höhen der Berge finden Sie in der Tabelle *Mountain*. Zur Lösung dieser Aufgabe müssen drei Tabellen betrachtet werden: • *Encompasses*: enthält die Information Land -> Kontinent • *Geo_Mountain*: enthält die Information Land -> Berg • *Mountain*: enthält die Information über die Höhe der Berge • *Country*: enthält die Ländernamen In der **JOIN**-Bedingung muss also zunächst *Encompasses* mit *Geo_Mountain* verknüpft werden. Dies geschieht wie folgt: encompasses e **JOIN** geo_mountain g **ON** e.country = g.country Hieraus entsteht eine **neue Zwischentabelle**. Diese wird dann mit der Tabelle *Country* verknüpft: (encompasses e **JOIN** geo_mountain g **ON** e.country = g.country) **JOIN** country c **ON** c.code = e.country Zu guter Letzt müssen wir diese Zwischentabelle mit noch mit der Tabelle *Mountain* verknüpfen. ((encompasses e **JOIN** geo_mountain g **ON** e.country = g.country) **JOIN** country c **ON** c.code = e.country) **JOIN** mountain m **ON** g.mountain = m.name

Die gesamte Anfrage würde demnach wie folgt lauten:

```
SELECT c.name, m.name, m.height
FROM  ( (encompasses e JOIN geo_mountain g ON
          e.country = g.country)  JOIN country c ON
          c.code = e.country ) JOIN mountain m ON
                                  g.mountain = m.name
WHERE e.continent = 'Asia'
ORDER BY c.name
```

bzw.

```
SELECT c.name, m.name, m.height
FROM encompasses e, geo_mountain g, country c, mountain  m
WHERE e.country = g.country AND
        c.code = e. country   AND
        g.mountain = m.name AND
        e.continent = 'Asia'
ORDER BY c.name
```

Die Reihenfolge der JOIN's in der FROM-Zeile hat keinen Einfluss auf das Ergebnis der SQL-Anfrage. Es kann aber auf die Perfomance der Anfrage Auswirkungen haben.

Frage 46:

Geben Sie zu jedem Land den Namen des Landes sowie den Namen des *höchsten* Berges und deren Höhe an!

Mit der folgende Anfrage

```
SELECT g.country, MAX(m.height)
FROM geo_mountain g, mountain m
WHERE g.mountain = m.name
GROUP BY g.country
```

können nur die die Namen der Länder sowie die Höhe ihrer jeweiligen höchsten Berge **aber NICHT** die Namen dieser Berge angegeben werden. Warum?

Zur Erinnerung:

In der SELECT-Zeile dürfen nur solche Spaltenbezeichner stehen, die auch in der GROUP-BY-Klausel vorkommen. In der obigen Anfrage enthält die GROUP-BY-Klausel nur *g.country*. Also darf in der SELECT-Zeile auch nicht *m.name* auftauchen.

Es besteht aber die Möglichkeit, die obige Anfrage in Form einer **virtuellen Tabelle (View)** zwischen zu speichern und in einer Folgeanfrage wie eine Tabelle in der FROM-Zeile zu verwenden..

```
CREATE OR REPLACE VIEW Land_und_Berghoehe AS
SELECT g.country AS country, MAX(m.height) AS maxHoehe
FROM geo_mountain g, mountain m
WHERE g.mountain = m.name
GROUP BY g.country
```

Jetzt können wir die virtuelle Tabelle *Land_und_* Berghoehe die aus den Spalten *country* und *maxHoehe* besteht in der folgenden Anfrage verwenden:

```
SELECT c.name, m.height, m.name
FROM  Land_und_Berghoehe l, mountain m,
        country c, geo_mountain g
WHERE l.maxHoehe = m.height AND l.country = c.code AND
          c.code = g.country AND g.mountain = m.name
 ORDER BY 1
```

In der obigen Anfrage ist es **notwendig**, in der FROM-Zeile auch die Tabelle *geo_mountain* zu betrachten, da sonst Berge die zufällig dieselbe Höhe haben, aber in unterschiedlichen Ländern sich befinden auch einander zugeordnet werden können.

So gibt es ein Berg namens *Saser Kangri III* in Indien mit der Höhe 7495 m und ein Berg namens *Ismail Samani Peak* in Tajikistan ebenfalls mit der Höhe 7495 m. In Tajikistan ist *Ismail Samani Peak* der höchste Berg des Landes.

Die folgende (falsche) Lösung ohne *geo_mountain*

```
SELECT c.name, m.height, m.name
FROM  Land_und_Berghoehe l, mountain m, country c
WHERE l.maxHoehe = m.height AND
          l.country = c.code
ORDER BY 1
```

enthält in der Ausgabe **fälschlischerweise** auch den Datensatz: (Tajikistan,7495,Saser Kangri III).

Alternativ kann die obige Anfrage auch wie folgt mittels einer so genannten **Unterabfrage** gelöst werden:

```
SELECT c.name, m.name, m.height
  FROM country c, mountain m, geo_mountain g
 WHERE c.code = g.country AND
          g.mountain = m.name AND
  NOT EXISTS (SELECT *
                FROM geo_mountain g1, mountain m1
                WHERE g1.mountain = m1.name AND
                        g.country = g1.country    AND
                        m1.height > m.height)
```

Die zweite Anfrage besagt intuitiv:

Gebe zu jedem Land einen Berg und dessen Höhe nur dann an, wenn es in dem Land keinen anderen Berg gibt, dessen Höhe größer ist. SQL-Unterabfragen können wie im obigen Beispiel in der WHERE-Zeile eines anderen SQL-Ausdrucks, aber auch in der FROM- bzw. SELECT-Zeile vorkommen. Würde man in der ersten Lösung die virtuelle Tabelle durch ihre eigentliche SQL-Definition ersetzen, so hätte man ein Beispiel für eine Unterabfrage in der FROM-Zeile.

Frage 47: Geben Sie alle Paare von Ländern an, die im selben Kontinent liegen!

```
SELECT e1.country, e2.country
FROM encompasses ASe1, encompasses AS e2
WHERE e1.continent = e2.continent AND
        e1.country < e2.country
```

oder

```
SELECT e1.country, e2.country
FROM encompasses e1 JOIN encompasses e2 ON
      e1.continent = e2.continent
WHERE e1.country < e2.country
```

Da in der obigen Anfrage die Tabelle *encompasses* mit sich selbst verknüpft wird, **müssen** wir für jede Tabelle einen separaten Aliasnamen definieren (e1 und e2) definieren, da in solchen Fällen auch die Notation *Tabellenname.Spaltenname* zu Namenskollisionen führen wird.

Warum in der WHERE-Zeile zusätzlich die Bedingung **e1.country < e2.country** definiert wird, verdeutlicht die folgende Überlegung:

Continent	Country
Europe	de
Europe	fr
Asia	tr
Europe	tr

Der JOIN zwischen encompasses e1 und encompasses e2 liefert die folgende Zwischentabelle:

e1.Continent	e1.Country	e2.Continent	e2.Country
Europe	de	Europe	de
Europe	de	Europe	fr
Europe	de	Europe	tr
Europe	fr	Europe	de
Europe	fr	Europe	fr
Europe	fr	Europe	tr
Asia	tr	Asia	tr
Europe	tr	Europe	de
Europe	tr	Europe	fr

In dieser Tabelle sind offensichtlich die Spalten e1.continent und e2. continent identisch, so wie es in der obigen JOIN-Bedingung definiert wurde. Ohne die zusätzliche Bedingung e1.country < e2.country würde das Endergebnis der Anfrage wie folgt aussehen:

e1.Country	e2.Country
de	de
de	fr
de	tr
fr	de
fr	fr
fr	tr
tr	tr
tr	de
tr	fr

Abgesehen davon, dass uns die Paare (de,de), (af,af) und (tr,tr) gar nicht in der Ausgabe interessieren sind auch die Paare (de,fr) und (fr,de), (de,tr) und (tr,de) sowie (fr,tr) und (tr,fr) identisch. Die Bedingung **e1.country < e2.country** sorgt also dafür, dass aus der obigen Tabelle nur die folgenden Paare im Endergebnis erscheinen:

e1.Country	e2.Country
de	fr
de	tr
fr	tr

Im Folgenden behandeln wir einen Spezialfall der obigen Aufgabe. Dies

ergibt sich aus der Tatsache, dass neben der Türkei auch Russland und Ägypten in zwei Kontinente liegen. Betrachten wir als Tabelle *Encompasses* die folgende Tabelle:

Continent	Country
Europe	tr
Asia	tr
Europe	ru
Asia	ru

Dann würde die **JOIN**-Bedingung der obigen Aufgabe die folgende Zwischentabelle liefern:

e1.Continent	e1.Country	e2.Continent	e2.Country
Europe	tr	Europe	tr
Europe	tr	Europe	ru
Asia	tr	Asia	tr
Asia	tr	Asia	ru
Europe	ru	Europe	ru
Europe	ru	Europe	tr
Asia	ru	Asia	ru
Asia	ru	Asia	tr

Nun eliminiert die zusätzliche Bedingung **e1.country < e2.country** einige Zeilen und die Zwischentabelle schrumpft zu folgender Tabelle:

e1.Continent	e1.Country	e2.Continent	e2.Country
Europe	ru	Europe	tr
Asia	ru	Asia	tr

Das Ergebnis der Anfrage

```
SELECT e1.country, e2.country
FROM encompasses e1 JOIN encompasses e2 ON
        e1.continent = e2.continent
WHERE e1.country < e2.country
```

würde dann die folgende Tabelle sein:

e1.Country	e2.Country
ru	tr
ru	tr

Offensichtlich muss die obige Anfrage also mit **DISTINCT** erweitert werden, um solche Duplikate auszuschließen:

```
SELECT DISTINCT e1.country, e2.country
FROM encompasses e1 JOIN encompasses e2 ON
                    e1.continent = e2.continent
WHERE e1.country < e2.country
```

Frage 48: Bestimme für jedes Land die Anzahl der Sprachen, die in dem Land gesprochen wird.

```
SELECT c.code, COUNT(*)
FROM country c, language l
WHERE c.code = l.country
GROUP BY c.code
```

Frage 49: Geben Sie für jedes Land den Namen des Landes, die Fläche des Landes und die Anzahl der Organisationen in denen das Land mitglied ist aus!

```
SELECT c.name, c.area, COUNT(*)
FROM country c, is_member i
WHERE c.code = i.country
GROUP BY c.code,c.name,c.area
```

Da in der SELECT-Zeile die Spalten *name* und *area* mit angegeben werden sollen, **müssen** in dieser Lösung die Spalten *name* und *area* auch in der GROUP-BY-Klausel erscheinen. D.h. hier werden nach dem JOIN zwischen *Country* und *Is_member* nur solche **Gruppen** gebildet, die in den Spalten ***code***, *name* und *area* **denselben Wert** besitzen. Die Spalte *code* wird dabei zur eindeutigen Identifizierung einer (Zeile) Gruppe verwendet. Die folgende Tabelle verdeutlicht nochmal diesen Sachverhalt:

Country:

Name	Code	Area
Germany	de	357021
France	fr	547030
Afghanistan	af	647500

Is_member:

Country	Orgaization
de	EU
fr	EU
af	WHO
de	WHO
de	NATO

Das Datenbankmanagementsystem erzeugt aufgrund der angegebenen FROM-Zeile aus den beiden ursprünglichen Tabellen die folgende temporäre JOIN-Tabelle:

	Name	Code	Area	Country	Organization
1.	Germany	de	357021	de	EU
2.	Germany	de	357021	de	WHO
3.	Germany	de	357021	de	NATO
4.	France	fr	547030	fr	EU
5.	Afghanistan	af	647500	af	WHO

Jetzt werden mithilfe der GROUP-BY-Klausel die ersten drei Zeilen zu einer Gruppe, die vierte Zeile zu einer Gruppe und die fünfte Zeile zu einer Gruppe zusammengefasst. Man erkennt, dass alle Gruppen in den Spalten *code*, *name* und *area* denselben Wert haben (müssen).

Zusatzfrage:

Warum ist es nach der Erkenntnis dieser Aufgabe (in der GROUP-BY-Klausel dürfen mehrere Spalten stehen) nicht möglich, **Frage 47** wie folgt zu lösen:

```
SELECT g.country, m.name, max(m.height)
FROM geo_mountain g, mountain m
WHERE g.mountain = m.name
GROUP BY g.country, m.name
```

Frage 50: Welche Länder befinden sich mit Russland auf einem Kontinent?

```
SELECT country
FROM encompasses
WHERE continent IN (SELECT continent
                    FROM encompasses
                    WHERE country = 'ru')
```

Alternative Lösung **ohne IN**:

```
SELECT e.country
FROM encompasses e, encompasses e2
WHERE e2.country = 'ru' AND
      e.continent = e2.continent;
```

Frage 51: Welche Länder befinden sich nicht mit Russland auf einem Kontinent?

```
SELECT country
FROM encompasses
WHERE continent NOT IN (SELECT continent
                        FROM encompasses
                        WHERE country = 'ru')
```

Frage 52: Welche Länder liegen **nicht** auf dem europäischen Kontinent?

Falsche Lösung:

```
SELECT country
FROM encompasses
WHERE NOT continent = 'Europe'
```

Länder wie Russland oder Türkei werden in dieser Lösung fälschlicherweise ausgegeben, weil sie sowohl auf dem europäischen Kontinent, als auch auf einem nichteuropäischen Kontinent liegen.

Bessere Lösung:

```
SELECT code
FROM country
WHERE code NOT IN (SELECT country
                   FROM encompasses
                   WHERE continent = 'Europe')
```

Frage 53: Geben Sie zu Frage 50 und 51 die vollständigen Namen der Länder aus!

```
SELECT c.name
FROM encompasses e, country c
WHERE e.country = c.code AND
      continent IN (SELECT continent
                    FROM encompasses
                    WHERE country = 'ru')
```

Inverse Abfrage:

```
SELECT c.name
FROM encompasses e, country c
WHERE e.country = c.code AND
      continent NOT IN (SELECT continent
                        FROM encompasses
                        WHERE country = 'ru')
```

Frage 54: Welche Organisationen (siehe Tabelle *Organization*) haben keine Mitgliedsländer?

```
SELECT name
FROM organization
WHERE abbreviation NOT IN (SELECT organization
                           FROM is_member)
```

Frage 55: An welche Meeren grenzen keine Länder?

```
SELECT name
FROM sea
WHERE name NOT IN (SELECT sea
                   FROM geo_sea)
```

Frage 56: Welche Länder liegen am Mittelmeer?

```
SELECT name
FROM country
WHERE code IN (SELECT country
               FROM geo_sea
               WHERE sea='Mediterranean Sea')
```

Alternative Lösung:

```
SELECT DISTINCT name
FROM country, geo_sea
WHERE country.code = geo_sea.country AND
      geo_sea.sea =  'Mediterranean Sea'
```

Frage 57: Geben Sie **von jeder** Organisation (d. h. Berücksichtigen auch solche Organisationen die keine Mitgliedsländer haben) die Summe der Einwohner aller Mitgliedsländer absteigend geordnet an (lassen Sie die verschiedenen Arten von Mitgliedschaften unberücksichtigt).

```
SELECT organization, SUM(population) AS Sumpop
FROM Is_member i, country c
WHERE i.country = c.code
GROUP BY organization
UNION
SELECT abbreviation, 0 AS Sumpop
FROM Organization
WHERE abbreviation NOT IN (SELECT organization
                           FROM IS_member)
ORDER BY 2
```

UNION ist ein so genannter Mengen-Operator. Er vereinigt die Lösung von zwei Anfragen zu einer Anfrage. Voraussetzung ist, dass beide Anfragen

dieselbe Anzahl an Spalten mit denselben Datentypen besitzen.

Neben UNION gibt es noch den Durchschnitts-Operator **INTERSECT** und den Differenz-Operator **MINUS**. Mehr zu diesen Operatoren siehe Frage 62!

Frage 58: Bestimme diejenigen asiatischen Länder, deren Flächeninhalt in Asien kleiner ist als der Anteil der Türkei in Asien!

```
SELECT name
FROM country JOIN encompasses ON country = code
WHERE continent = 'Asia' AND
            (area * percentage/100) <
                  (SELECT area * (SELECT percentage
                                  FROM encompasses
                                  WHERE country = 'tr'  AND
                                    continent='Asia')/100
                   FROM country
                   WHERE code='tr')
```

Frage 59: Zu welchen Ländern ist genau ein Kontient bekannt?

```
SELECT c.name
 FROM country c
 WHERE UNIQUE (SELECT e.country
                FROM encompasses e
                WHERE c.country = e.country)
```

UNIQUE bedeutet hier, dass das Ergebnis der Unterabfrage genau einen Wert zurück liefern **muss**! Ein Land wird in die Ergebnistabelle aufgenommen, wenn die Unterabfrage für dieses Land genau einen Wert zurück liefert. Die Verbindung zwischen der übergeordneten Abfrage und der Unterabfrage geschieht über die Variable **c**. D.h. In Fällen in denen Unter- und Überabfragen miteinander verknüpft werden, müssen für die entsprechenden Tabellen Kurzbezeichner definiert werden.

Die Abfrage hat im Gegensatz zu den vorherigen Abfragen einen entscheidenden Unterschied. Hier wird für **jedes** betrachtete Land c.name die Unterabfrage einmal ausgewertet. In den obigen Beispielen wurden die Unterabfragen insgesamt nur einmal ausgewertet. Warum?
Alternative Lösung:

```
SELECT name
FROM country, encompasses
WHERE code = country
GROUP BY code, name
HAVING COUNT(*) = 1
```

Frage 60: Welche Länder haben eine größere Fläche als **mindestens ein** anderes Land in Europa?

```
SELECT c1.name
FROM country c1
WHERE c1.area > ANY (SELECT c2.area
                     FROM country c2, encompasses e
                     WHERE c2.code = e.country
                     AND e.continent = 'Europe')
```

Die WHERE-Bedinung ist jedesmal erfüllt, wenn in der Unterabfrage **mindestens eine** Fläche gefunden wurde, die kleiner als c1.area ist!

Alternative Lösung:

```
SELECT c1.name
FROM country c1
WHERE EXISTS (SELECT c2.name
              FROM country c2, encompasses e
              WHERE c2.code = e.country      AND
                    e.continent = 'Europe'   AND
                    c2.area < c1.area)
```

Es werde somit die Namen aller Länder aus c1 ausgegeben für die die Auswertung der Unterabfrage **mindestens ein Element** enthält, also **nicht leer** ist!

Die Unterabfrage gibt die Namen aller Länder aus deren Fläche kleiner als die Fläche des aktuell betrachteten Landes in c1 ist.

Frage 61: Welche Länder haben eine größere Fläche als **alle** anderen Länder in Europa?

```
SELECT c.name, c.area
FROM country c
WHERE c.area > ALL (SELECT c1.area
                    FROM country c1, encompasses e
                    WHERE c1.code = e.country
                    AND e.continent = 'Europe' AND
                        c.code != c1.code)
```

Die WHERE-Bedinung ist jedesmal erfüllt, wenn in der Unterabfrage **alle** Flächen kleiner als c.area sind!

Alternative Lösung:

```
SELECT c.name, c.area
FROM country c
WHERE c.area IN (SELECT MAX(c1.area)
                 FROM country c1, encompasses e
                 WHERE c1.code = e.country     AND
                       e.continent = 'Europe' )
```

Würde man in dieser statt **IN** lieber „=“ verwenden, dann darf es keine zwei Länder geben, die beide dieselbe (größte) Fläche besitzen.

Alternative Lösung:

```
SELECT c.name, c.area
FROM country c
WHERE NOT EXISTS (SELECT c1.name
                  FROM country c1, encompasses e
                  WHERE c1.code = e.country       AND
                        e.continent = 'Europe'    AND
                        NOT c1.code = c.code      AND
                            c1.area > c.area)
```

Es wird jedes Land aus c1 ausgegeben, für das die Unterabfrage **kein Ergebnis** liefert. D. h., dass es kein anderes Land in Europa existiert, dessen Fläche größer als die Fläche von dem aktuell betrachteten Land aus c1 ist.

Frage 62: Geben Sie den Namen und das Gründungsdatum der jüngsten Organisation an.

```
SELECT o.Name, o.established AS Gründungsjahr
FROM organization o
WHERE  o.established IS NOT NULL AND
       o.established > ALL (SELECT  o2.established
                            FROM organization o2
                            WHERE o2.established IS NOT NULL
                              AND o2.name !=o.name)
```

Alternative Lösung:

```
SELECT o.Name, o.established AS Gründungsjahr
FROM organization o
WHERE  o.established IS NOT NULL AND
NOT EXISTS (SELECT *
            FROM organization o2
            WHERE o2.Name != o.Name AND
                  o2.established > o.established)
```

Frage 63: Berechne die Anzahl der Menschen aller Länder, die in der größten Stadt ihres Landes leben!

```
SELECT SUM(GroßStädter)
FROM (SELECT country, MAX(population) AS GroßStädter
        FROM city
      GROUP BY country)
```

Eine Verschachtelung von Aggregierungsfunktion wie etwa **SUM(MAX(**Population**))** ist bisher **nicht erlaubt**!

Frage 64: Liste alle Namen, die für Städte und Länder verwendet werden, auf.

```
SELECT name
FROM city
UNION
SELECT name
FROM country
```

Die SELECT-Ausgaben in den beiden Teilanfragen müssen dieselbe Anzahl von Spalten mit demselben Datentyp haben! Im obigen Beispiel ist Name sowohl in der Tabelle *city*, als auch in der Tabelle *country* vom Typ VARCHAR.

Neben dem Vereinigungs-Operator stellt der SQL-Standard auch den Durchschitts-Operator **INTERSECT** und den Differenz-Operator **MINUS** zu Verfügung. Mengen-Operatoren eliminieren standardmäßig Duplikate. Möchte man Duplikate in der Ausgabe sehen, dann verwendet man zum Beispiel statt **UNION** den **UNION ALL** – Operator, bzw. den **INTERSECT ALL** - oder den **MINUS ALL** – Operator.

Frage 65: Bestimme alle Berge die sowohl in Frankreich, als auch in Deutschland liegen.

```
SELECT mountain
FROM geo_mountain
WHERE country='fr'
INTERSECT
SELECT mountain
FROM geo_mountain
WHERE country='de'
```

Frage 66: Bestimmen Sie alle Organisationen, die auf jedem Kontinent mindestens ein Mitgliedsland haben!

```
SELECT DISTINCT i.organization
FROM is_member i
WHERE
NOT EXISTS ((SELECT DISTINCT name
             FROM continent)
             MINUS
            (SELECT DISTINCT e.continent
             FROM encompasses e, is_member i2
            WHERE e.country = i2.country  AND
                  i2.organization = i.organization))
```

Die Unterabfrage ermittelt für jede Organisation der übergeordeten Anfrage die Kontinente der jeweiligen Länder.

Beispiel {'Europe','Asia'} **MINUS** {'Europe','Asia'} liefert die leere Menge!

Frage 67: Geben Sie alle Paare von europäischen Ländern aus, die an denselben Meeren liegen.

```
SELECT c1.name, c2.name
FROM country c1, country c2, encompasses e1, encompasses e2
WHERE c1.code = e1.country     AND
      c2.code = e2.country     AND
      e1.continent = 'Europe'  AND
      e2.continent = 'Europe'  AND
      c1.name < c2.name        AND
      c1.code IN (SELECT country FROM geo_sea) AND
      c2.code IN (SELECT country FROM geo_sea) AND
      NOT EXISTS(((SELECT g1.sea
                   FROM geo_Sea g1, encompasses e3
                   WHERE g1.country = e3.country     AND
                         e3.continent = 'Europe'     AND
                         e3.country   = c1.code)
                  UNION
                  (SELECT g2.sea
                   FROM geo_Sea g2, encompasses e4
                   WHERE g2.country = e4.country     AND
                         e4.continent = 'Europe'     AND
                         e4.country   = c2.code))
                  MINUS
                  ((SELECT g1.sea
                    FROM geo_Sea g1, encompasses e3
                    WHERE g1.country = e3.country    AND
                          e3.continent = 'Europe'    AND
                          e3.country   = c1.code)
                   INTERSECT
                   (SELECT g2.sea
                    FROM geo_Sea g2, encompasses e4
                    WHERE g2.country = e4.country    AND
                          e4.continent = 'Europe'    AND
                          e4.country   = c2.code)))
```

Einen Gleichheitsoperator für Mengen gibt es in SQL nicht! Zwei Mengen sind gleich, wenn ihre Vereinigung minus ihrem Schnitt leer ist!

Frage 68: Geben Sie für jede Organisation die Gesamtlänge ihrer Außengrenzen an.

```
SELECT o.name, SUM(b.Length) AS Länge
FROM organization o, borders b
WHERE
  EXISTS (SELECT *
            FROM is_member
            WHERE country = b.country1  AND
                  organization = o.abbreviation) AND
  NOT EXISTS(SELECT *
                FROM is_member
                WHERE country = b.country2 AND
                      organization = o.abbreviation)  OR
        EXISTS(SELECT *
                FROM is_member
                WHERE country = b.country2 AND
                      organization = o.abbreviation) AND
  NOT EXISTS(SELECT *
                FROM is_member
                WHERE country = b.country1 AND
                     organization = o.abbreviation)
GROUP BY o.name
UNION
SELECT o.name, 0 AS Länge
FROM organization o
WHERE NOT EXISTS (SELECT *
                      FROM  is_member
                      WHERE organization = o.abbreviation)
```