Themen

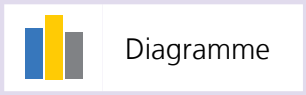
Daten graphisch darstellen

Daten lassen sich in Tabellen kompakt und übersichtlich zusammenfassen. Doch umfangreiche Tabellen sind mühsam zu lesen und häufig ist es schwer, Zusammenhänge zwischen den Werten zu erkennen.

Diagramme helfen, die Tabellendaten anschaulich darzustellen, so dass Verläufe, Trends und Größenvergleiche leichter zu erfassen sind.

Tabellenkalkulations-Software unterstützt die Darstellung der Tabellendaten in Diagrammform durch vordefinierte Diagrammtypen.

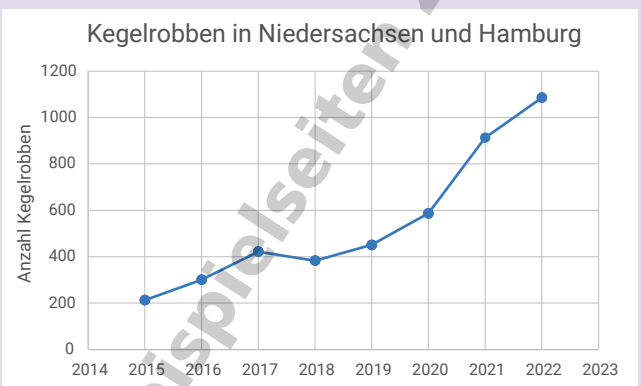
Im Menü „Einfügen“ gelangt man – je nach Software – über eines dieser Symbole zu einem Menü, das bei der Auswahl eines geeigneten Diagrammtyps hilft.



Jahr	Kegelrobben [1]
2015	213
2016	301
2017	422
2018	383
2019	451
2020	587
2021	913
2022	1086

Für die Darstellung der Entwicklung des Kegelrobben-Bestandes im Wattenmeer Niedersachsens und Hamburgs bietet sich beispielsweise ein XY-Diagramm an, das auch Streu- oder Punktdiagramm genannt wird.

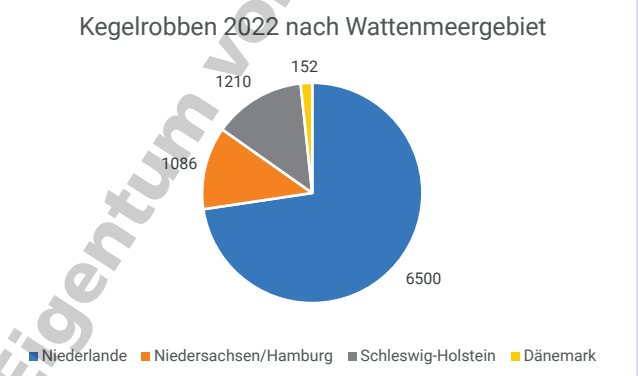
Im Diagramm-Assistenten oder über dieses Symbol lassen sich weitere Diagrammelemente wie eine Legende, Achsenbeschriftungen oder eine Überschrift zum Diagramm hinzufügen.



Das Vergleichen unterschiedlicher Mengen gelingt am anschaulichsten mit Kreisdiagrammen.

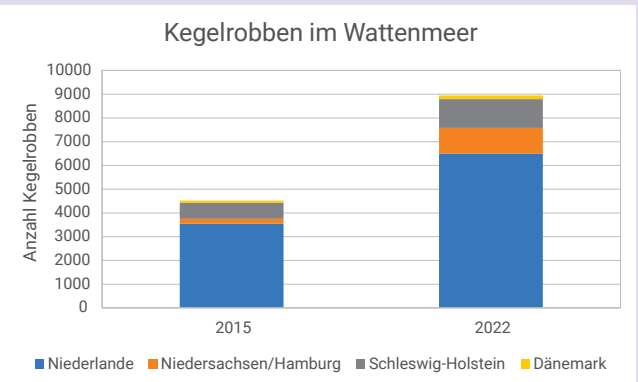
Wattenmeergebiet	Kegelrobben [1] 2015	2022
Niederlande	3544	6500
Niedersachsen/Hamburg	213	1086
Schleswig-Holstein	676	1210
Dänemark	88	152

Dieses Kreisdiagramm zeigt beispielsweise auf Anhieb, dass fast drei Viertel der Kegelrobben im niederländischen Teil des Wattenmeeres leben.



In Kreisdiagrammen sind die Daten immer als Teile eines Ganzen, also in prozentualer Verteilung dargestellt. Es eignet sich daher nicht, um die in unterschiedlichen Jahren im Wattenmeer gezählten Kegelrobben miteinander zu vergleichen.

So genannte gestapelte Säulendiagramme zeigen sowohl die Mengenverteilung innerhalb jeder Säule als auch den Vergleich der Gesamtmengen anhand der Säulenhöhe. Sie bieten sich daher an, die 2015 und 2022 im Wattenmeer gezählten Kegelrobben miteinander zu vergleichen.



[1] Monitoring-Ergebnisse der Kegelrobbenzählungen im Wattenmeer in Niedersachsen und Hamburg https://www.waddensea-secretariat.org/de/seehunde (Stand April 2023)

Daten graphisch darstellen

Aufgabe 1

Öffne die Datei Temperatur_Hameln_02-2023.xlsx [1].

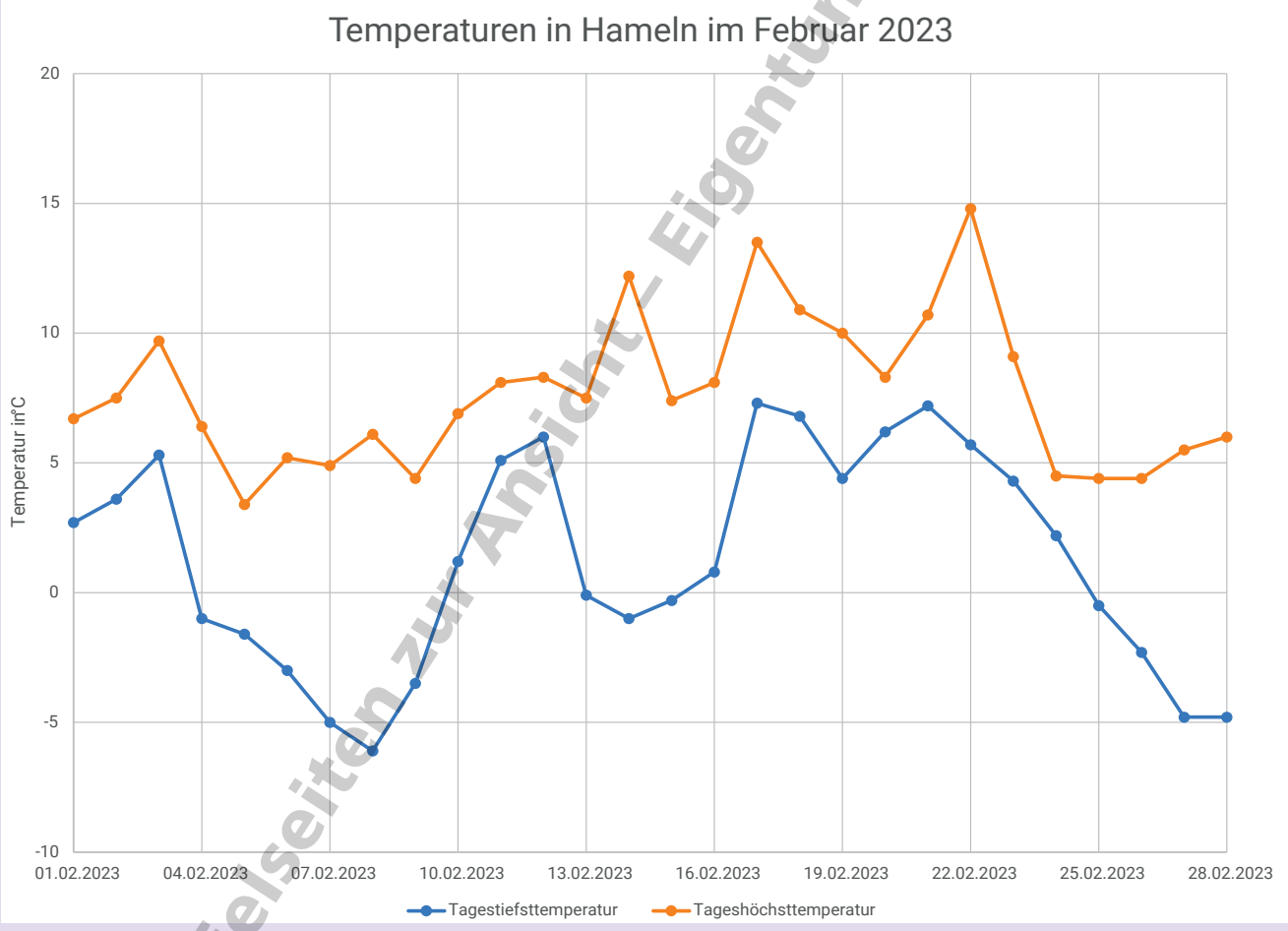
a) Erstelle ein XY-Diagramm, das die Tagestiefst- und die Tageshöchsttemperaturen für Hameln im Februar 2023 darstellt.

b) Füge eine Überschrift und einen Achsentitel an der Y-Achse ein.

Zusatzaufgaben

c) Formatiere die Y-Achse so, dass die horizontale Achse sie bei –10 °C schneidet.

d) Formatiere die X-Achse so, dass sie mit dem 1. Februar beginnt und mit dem 28. Februar endet.
Beachte dabei, dass Microsoft Excel anstelle des Datums den so genannten DATWERT anzeigt. Das ist eine Funktion, die vom 1. Januar 1900 (DATWERT = 1) ausgehend fortlaufend die Tage zählt. Der 1. Februar 2023 hat den DATWERT 44958 und der 28. Februar den DATWERT 44985.



Datenquellen:

[1] https://meteostat.net/de/place/de/hameln?s=D3675&t=2023-02-01/2023-02-28 (Stand April 2023)

[2] https://www1.nls.niedersachsen.de/statistik/html/default.asp; 12411 - Fortschreibung des Bevölkerungsstandes; Bevölkerung nach Altersgruppen (23) und Geschlecht (Gemeinde) (Stand Februar 2023)

[3] Flächenerhebung nach Art der tatsächlichen Nutzung, https://www.statistik.niedersachsen.de/flaechenerhebung/flachenerhebung-nach-art-der-tatsachlichen-nutzung-statistische-berichte-87671.html (Stand April 2023)

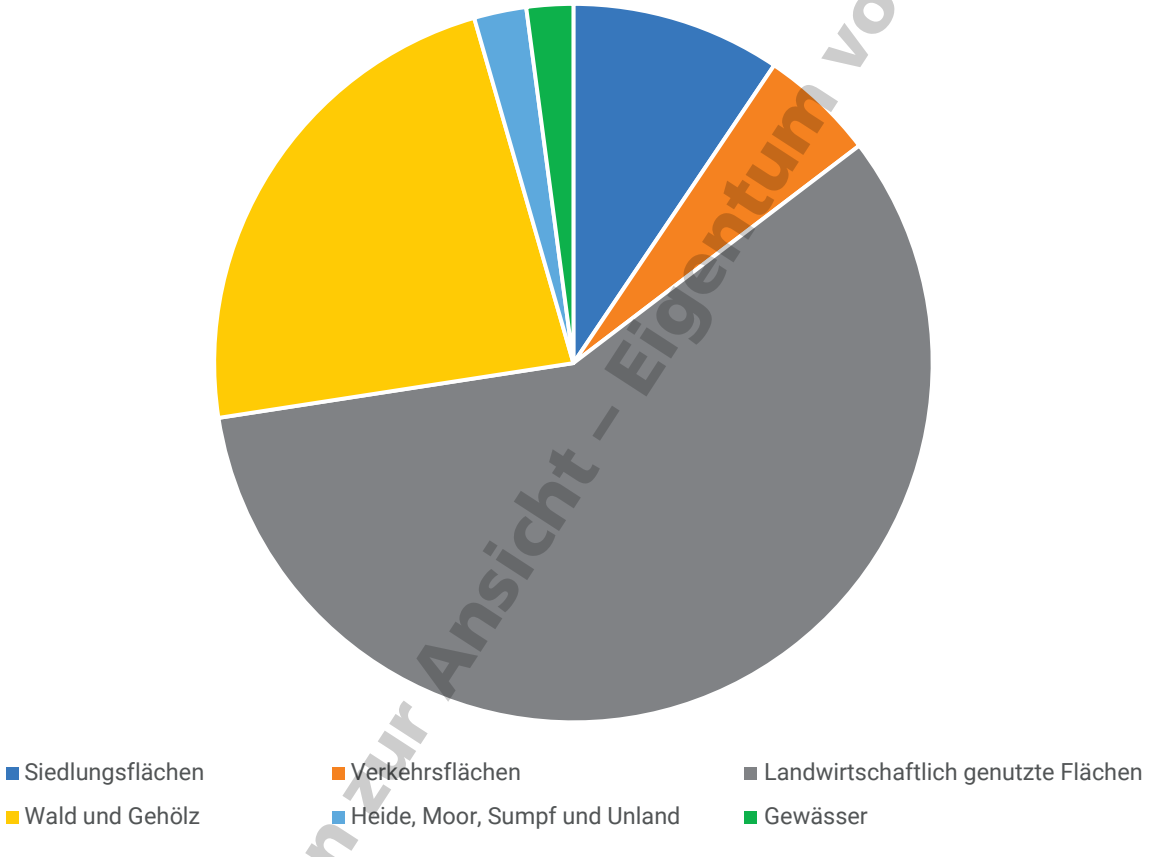
Daten graphisch darstellen

Aufgabe 2

Öffne die Datei Flaechennutzung_NI.xlsx [3], die Daten zur Nutzung von Flächen in Niedersachsen enthält.

a) Erstelle ein Kreisdiagramm für die Flächennutzung in Niedersachsen.

b) Füge eine Überschrift und eine Legende hinzu.



Datenquellen:

[1] https://meteostat.net/de/place/de/hameln?s=D3675&t=2023-02-01/2023-02-28 (Stand April 2023)

[2] https://www1.nls.niedersachsen.de/statistik/html/default.asp; 12411 - Fortschreibung des Bevölkerungsstandes; Bevölkerung nach Altersgruppen (23) und Geschlecht (Gemeinde) (Stand Februar 2023)

[3] Flächenerhebung nach Art der tatsächlichen Nutzung, https://www.statistik.niedersachsen.de/flaechenerhebung/flachenerhebung-nach-art-der-tatsachlichen-nutzung-statistische-berichte-87671.html (Stand April 2023)

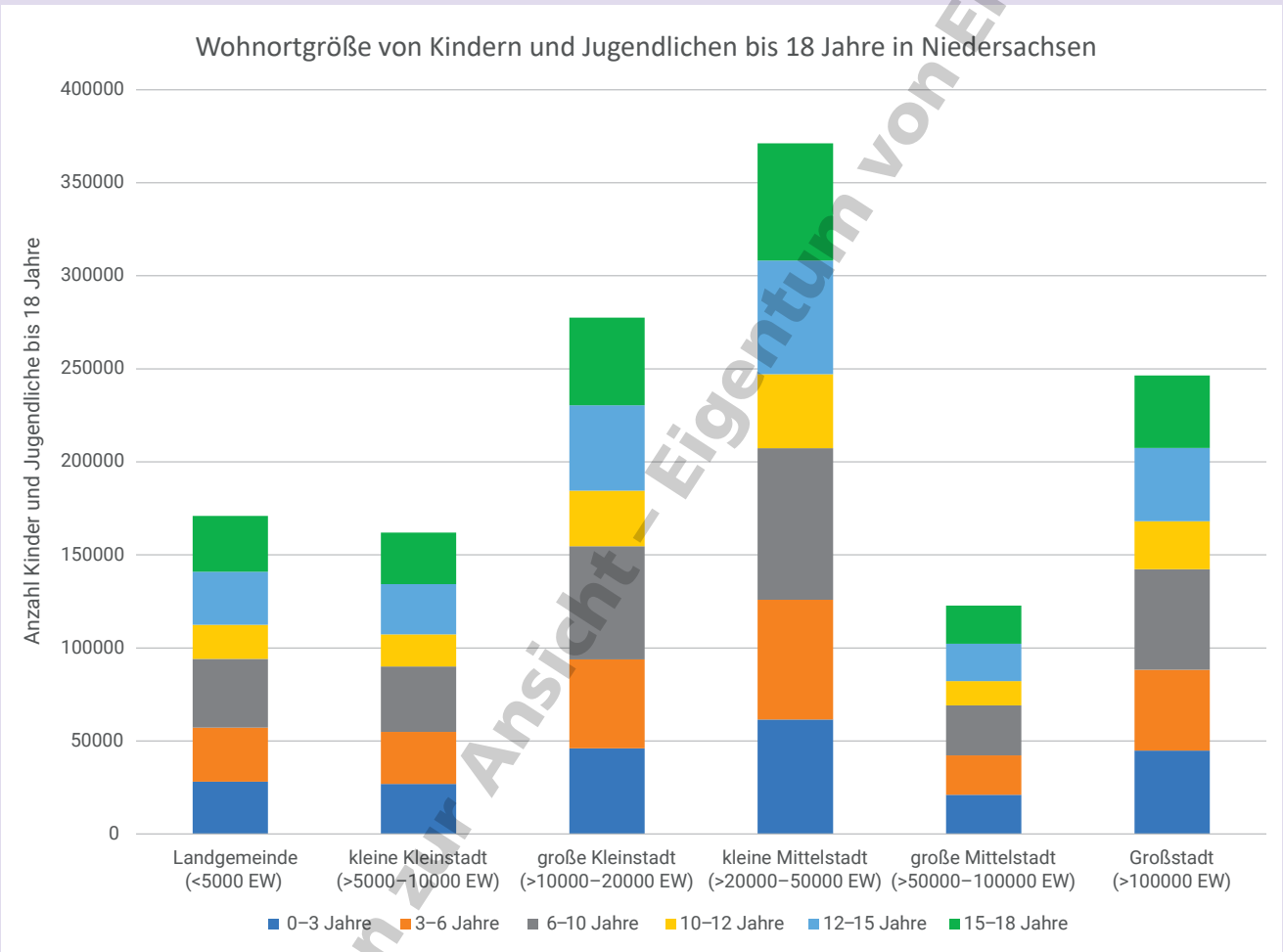
Daten graphisch darstellen

Aufgabe 3

Öffne die Datei Wohnortgroesse.xlsx [2], die Daten zur Größe der Wohnorte von Kindern und Jugendlichen bis 18 Jahren in Niedersachsen enthält.

a) Erstelle ein gestapeltes Balkendiagramm, in dem für jede Wohnortgröße die Anzahl der dort lebenden Jugendlichen in den Altersgruppen aufgetragen ist.

b) Füge eine Überschrift und einen Achsentitel an der vertikalen Achse ein.



Datenquellen:

[1] https://meteostat.net/de/place/de/hameln?s=D3675&t=2023-02-01/2023-02-28 (Stand April 2023)

[2] https://www1.nls.niedersachsen.de/statistik/html/default.asp; 12411 - Fortschreibung des Bevölkerungsstandes; Bevölkerung nach Altersgruppen (23) und Geschlecht (Gemeinde) (Stand Februar 2023)

[3] Flächenerhebung nach Art der tatsächlichen Nutzung, https://www.statistik.niedersachsen.de/flaechenerhebung/flachenerhebung-nach-art-der-tatsachlichen-nutzung-statistische-berichte-87671.html (Stand April 2023)

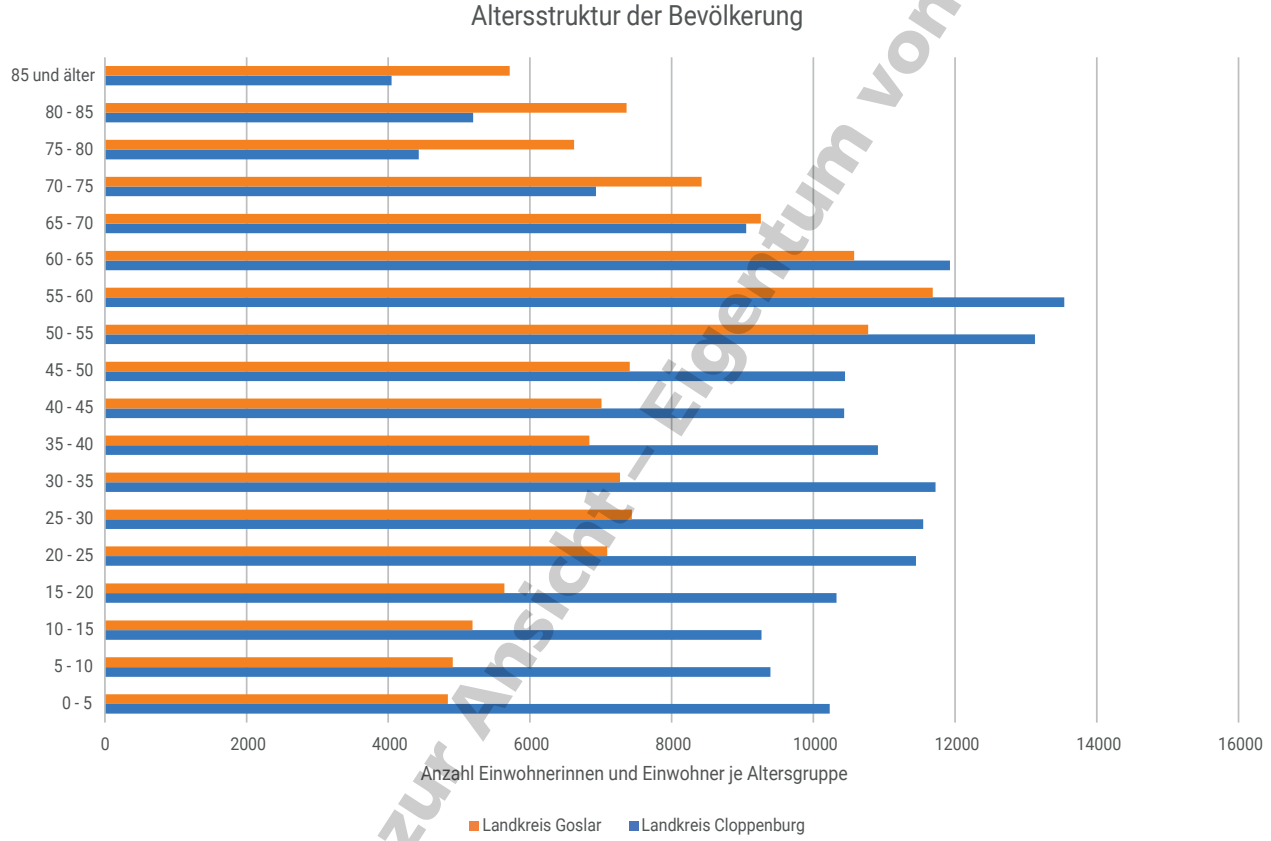
Daten graphisch darstellen

Aufgabe 4

Öffne die Datei Altersstruktur_CLP_GS.xlsx [2], die Daten zur Altersstruktur der Bevölkerung der Landkreise Cloppenburg und Goslar enthält.

a) Erstelle ein Balkendiagramm, das die Altersstruktur beider Landkreise enthält.

b) Füge eine Überschrift und einen Achsentitel an der horizontalen Achse ein.



Datenquellen:

[1] https://meteostat.net/de/place/de/hameln?s=D3675&t=2023-02-01/2023-02-28 (Stand April 2023)

[2] https://www1.nls.niedersachsen.de/statistik/html/default.asp; 12411 - Fortschreibung des Bevölkerungsstandes; Bevölkerung nach Altersgruppen (23) und Geschlecht (Gemeinde) (Stand Februar 2023)

[3] Flächenerhebung nach Art der tatsächlichen Nutzung, https://www.statistik.niedersachsen.de/flaechenerhebung/flachenerhebung-nach-art-der-tatsachlichen-nutzung-statistische-berichte-87671.html (Stand April 2023)

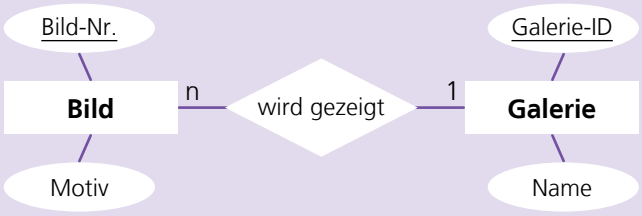
Relationales Datenbankschema

Das relationale Datenbankschema ist ein häufig eingesetztes Datenbankschema, das auch von den populärsten Datenbankmanagementsystemen Oracle und MySQL verwendet wird. [1)]

Das Wort Relation geht auf das lateinische relatio zurück, das u.a. für Beziehung oder Verhältnis steht. Ein relationales Datenbankschema beruht auf Tabellen mit Eigenschaften, die zueinander in Beziehung stehen.

Vom ER-Diagramm zum Datenbankschema

Zur Veranschaulichung greifen wir wieder auf unser Beispiel aus der vorherigen Lektion zurück. Dieses einfache ER-Diagramm besteht aus zwei Entitätstypen mit jeweils einer Eigenschaft (Motiv, Name), die durch eine 1:n-Beziehung miteinander verbunden sind.



Im Datenbankschema wird für jeden Entitätstyp eine Tabelle erstellt. Die Tabellen haben jeweils eine Spalte für die Eigenschaften Motiv bzw. Name. Zusätzlich werden eine Spalte mit einer Bild-Nummer und eine Spalte mit einer Galerie-ID eingefügt. Sie dienen als so genannter Primärschlüssel und ermöglichen eine eindeutige Unterscheidung zweier Bilder auch dann, wenn die Motive gleich benannt sind. Im ER-Diagramm werden die Primärschlüssel durch Unterstreichen gekennzeichnet.

Bild

Bild-Nr.	Motiv
023	Rafting
286	Mondlicht
541	Skyline
881	Alte Brücke

Galerie

Galerie-ID	Name
gal_14	Galerie 14
kun_hof	Kunst im Hof
par_gal	Park-Galerie

Die Beziehung zwischen den beiden Tabellen wird hergestellt, indem der Primärschlüssel einer Tabelle als so genannter Fremdschlüssel in die andere Tabelle aufgenommen wird. Dafür haben wir in unserem Beispiel theoretisch zwei Möglichkeiten: die Bild-Nr. wird in die Tabelle Galerie eingefügt oder die Galerie-ID wird in die Tabelle Bild eingefügt. Beginnen wir mit der ersten Möglichkeit.

Wenn in einer Galerie mehrere Bilder gezeigt werden, führt das Eintragen der Bild-Nummer in die Tabelle Galerie zwangsläufig zu einer Verdopplung einzelner Zeilen. Damit ist zum einen der Primärschlüssel „Galerie-ID“ nicht mehr eindeutig und zum anderen wird der Name der Galerien wiederholt. Eine solche Wiederholung nennt man auch Redundanz.

Bild

Bild-Nr.	Motiv
023	Rafting
286	Mondlicht
541	Skyline
881	Alte Brücke

Galerie

Galerie-ID	Name	Bild-Nr.
gal_14	Galerie 14	541
gal_14	Galerie 14	881
kun_hof	Kunst im Hof	286
par_gal	Park-Galerie	023

Redundanzen in Datenbanken sind unerwünscht, weil sie die Konsistenz der Daten gefährden, also nicht mehr sichergestellt ist, dass die Daten korrekt, einheitlich und aktuell sind. Wenn beispielsweise in einer Zeile die „Galerie 14“ in „Galerie 144“ umbenannt würde, wäre nicht mehr klar, wie die Galerie tatsächlich heißt.

Der Fremdschlüssel wird daher bei einer **1:n-Beziehung** stets in die Tabelle des Entitätstyps eingefügt, bei dem im ER-Diagramm das n steht. Nur dann führt das Einfügen nicht zu einer unzulässigen Verdoppelung einzelner Primärschlüssel.

Bild

Bild-Nr.	Motiv	Galerie-ID
023	Rafting	par_gal
286	Mondlicht	kun_hof
541	Skyline	gal_14
881	Alte Brücke	gal_14

Galerie

Galerie-ID	Name
gal_14	Galerie 14
kun_hof	Kunst im Hof
par_gal	Park-Galerie

Für **n:m-Beziehungen** würde das Einfügen der Fremdschlüssel zwangsläufig zu Redundanzen in den Tabellen führen. Deshalb erstellt man hier noch eine weitere Tabelle, die nur die Beziehung zwischen den beiden Primärschlüsseln herstellt.

Künstler

Künstler-ID	Name
meye	Meyer
muel	Müller
schm	Schmidt

Galerie

Galerie-ID	Name
gal_14	Galerie 14
kun_hof	Kunst im Hof
par_gal	Park-Galerie

stellt aus

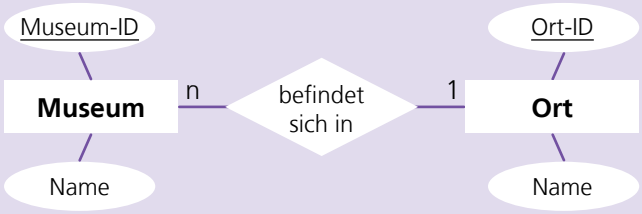
Künstler-ID	Galerie-ID
meye	gal_14
meye	kun_hof
muel	gal_14
muel	kun_hof
muel	par_gal
schm	gal_14
schm	par_gal

[1)] https://db-engines.com/de/ranking (Stand Juli 2024)

Relationales Datenbankschema

Aufgabe 1

Erstelle für das folgende ER-Diagramm alle für ein relationales Datenbankschema benötigten Tabellen mit den erforderlichen Primärschlüsseln und Fremdschlüsseln.



Befülle die Tabellen mit diesen Beispieldaten:

- Schloss Benrath in Düsseldorf
- Deutsches Fußballmuseum in Dortmund
- Museum Folkwang in Essen
- Museum Ludwig in Köln
- Schokoladenmuseum in Köln
- Zeche Zollverein in Essen

Beispiellösung

Museum

Museum-ID	Name	Ort-ID
benr	Schloss Benrath	duess
defu	Deutsches Fußballmuseum	dortm
folk	Museum Folkwang	essen
ludw	Museum Ludwig	koeln
scho	Schokoladenmuseum	koeln
zoll	Zeche Zollverein	essen

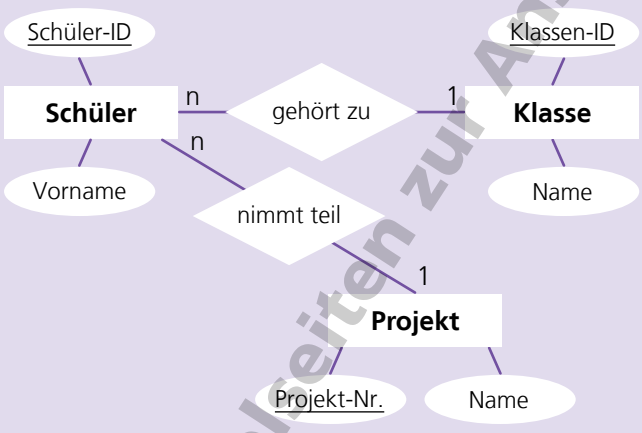
Ort

Ort-ID	Name
dortm	Dortmund
duess	Düsseldorf
essen	Essen
koeln	Köln

Aufgabe 2

Erstelle für das folgende ER-Diagramm alle für ein relationales Datenbankschema benötigten Tabellen mit den erforderlichen Primärschlüsseln und Fremdschlüsseln.

Befülle die Tabellen mit eigenen Beispieldaten.



Beispiellösung

Schüler

Schüler-ID	Vorname	Klassen-ID	Projekt-ID
1003	Linus	8a	24D
1213	Murat	8b	24C
1345	Nele	8c	24A
1387	Milan	8c	24B
1444	Marie	8b	24C
1502	Özlem	8a	24A

Klasse

Klassen-ID	Klasse
8a	Klasse 8 a
8b	Klasse 8 b
8c	Klasse 8 c

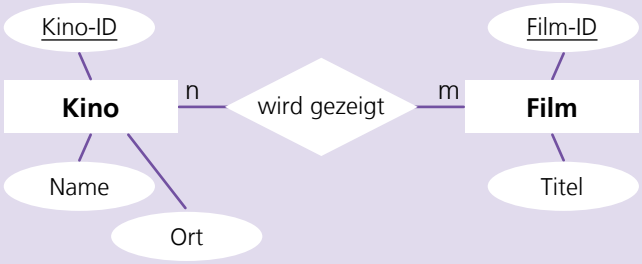
Projekt

Projekt-ID	Name
24A	Gebärdensprache lernen
24B	Müll sammeln am See
24C	Physik im Kindergarten
24D	Bauprojekt im Tierheim

Relationales Datenbankschema

Aufgabe 3

Erstelle für das folgende ER-Diagramm alle für ein relationales Datenbankschema benötigten Tabellen mit den erforderlichen Primärschlüsseln und Fremdschlüsseln.



Befülle die Tabellen mit diesen Beispieldaten:

Filme

- Ich war neunzehn
- Die Legende von Paul und Paula
- Das Boot
- Der Himmel über Berlin
- Gegen die Wand

Kinos

- Cinema, Düsseldorf
- Camera, Dortmund
- filmforum, Duisburg
- Lichtburg, Essen
- Metropolis, Köln

Beispiellösung

Kino

Kino-ID	Name	Ort
d-cin	Cinema	Düsseldorf
do-cam	Camera	Dortmund
du-fil	filmforum	Duisburg
e-lic	Lichtburg	Essen
k-met	Metropolis	Köln

Film

Film-ID	Titel
D097	Ich war neunzehn
D143	Die Legende von Paul und Paula
D384	Das Boot
D420	Der Himmel über Berlin
D523	Gegen die Wand

wird gezeigt

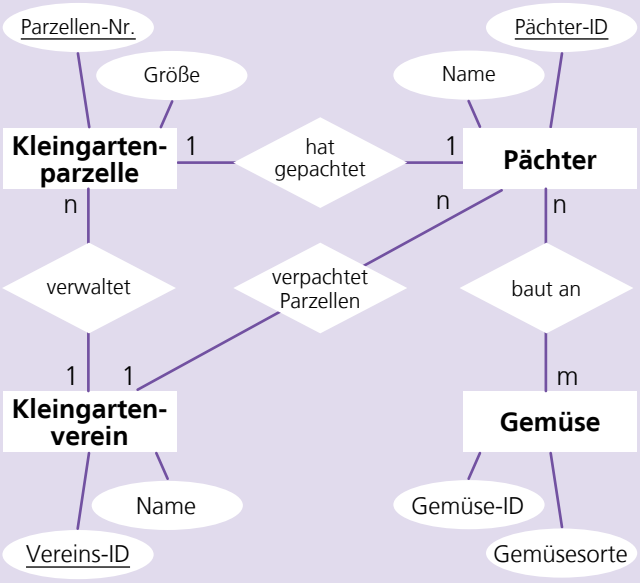
Kino-ID	Film-ID
d-cin	D143
d-cin	D523
do-cam	D384
do-cam	D420
du-fil	D097
du-fil	D143

Kino-ID	Film-ID
du-fil	D523
e-lic	D143
e-lic	D384
k-met	D097
k-met	D384
k-met	D420

Relationales Datenbankschema

Aufgabe 4

Erstelle für das folgende ER-Diagramm alle für ein relationales Datenbankschema benötigten Tabellen mit den erforderlichen Primärschlüsseln und Fremdschlüsseln.



Befülle die Tabellen mit diesen Beispieldaten:

Kleingartenvereine
- Am Mühlenberg
- Naturfreunde

Pächter
- Schubert
- Kowalski
- Yılmaz
- Otto

Gemüsesorte
- Blumenkohl
- Buschbohnen
- Feldsalat
- Kohlrabi
- Radieschen
- Zucchini

Kleingartenparzelle
- 370 m²
- 425 m²
- 300 m²
- 390 m²

Beispiellösung

Kleingartenparzelle

Parzellen-Nr.	Größe	Vereins-ID	Pächter-ID
mueh-025	370 m²	mueh	1785
mueh-103	425 m²	mueh	366
natu-001	300 m²	natu	3389
natu-251	390 m²	natu	2417

Pächter

Pächter-ID	Name	Vereins-ID
366	Schubert	mueh
1785	Kowalski	mueh
2417	Yilmaz	natu
3389	Otto	natu

Gemüse

Gemüse-ID	Gemüsesorte
blukoh	Blumenkohl
buschb	Buschbohnen
feldsa	Feldsalat
kohlra	Kohlrabi
radies	Radieschen
zucchi	Zucchini

Kleingartenverein

Vereins-ID	Name
mueh	Am Mühlenberg
natu	Naturfreunde

baut an

Pächter-ID	Gemüse-ID
366	buschb
366	feldsa
366	kohlra
366	radies
1785	blukoh
1785	feldsa
1785	zucchi
2417	buschb
2417	kohlra
2417	zucchi
3389	blukoh
3389	radies

Daten abfragen mit SQL

Für SQL wird häufig die nicht ganz korrekte Bezeichnung „Structured Query Language“ (deutsch: Strukturierte Abfrage-Sprache) verwendet. Das deutet darauf hin, dass das Abfragen von Informationen aus Datenbanken die vordringliche Anwendung für diese Sprache war und ist.

Anhand der beiden folgenden Tabellen zeigen wir die wichtigsten dafür benötigten SQL-Befehle.

Schiffe [1)]

schiff_id	name	container	reeder_id
1	Emma Maersk	17816	2
2	Ever Alot	24000	3
3	Madrid Maersk	20568	2
4	MSC Irina	24346	1
5	MSC Jade	19437	1

Reedereien

reeder_id	name	sitz
1	A. P. Moller-Maersk Group	Kopenhagen
2	Evergreen Marine Corp. (Taiwan) Ltd.	Taipeh
3	Mediterranean Shipping Company	Genf

Der wichtigste Befehl für SQL-Abfragen lautet `SELECT ... FROM`. Wird er mit einem Sternchen verwendet, liefert die Anweisung den gesamten Inhalt der entsprechenden Tabelle. Werden Spaltennamen in die Anweisung eingefügt, erhält man den Inhalt der genannten Spalten.

```
SELECT * FROM Schiffe;
```

```
SELECT name, container FROM Schiffe;
```

Auch bei Abfragen kann man Bedingungen mit `WHERE` verwenden, um das Ergebnis einzuschränken. Dabei können die Vergleichsoperatoren `<`, `>`, `=` oder der `BETWEEN`-Operator eingesetzt werden.

```
SELECT name, container FROM Schiffe
WHERE container < 20000;
```

```
SELECT name, container FROM Schiffe
WHERE container BETWEEN 20000 AND 25000
ORDER BY container ASC;
```

Mit dem SQL-Befehl `ORDER BY` bewirkt man das Sortieren des Abfrageergebnisses. `ASC` führt zu einer aufsteigenden (a–z), `DESC` zu einer absteigenden (z–a) Sortierung.

Relationale Datenbanken können aus zahlreichen Tabellen bestehen, die miteinander in Beziehung stehen. Das gelingt, indem der Primärschlüssel einer Tabelle als Fremdschlüssel in eine andere Tabelle aufgenommen wird. In unseren Beispieltabellen ist die reeder_id dieser Fremdschlüssel, der die Tabellen Reedereien und Schiffe verknüpft.

Durch diese Beziehung ist es möglich, Daten aus mehreren Tabellen abzufragen.

```
SELECT Schiffe.name, Reedereien.name
FROM Schiffe
INNER JOIN Reedereien
ON Schiffe.reeder_id=Reedereien.reeder_id;
```

Sobald Daten aus mehreren Tabellen abgefragt werden, verwendet man die Spaltennamen mit vorangestelltem Tabellennamen, um Verwechslungen zu vermeiden (z. B. `Reedereien.name`).

Mit dem SQL-Befehl `INNER JOIN` wird angegeben, welche zweite Tabelle in die Abfrage aus unserer Tabelle Schiffe eingebunden werden soll. Nach dem Schlüsselwort `ON` stehen die Spaltennamen aus beiden Tabellen, die die Beziehung bilden. Die obige Anweisung liefert eine Tabelle mit den fünf Schiffen und den dazugehörenden Reedereien.

Durch das Anfügen von Bedingungen (`WHERE`) und den Befehl `ORDER BY` können auch diese Ergebnisse eingeschränkt und sortiert werden.

Der SQL-Befehl `COUNT` ermöglicht das Zählen von Zeilen in einer Tabelle, die einem Kriterium aus einer anderen Tabelle entsprechen. In unserem Beispiel werden die Schiffe gezählt, die zu einer Reederei gehören, und unter dem Spaltennamen anzahl_schiffe als Ergebnis ausgegeben.

```
SELECT Reedereien.name,
COUNT(Schiffe.reeder_id) AS anzahl_schiffe
FROM Schiffe
INNER JOIN Reedereien
ON Schiffe.reeder_id=Reedereien.reeder_id
GROUP BY Reedereien.name;
```

Das Zuordnen zu den jeweiligen Namen der Reedereien erfolgt mit Hilfe des SQL-Befehls `GROUP BY`.

[1)] https://de.wikipedia.org/wiki/Containerschiff (Stand 09/2024)

Daten abfragen mit SQL

Für die Bearbeitung der folgenden Aufgaben werden die befüllten Tabellen Bundesländer und Großstädte benötigt. Beide Tabellen müssen zunächst per SQL-Anweisungen im Programiz-Editor angelegt und befüllt werden.

Kopiere dafür nacheinander den Inhalt der Dateien SQL-Skript_Tabelle_bundeslaender.txt und SQL-Skript_Tabelle_grossstaedte.txt in den Editor und führe sie aus.

Aufgabe 1

Aus der Tabelle Grossstaedte sollen alle Städte herausgesucht werden, die mehr als 600000 Einwohner haben. Name und Einwohnerzahl der Städte sollen aufgelistet werden, absteigend nach der Einwohnerzahl sortiert.

Schreibe eine entsprechende SQL-Abfrage.

Beispiellösung

```
SELECT name, einwohner
FROM Grossstaedte
WHERE einwohner > 600000
ORDER BY einwohner DESC;
```

Output

name	einwohner
Berlin	3782202
Hamburg	1910160
München	1510378
Köln	1087353
Frankfurt am Main	775790
Stuttgart	633484
Düsseldorf	631217
Leipzig	619879

Aufgabe 2

Aus der Tabelle Grossstaedte sollen alle Städte herausgesucht werden, die weniger als 75 km² Fläche einnehmen. Name und Fläche der Städte sollen aufgelistet werden, aufsteigend nach der Fläche sortiert.

Schreibe eine entsprechende SQL-Abfrage.

Beispiellösung

```
SELECT name, flaeche
FROM Grossstaedte
WHERE flaeche < 75
ORDER BY flaeche ASC;
```

Output

name	flaeche
Offenbach am Main	44.88
Herne	51.42
Fürth	63.35
Recklinghausen	66.5
Moers	67.64
Remscheid	74.52

Daten abfragen mit SQL

Aufgabe 3

Aus der Tabelle Grossstaedte sollen alle Städte herausgesucht werden, die weniger als 110000 Einwohner haben. Name, Einwohner und Bundesland sollen aufgelistet werden, aufsteigend nach der Einwohnerzahl sortiert.

Schreibe eine entsprechende SQL-Abfrage. Nutze die land_id und den Befehl `INNER JOIN`.

Beispiellösung

```
SELECT Grossstaedte.name, Grossstaedte.einwohner, Bundeslaender.land
FROM Grossstaedte
INNER JOIN Bundeslaender ON Grossstaedte.land_id = Bundeslaender.land_id
WHERE Grossstaedte.einwohner < 110000
ORDER BY Grossstaedte.einwohner ASC;
```

Output

name	einwohner	land
Cottbus/Chósebuz	100010	Brandenburg
Kaiserslautern	101486	Rheinland-Pfalz
Siegen	102114	Nordrhein-Westfalen
Hildesheim	102325	Niedersachsen
Gütersloh	102464	Nordrhein-Westfalen
Hanau	103184	Hessen
Salzgitter	105039	Niedersachsen
Moers	105606	Nordrhein-Westfalen

Daten abfragen mit SQL

Aufgabe 4

Aus der Tabelle Grossstaedte sollen alle Städte herausgesucht werden, die im Bundesland Nordrhein-Westfalen liegen. Name, Einwohner und Fläche sollen aufgelistet werden, aufsteigend nach dem Namen der Großstadt sortiert.

Schreibe eine entsprechende SQL-Abfrage. Nutze die land_id und den Befehl `INNER JOIN`.

Beispiellösung

```
SELECT Grossstaedte.name, Grossstaedte.einwohner, Grossstaedte.flaeche
FROM Grossstaedte
INNER JOIN Bundeslaender ON Grossstaedte.land_id = Bundeslaender.land_id
WHERE Bundeslaender.land = "Nordrhein-Westfalen"
ORDER BY Grossstaedte.name;
```

Output

name	einwohner	flaeche
Aachen	252769	160.85
Bergisch Gladbach	112660	83.09
Bielefeld	338410	258.83
Bochum	366385	145.66
Bonn	335789	141.06
Bottrop	118705	100.61
Dortmund	595471	280.71
Duisburg	503707	232.84
Düsseldorf	631217	217.41
Essen	586608	210.34
Gelsenkirchen	265885	104.94
Gütersloh	102464	112.02
Hagen	190490	160.45
Hamm	180761	226.43
Herne	157896	51.42
Krefeld	228550	137.78
Köln	1087353	405.02
Leverkusen	166414	78.87
Moers	105606	67.64
Mönchengladbach	268943	170.47
Mülheim an der Ruhr	173255	91.28
Münster	322904	303.28
Neuss	155163	99.52
Oberhausen	211099	77.09
Paderborn	155749	179.59
Recklinghausen	111693	66.5
Remscheid	112970	74.52
Siegen	102114	114.69
Solingen	161545	89.54
Wuppertal	358938	168.39

Daten abfragen mit SQL

Aufgabe 5

Aus der Tabelle Grossstaedte sollen alle Städte herausgesucht werden, die zwischen 200000 und 250000 Einwohner haben. Name, Einwohner und Bundesland sollen aufgelistet werden, aufsteigend nach der Einwohnerzahl sortiert.

Schreibe eine entsprechende SQL-Abfrage. Nutze die land_id und den Befehl `INNER JOIN`.

Beispiellösung

```
SELECT Grossstaedte.name, Grossstaedte.einwohner, Bundeslaender.land
FROM Grossstaedte
INNER JOIN Bundeslaender ON Grossstaedte.land_id = Bundeslaender.land_id
WHERE Grossstaedte.einwohner BETWEEN 200000 AND 250000
ORDER BY Grossstaedte.einwohner ASC;
```

Output

name	einwohner	land
Kassel	204687	Hessen
Rostock	210795	Mecklenburg-Vorpommern
Oberhausen	211099	Nordrhein-Westfalen
Erfurt	215675	Thüringen
Lübeck	219044	Schleswig-Holstein
Mainz	222889	Rheinland-Pfalz
Krefeld	228550	Nordrhein-Westfalen
Freiburg im Breisgau	237244	Baden-Württemberg
Magdeburg	240114	Sachsen-Anhalt
Halle (Saale)	242172	Sachsen-Anhalt
Kiel	248873	Schleswig-Holstein

Daten abfragen mit SQL

Aufgabe 6

Aus der Tabelle Grossstaedte sollen alle Städte herausgesucht werden, die Hauptstadt eines Bundeslandes sind. Hauptstadt, Einwohner und Bundesland sollen aufgelistet werden, aufsteigend nach der Hauptstadt sortiert.

Schreibe eine entsprechende SQL-Abfrage. Nutze den Befehl `INNER JOIN`.

Beispiellösung

```
SELECT Bundeslaender.hauptstadt, Grossstaedte.einwohner, Bundeslaender.land
FROM Bundeslaender
INNER JOIN Grossstaedte ON Grossstaedte.name = Bundeslaender.hauptstadt
ORDER BY Bundeslaender.hauptstadt ASC;
```

Output

hauptstadt	einwohner	land
Berlin	3782202	Berlin
Bremen	577026	Bremen
Dresden	566222	Sachsen
Düsseldorf	631217	Nordrhein-Westfalen
Erfurt	215675	Thüringen
Hamburg	1910160	Hamburg
Hannover	548186	Niedersachsen
Kiel	248873	Schleswig-Holstein
Magdeburg	240114	Sachsen-Anhalt
Mainz	222889	Rheinland-Pfalz
München	1510378	Bayern
Potsdam	187119	Brandenburg
Saarbrücken	183509	Saarland
Stuttgart	633484	Baden-Württemberg
Wiesbaden	285522	Hessen

Daten abfragen mit SQL

Aufgabe 7

In der Tabelle Grossstaedte sollen die Großstädte gezählt werden, die zu einem Bundesland gehören. Bundesland und Anzahl der darin befindlichen Großstädte sollen aufgelistet werden, absteigend nach der Anzahl der Großstädte und aufsteigend nach dem Namen des Bundeslandes.

Schreibe eine entsprechende SQL-Abfrage. Nutze die land_id und den Befehl `INNER JOIN`.

Beispiellösung

```
SELECT Bundeslaender.land, COUNT (Grossstaedte.land_id) AS anzahl_grossstaedte
FROM Grossstaedte
INNER JOIN Bundeslaender ON Grossstaedte.land_id = Bundeslaender.land_id
GROUP BY Bundeslaender.land
ORDER BY anzahl_grossstaedte DESC, Bundeslaender.land ASC;
```

Output

land	anzahl_grossstaedte
Nordrhein-Westfalen	30
Baden-Württemberg	9
Bayern	8
Niedersachsen	8
Hessen	6
Rheinland-Pfalz	5
Sachsen	3
Brandenburg	2
Bremen	2
Sachsen-Anhalt	2
Schleswig-Holstein	2
Thüringen	2
Berlin	1
Hamburg	1
Mecklenburg-Vorpommern	1
Saarland	1