Einsatz und Realisierung von Datenbanksystemen

ERDB Übungsleitung

Alice Rey, Maximilian Bandle, Michael Jungmair

i3erdb@in.tum.de

Folien erstellt von Maximilian Bandle & Alexander Beischl

Organisatorisches
Disclaimer

Die Folien werden von der Übungsleitung allen Tutoren zur Verfügung gestellt.

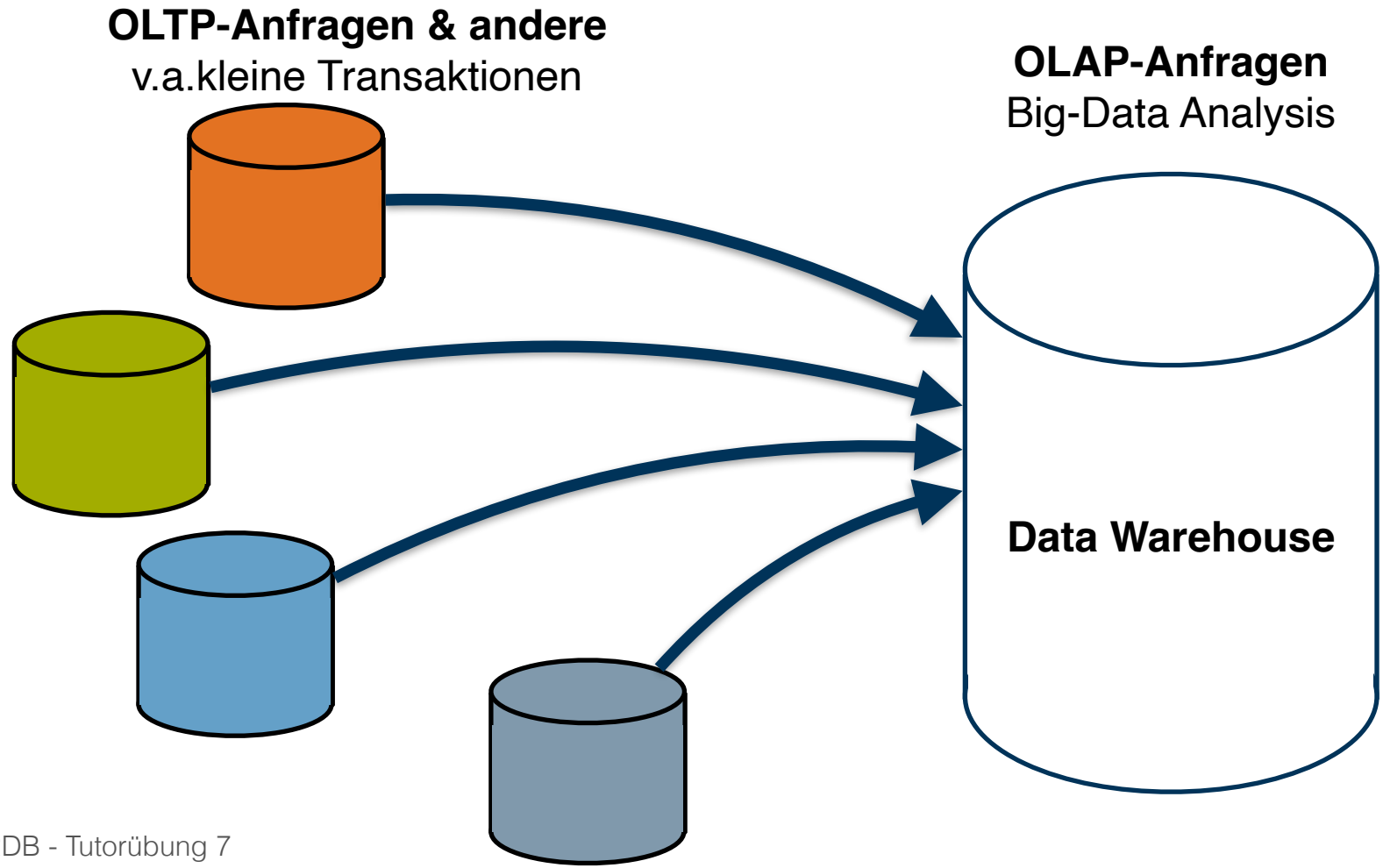
Sollte es Unstimmigkeiten zu den Vorlesungsfolien von Prof. Kemper geben, so sind die Folien aus der Vorlesung ausschlaggebend.

Falls Ihr einen Fehler oder eine Unstimmigkeit findet, schreibt an i3erdb@in.tum.de mit Angabe der Foliennummer.

Betriebliche Anwendungen

Betriebliche Anwendungen

Data Warehouses



Betriebliche Anwendungen

Online Transaction Processing

- Realisiert „operationale“ Tagesgeschäfte („mission-critical“)

- Charakterisierung
  - Hoher Parallelitätsgrad
  - Viele kurze TA (Tausende pro Sekunde)
  - Begrenzte Datenmenge pro TA
  - Operieren auf jüngsten, aktuell gültigen Zustand der DB
  - Hohe Verfügbarkeit muss gewährleistet sein

- Normalisierte Relationen (möglichst geringe Update-Kosten)

- Wenige Indexe (Fortschreibungskosten)

Online Analytical Processing

- Zur strategischen Unternehmensplanung

- Große Datenmengen

- Greift auf historische Daten zu

➡ Gewährt Rückschlüsse auf Entwicklungen

➡ Bestandteil von Decision-Support-Systeme/Management-Informationssysteme

SQL
Window Functions

- Sehr vielseitig und geeignet für
  - Zeitliche Analysen
  - Rangbasierte Anfragen
  - Top-K
  - Gleitender Durchschnitt
  - Kumulative Summe

- Window Functions werden nach **group by** und vor **order by** ausgewertet

Betriebliche Anwendungen

Window Funktionen

erdb		
name	**übung**	**punkte**
Thuy	1	1
Anna	1	1
Domi	1	2
Tobi	2	1
Thuy	2	1
Thuy	3	2
Anna	3	1
Domi	3	1
Thuy	4	3
Tobi	5	2
Domi	5	1
Anna	5	1
Tobi	6	2
Thuy	6	1

```
SELECT name, übung, (100.0*punkte)/
    sum(punkte)
    over (partition by übung) as prozent
FROM erdb
```

Betriebliche Anwendungen

Window Funktionen

erdb		
name	**übung**	**punkte**
Thuy	1	1
Anna	1	1
Domi	1	2
Tobi	2	1
Thuy	2	1
Thuy	3	2
Anna	3	1
Domi	3	1
Thuy	4	3
Tobi	5	2
Domi	5	1
Anna	5	1
Tobi	6	2
Thuy	6	1

```
SELECT name, übung, (100.0*punkte)/
       sum(punkte)
       over (partition by übung) as prozent
FROM erdb
```

Ergebnis		
name	**übung**	**prozent**
Thuy	1	25.0
Anna	1	25.0
Domi	1	50.0
Tobi	2	50.0
Thuy	2	50.0
Thuy	3	50.0
Anna	3	25.0
Domi	3	25.0
Thuy	4	100.0
Tobi	5	50.0
Domi	5	25.0
Anna	5	25.0
Tobi	6	66.7
Thuy	6	33.3

Betriebliche Anwendungen

Window Funktionen

erdb		
name	**übung**	**punkte**
Anna	1	1
Anna	3	1
Anna	5	1
Domi	1	2
Domi	3	1
Domi	5	1
Thuy	1	1
Thuy	2	1
Thuy	3	2
Thuy	4	3
Thuy	6	1
Tobi	2	1
Tobi	5	2
Tobi	6	2

```
SELECT name, übung, sum(punkte)
over ( partition by name
order by übung)
FROM erdb
```

Ergebnis		
name	**übung**	**sum**
Anna	1	1
Anna	3	2
Anna	5	3
Thuy	1	1
Thuy	2	2
Thuy	3	4
Thuy	4	7
Thuy	6	8
Domi	1	2
Domi	3	3
Domi	5	4
Tobi	2	1
Tobi	5	3
Tobi	6	5

Betriebliche Anwendungen

Window Funktionen

erdb		
name	**übung**	**punkte**
Anna	1	1
Anna	3	1
Anna	5	1
Domi	1	2
Domi	3	1
Domi	5	1
Thuy	1	1
Thuy	2	1
Thuy	3	2
Thuy	4	3
Thuy	6	1
Tobi	2	1
Tobi	5	2
Tobi	6	2

```
SELECT name, übung, sum(punkte)
    over ( partition by name
        order by übung
        range between unbounded
        preceding and current row)
FROM erdb
```

Ergebnis		
name	**übung**	**sum**
Anna	1	1
Anna	3	2
Anna	5	3
Thuy	1	1
Thuy	2	2
Thuy	3	4
Thuy	4	7
Thuy	6	8
Domi	1	2
Domi	3	3
Domi	5	4
Tobi	2	1
Tobi	5	3
Tobi	6	5

Betriebliche Anwendungen

Window Funktionen

erdb		
name	**übung**	**punkte**
Anna	1	1
Anna	3	1
Anna	5	1
Domi	1	2
Domi	3	1
Domi	5	1
Thuy	1	1
Thuy	2	1
Thuy	3	2
Thuy	4	3
Thuy	6	1
Tobi	2	1
Tobi	5	2
Tobi	6	2

```
SELECT name, übung, sum(punkte)
  over (partition by name
    order by übung
    range between 1 preceding
      and 1 following)
FROM erdb
```

Ergebnis		
name	**übung**	**sum**
Anna	1	1
Anna	3	1
Anna	5	1
Thuy	1	2
Thuy	2	4
Thuy	3	6
Thuy	4	5
Thuy	6	1
Domi	1	2
Domi	3	1
Domi	5	1
Tobi	2	1
Tobi	5	4
Tobi	6	4

Betriebliche Anwendungen

Window Funktionen

erdb		
name	**übung**	**punkte**
Anna	1	1
Anna	3	1
Anna	5	1
Domi	1	2
Domi	3	1
Domi	5	1
Thuy	1	1
Thuy	2	1
Thuy	3	2
Thuy	4	3
Thuy	6	1
Tobi	2	1
Tobi	5	2
Tobi	6	2

```
SELECT name, übung, sum(punkte)
  over (partition by name
    order by übung
    rows between 1 preceding
      and 1 following)
FROM erdb
```

Ergebnis		
name	**übung**	**sum**
Anna	1	2
Anna	3	3
Anna	5	2
Thuy	1	2
Thuy	2	4
Thuy	3	6
Thuy	4	6
Thuy	6	4
Domi	1	3
Domi	3	4
Domi	5	2
Tobi	2	3
Tobi	5	5
Tobi	6	4

Betriebliche Anwendungen

Window Funktionen

erdb		
name	**übung**	**punkte**
Anna	1	1
Anna	3	1
Anna	5	1
Domi	1	2
Domi	3	1
Domi	5	1
Thuy	1	1
Thuy	2	1
Thuy	3	2
Thuy	4	3
Thuy	6	1
Tobi	2	1
Tobi	5	2
Tobi	6	2

```
SELECT name, sum(punkte) as gesamt
FROM erdb
GROUP BY name
ORDER BY gesamt desc
```

erdb nach group	
name	**gesamt**
Thuy	8
Tobi	5
Domi	4
Anna	3

Betriebliche Anwendungen

Window Funktionen

erdb nach group	
name	**gesamt**
Thuy	8
Tobi	5
Domi	4
Anna	3

```
SELECT name, gesamt,
    rank() over (order by gesamt desc)
FROM (
    SELECT name, sum(punkte) as
        gesamt
    FROM erdb
    GROUP BY name desc)
```

Ergebnis		
name	**gesamt**	**rank**
Thuy	8	1
Tobi	5	2
Domi	4	3
Anna	3	4

Betriebliche Anwendungen

Window Funktionen

erdb nach group	
name	**gesamt**
Thuy	8
Tobi	5
Domi	4
Anna	3

```
SELECT name,
    lag(name) over(order by gesamt desc)
    as mehr,
    lead(name) over(order by gesamt desc)
    as weniger
FROM (
    SELECT name, sum(punkte) as gesamt
    FROM erdb
    GROUP BY name desc)
```

Ergebnis		
name	**mehr**	**weniger**
Thuy	null	Tobi
Tobi	Thuy	Domi
Domi	Tobi	Anna
Anna	Domi	null

Aufgabe 1

Analysieren wir die Gehälter von Professoren mittels Windowfunctions und führen Sie die Abfragen unter `hyper-db.de` aus. Dazu orientieren wir uns an der Relation *Professoren* des erweiterten Universitätsschemas:

1. Ermitteln Sie zu jedem Professor das Durchschnittsgehalt aller Professoren.
2. Ermitteln Sie zu jedem Professor das Durchschnittsgehalt aller Professoren partitioniert nach Rang.
3. Ermitteln Sie nun die wachsende Summe (das Quantil) des Gehaltes aller Professoren partitioniert nach Rang und absteigend sortiert nach ihrem Gehalt. Gleich verdienende Professoren sind im selben Quartil.
4. Ermitteln Sie nun die wachsende Summe des Gehaltes aller Professoren partitioniert nach Rang und absteigend (total) sortiert nach ihrem Gehalt (reihenweise, nicht als Range-Query).

Aufgabe 1

Analysieren wir die Gehälter von Professoren mittels Windowfunctions und führen Sie die Abfragen unter `hyper-db.de` aus. Dazu orientieren wir uns an der Relation *Professoren* des erweiterten Universitätsschemas:

5. Ermitteln Sie nun das gleitende Durchschnittsgehalt aus genau zwei mehr bzw. weniger verdienenden Professoren sortiert nach Gehalt und partitioniert nach Rang.
6. Ermitteln Sie nun das gleitende Durchschnittsgehalt aus den 500 Einheiten mehr bzw. weniger verdienenden Professoren sortiert nach Gehalt und partitioniert nach Rang. (weder in PostgreSQL noch in HyPer implementiert, fragen Sie Ihren Tutor)
7. Geben sie zu jedem Professor das Gehalt des eins besser wie eins schlechter verdienenden.
8. Ermitteln Sie die drei bestverdienendsten Professoren einmal mit und einmal ohne Windowfunctions.

Aufgabe 2

1. Bestimmen Sie die Durchschnittsnote für jeden Studenten.
2. Basierend auf dieser Durchschnittsnote, bestimmen Sie für alle Studenten ihren Rangplatz innerhalb ihrer Kohorte (Studenten desselben Semesters).
3. Berechnen Sie zusätzlich für jeden Studenten auch noch die **Abweichung** seiner Durchschnittsnote von der Durchschnittsnote der Kohorte (also vom Durchschnitt der Durchschnittsnote der Studenten der Kohorte) ausgegeben werden.

Lösen Sie Teilaufgaben 2 und 3 jeweils einmal mit und einmal ohne Nutzung von Windowfunktionen. Ihre Anfragen können Sie auf `hyper-db.de` testen. Nutzen Sie folgende erweiterte *pruefen* Relation:

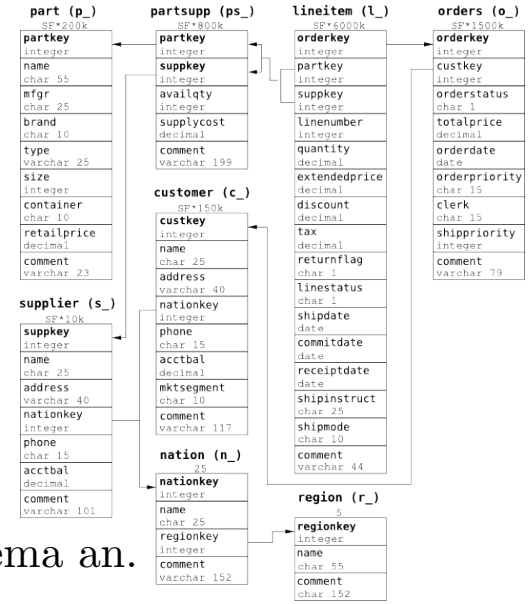
```
with mehr_pruefen(MatrNr,VorlNr,PersNr,Note) as (
 select * from pruefen
 union
 values (29120,0,0,3.0),(29555,0,0,2.0),(29555,0,0,1.3),(29555,0,0,1.0)
)
```

Aufgabe 3

Betrachten wir das bekannte Uni-Schema mit den Faktentabellen `hoeren` und `pruefen`.

1. Ermitteln Sie in SQL mittels Fensterfunktionen (Windowfunctions) die Top-3 Studenten pro Vorlesung und geben Sie deren Namen aus.
2. Ermitteln Sie mittels SQL-92, um wieviele Notenstufen Studenten, die die Vorlesung gehört haben, in der Prüfung besser abgeschnitten haben.

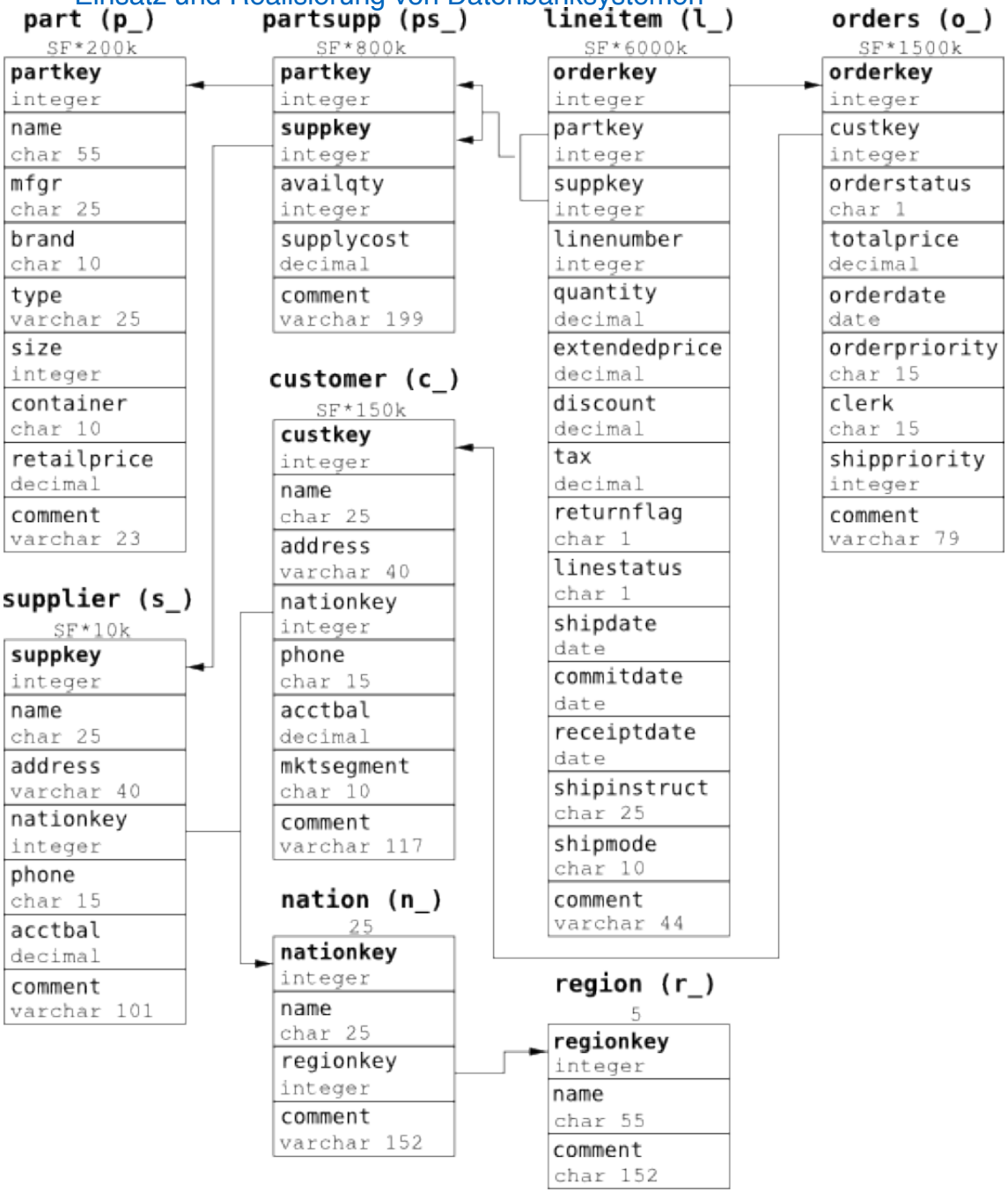
Aufgabe 4



Wenden wir nun Fensterfunktionen an dem fiktiven TPC-H Schema an.

1. Erstellen Sie in SQL eine Abfrage nach dem jährlichen Exportvolumen pro Jahr und Land. Verwenden Sie diese Abfrage als Hilfstabelle (`with`-Statement) in den folgenden Abfragen, orienteren Sie sich an TPC-H Anfrage 7.
2. Ranken Sie Länder anhand ihres jährlichen Exportvolumens, auf Platz eins ist das Land mit dem höchsten Volumen, das je in einem Jahr getätigt worden ist.
3. Küren Sie nun die Jahressieger. Ranken Sie dazu die Länder partitioniert nach Jahr.
4. Ermitteln Sie nun das laufende Exportmittel der Länge drei (Vorjahr, Nachjahr, falls erfasst).

TPCH - Schema


part (p_)
SF*200k
partkey integer
name char 55
mfgr char 25
brand char 10
type varchar 25
size integer
container char 10
retailprice decimal
comment varchar 23
partsupp (ps_)
SF*800k
partkey integer
suppkey integer
availqty integer
supplycost decimal
comment varchar 199
lineitem (l_)
SF*6000k
orderkey integer
partkey integer
suppkey integer
linenumber integer
quantity decimal
extendedprice decimal
discount decimal
tax decimal
returnflag char 1
linestatus char 1
shipdate date
commitdate date
receiptdate date
shipinstruct char 25
shipmode char 10
comment varchar 44
orders (o_)
SF*1500k
orderkey integer
custkey integer
orderstatus char 1
totalprice decimal
orderdate date
orderpriority char 15
clerk char 15
shippriority integer
comment varchar 79
customer (c_)
SF*150k
custkey integer
name char 25
address varchar 40
nationkey integer
phone char 15
acctbal decimal
mktsegment char 10
comment varchar 117
supplier (s_)
SF*10k
suppkey integer
name char 25
address varchar 40
nationkey integer
phone char 15
acctbal decimal
comment varchar 101
nation (n_)
25
nationkey integer
name char 25
regionkey integer
comment varchar 152
region (r_)
5
regionkey integer
name char 55
comment char 152

Aufgabe 5

Betrachten Sie die folgende Tabelle `Waren` mit verkauften Produkten in einem Supermarkt. Die Spalte `verkauft` besagt, wieviele Einheiten des jeweiligen Produktes verkauft worden sind.

Name	Preis	Kategorie	Verkauft
Brot	1.00	Backwaren	8128
Butter	0.80	Kühlwaren	496
Grill	60.00	Haushalt	6
Steak	8.00	Kühlwaren	28
...	...	...	...

1 Ermitteln Sie in SQL mittels Fensterfunktionen (Windowfunctions) den prozentualen Umsatzanteil jedes Produktes innerhalb seiner Kategorie.

2 Ermitteln Sie in SQL mittels Fensterfunktionen (Windowfunctions) für jedes Produkt das Mittel der Verkaufszahlen aus den 5 besser verkauften (höhere Verkaufszahlen) Produkten geordnet nach Verkaufszahlen.

3 Ermitteln Sie in SQL mittels Fensterfunktionen (Windowfunctions) die drei Produkte mit dem meisten Umsatz pro Kategorie.

Fragen?