Datenbanksysteme 2, 2. Übung Vertiefung SQL (Hierarchische Anfragen, CTE und Analytische Funktionen) Lösungshinweise

Aufgabe 2.1 Anfragen an die Movie-Datenbank

Formulieren Sie für die folgenden Aufgaben eine SQL-Anfrage und werten Sie sie auf der Datenbank aus.
Wenn im Folgenden von Filmen gesprochen wird, sind alle Einträge in MOVIE gemeint, also auch TV-Filme, Serien, etc. Wenn von Kinofilmen gesprochen wird, sind nur die mit Type=C gemeint.

a) Finden Sie die Regisseure, deren Kinofilme im Mittel das beste bzw. das schlechteste Rating erhalten haben sowie den jeweiligen Rating-Wert. Zur Vereinfachung dürfen Sie auch mehrere Anfragen verwenden. Wie könnte eine Lösung mit einer Anfrage aussehen (Hinweis: denken Sie an Common Table Expressions)?

Lösung mit CTEs:

```
with ar as
(select avg(r.rating) ar, p.name
 from moviedb.person p
 join moviedb.directs d on (p.id = d.director)
 join moviedb.movie m on (d.movie = m.id)
 join moviedb.rating r on (m.id = r.movie)
 where m.type = 'C'
 group by p.name
),
minmax as
(select min(ar) min_ar, max(ar) max_ar from ar)
select ar.name, ar.ar
from ar
join minmax on (ar.ar = minmax.min_ar or ar.ar = minmax.max_ar)
```

Alternative:

```
with ar as
(select avg(r.rating) ar, p.name
 from moviedb.person p
 join moviedb.directs d on (p.id = d.director)
 join moviedb.movie m on (d.movie = m.id)
 join moviedb.rating r on (m.id = r.movie)
 where m.type = 'C'
 group by p.name
),
minmax as
(select min(ar) min_ar from ar
 union
 select max(ar) max_ar from ar
```

```
)
select ar.name, ar.ar
from ar where ar.ar in (select * from minmax);
```

54 Ergebnisse (9 mal Rating 1, 45 mal Rating 9.9)

b) Wiederholen Sie a) und verwenden Sie dabei zusätzlich Analytische Funktionen. Formulieren sie dieses Mal nur eine Anfrage, die sowohl die besten als auch schlechtesten Regisseure auflistet.

Ansatz mit nur besten Ratings:

```
SELECT name, rating FROM (
      select p.name, avg(r.rating) as rating,
             RANK() OVER (ORDER BY avg(r.rating) DESC) AS rank
      from moviedb.person p
      join moviedb.directs d on (p.id = d.director)
      join moviedb.movie m on (d.movie = m.id)
      join moviedb.rating r on (m.id = r.movie)
      where m.type = 'C'
      group by p.name)
WHERE rank = 1;
```

Analog mit ASC für die am schlechtesten gerateten Regisseure.

```
select name, rating from (
select p.name, avg(r.rating) as rating,
       rank() over (order by avg(r.rating) desc) r1,
       rank() over (order by avg(r.rating) asc) r2
from moviedb.person p
join moviedb.directs d on (p.id = d.director)
join moviedb.movie m on (d.movie = m.id)
join moviedb.rating r on (m.id = r.movie)
where m.type = 'C'
group by p.name
) where r1 = 1 or r2 = 1;
```

54 Ergebnisse

c) Finden Sie die Namen aller Filme, bei denen das Budget mehr als 10.000.000 pro Woche, die sie gelaufen sind, betrug. Es dürfen die Währungen vernachlässigt werden! Sie sollten die maximale Laufzeit eines Films über alle Länder verwenden; wenn keine Laufzeit bekannt ist oder die Laufzeit 0 Wochen beträgt, sollten Sie mit 1 Woche rechnen. Wenn mehrere Budgets in mehreren Währungen für einen Film angegeben sind, verwenden sie den höchsten Betrag.

```
SELECT title, (budget / weeks) AS per_week
FROM (
  SELECT m.title AS title,
    MAX(b.budget) AS budget,
    NVL(DECODE(MAX(r.weeks),0,1,MAX(r.weeks)),1) AS weeks
  FROM moviedb.movie m
```

```
  JOIN moviedb.budget b ON (b.movie = m.id)
  LEFT JOIN moviedb.runs r ON (r.movie = m.id)
  GROUP BY m.id, m.title
)
WHERE (budget / weeks) > 10000000;
```

1.793 Ergebnisse

d) Finden Sie ID und Titel aller Kinofilme und die Anzahl der darin auftretenden Charaktere für die Kinofilme, in denen mindestens 15 Charaktere gespeichert sind.

```
SELECT m.id, m.title, COUNT(*) as anzahl
FROM moviedb.movie m
JOIN moviedb.plays pl ON (pl.movie = m.id)
WHERE m.type = 'C'
GROUP BY m.id, m.title
HAVING COUNT(*) >= 15;
```

117.962 Ergebnisse

Aufgabe 2.2 Analytische Funktionen

Sie haben in der Datenbank folgende Daten in einer Tabelle gespeichert:

ID	WERT1	WERT2
1	3	2
2	5	8
3	4	10
4	1	1
5	2	8
6	5	4
7	4	9
8	4	9
9	4	6
10	2	8

Berechnen Sie die Werte für folgende Analytische Funktionen für jede Zeile der Tabelle:

a) ROW_NUMBER() OVER (ORDER BY WERT1)
b) RANK() OVER (ORDER BY WERT1)
c) DENSE_RANK() OVER (ORDER BY WERT1)
d) ROW_NUMBER() OVER (PARTITION BY WERT1 ORDER BY WERT2)
e) RANK() OVER (PARTITION BY WERT1 ORDER BY WERT2)
f) DENSE_RANK() OVER (PARTITION BY WERT1 ORDER BY WERT2)
g) SUM(WERT2) OVER (PARTITION BY WERT1)
h) SUM(WERT2) OVER (PARTITION BY WERT1 ORDER BY ID)

Sie können das Ergebnis direkt in folgender Tabelle eintragen. Markieren sie dabei Zellen, in denen mehrere Ergebnisse möglich wären.
Erstellen sie anschließend eine passende Tabelle in der Datenbank und werten sie die Funktionen über eine SQL-Anfrage aus. Vergleichen sie die Ergebnisse.

ID	WERT1	WERT2	a)	b)	c)	d)	e)	f)	g)	h)
1	3	2	4	4	3	1	1	1	2	2
2	5	8	**9**	9	5	2	2	2	12	8
3	4	10	**5**	5	4	4	4	3	34	10
4	1	1	1	1	1	1	1	1	1	1
5	2	8	**2**	2	2	**1**	1	1	16	8
6	5	4	**10**	9	5	1	1	1	12	12
7	4	9	**6**	5	4	**2**	2	2	34	19
8	4	9	**7**	5	4	**3**	2	2	34	28
9	4	6	**8**	5	4	1	1	1	34	34
10	2	8	**3**	2	2	**2**	1	1	16	16