TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Alfons Kemper, Ph.D.

Übung zur Vorlesung *Einsatz und Realisierung von Datenbanken* im SoSe20

Maximilian {Bandle, Schüle}, Josef Schmeißer (i3erdb@in.tum.de)
http://db.in.tum.de/teaching/ss20/impldb/

Blatt Nr. 07

Hausaufgabe 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:

```
with Professoren (persnr, name, rang, raum, gehalt, steuerklasse) as (
  values (2125,'Sokrates','C4',226,85000,1),
  (2126,'Russel','C4',232,80000,3),
  (2127,'Kopernikus','C3',310,65000,5),
  (2128,'Aristoteles','C4',250,85000,1),
  (2133,'Popper','C3',52,68000,1),
  (2134,'Augustinus','C3',309,55000,5),
  (2136,'Curie','C4',36,95000,3),
  (2137,'Kant','C4',7,98000,1)
)
```

1. Ermitteln Sie zu jedem Professor das Durchschnittsgehalt aller Professoren.

   ```
   SELECT *, avg(gehalt) OVER () FROM Professoren;
   ```

2. Ermitteln Sie zu jedem Professor das Durchschnittsgehalt aller Professoren partitioniert nach Rang.

   ```
   SELECT *, avg(gehalt) OVER (PARTITION BY rang) FROM Professoren;
   ```

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.

   ```
   SELECT *, sum(gehalt) OVER (PARTITION BY rang ORDER BY gehalt DESC)
   FROM Professoren;
   -- aequivalent zu
   SELECT *, sum(gehalt) OVER (PARTITION BY rang ORDER BY gehalt DESC
   RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) FROM Professoren;
   ```

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).

   ```
   SELECT *, sum(gehalt) OVER (PARTITION BY rang ORDER BY gehalt DESC
   ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) FROM Professoren;
   ```

5. Ermitteln Sie nun das gleitende Durchschnittsgehalt aus genau zwei mehr bzw. weniger verdienenden Professoren sortiert nach Gehalt und partitioniert nach Rang.

   ```
   SELECT *, avg(gehalt) OVER (PARTITION BY rang ORDER BY gehalt DESC
   ROWS BETWEEN 2 PRECEDING AND 2 FOLLOWING) FROM Professoren;
   ```

6. Ermitteln Sie nun das gleitende Durchschnittsgehalt aus den 500 Einheiten mehr bzw. weniger verdienenden Professoren sortiert nach Gehalt und partitioniert nach Rang.
```
SELECT *, avg(gehalt) OVER (PARTITION BY rang ORDER BY gehalt DESC
RANGE BETWEEN 500 PRECEDING AND 500 FOLLOWING) FROM Professoren;
```
7. Geben sie zu jedem Professor das Gehalt des eins besser wie eins schlechter verdienenden.
```
SELECT *, lag(gehalt) OVER (ORDER BY gehalt DESC),
  lead(gehalt) OVER (ORDER BY gehalt DESC) FROM Professoren;
```
8. Ermitteln Sie die drei bestverdienendsten Professoren einmal mit und einmal ohne Windowfunctions.
```
SELECT * FROM (
   SELECT *, rank() OVER (ORDER BY gehalt desc) FROM Professoren
) WHERE rank < 4
```
```
SELECT * FROM professoren p_selbst WHERE 3>(SELECT count(*)
FROM professoren p_reich WHERE p_selbst.gehalt < p_reich.gehalt)
```

Hausaufgabe 2

Lösen Sie folgende Anfrage mit SQL basierend auf dem bekannten Universitätschema.

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)
)
```

Erzeugen der Hilfstabellen:

```
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)
), noten(MatrNr,Semester,Note) as (
  select s.matrnr, semester, avg(Note)
  from studenten s, mehr_pruefen p
  where s.matrnr=p.matrnr
  group by s.matrnr,semester
)
```

Ohne Windowfunktionen:

```
select *,
(select count(*)+1 from noten x
    where x.Semester=n.Semester and x.Note<n.Note) as Rang,
(select avg(x.Note) from noten x
    where x.Semester=n.Semester) as GPA,
(select avg(x.Note) from noten x
    where x.Semester=n.Semester) - note as Abweichung
from noten n order by semester, rang
```

Mit Windowfunktionen:

```
select *,
rank() over (partition by Semester order by Note asc) as Rang,
avg(Note) over (partition by Semester) as GPA,
avg(Note) over (partition by Semester) - note as Abweichung
from noten order by semester, rang
```

Hausaufgabe 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.

```
SELECT Name
FROM(
   SELECT s.Name, rank() over (PARTITION BY VorlNr ORDER BY Note)
   FROM pruefen p, Studenten s
   WHERE s.MatrNr = p.MatrNr
)
WHERE rank < 4
```

2. Ermitteln Sie mittels SQL-92, um wieviele Notenstufen Studenten, die die Vorlesung gehört haben, in der Prüfung besser abgeschnitten haben.

```
select anwesend.Note-abwesend.Note as besser
from(
   select avg(p.Note) as Note
   from studenten s, pruefen p
   where s.MatrNr=p.MatrNr and not exists (
     select *
     from hoeren h where h.MatrNr=s.MatrNr and h.VorlNr=p.VorlNr)
  ) abwesend,
   (select avg(p2.Note) Note
   from studenten s2, pruefen p2
   where s2.MatrNr=p2.MatrNr and exists (
     select * from hoeren h2
     where h2.MatrNr=s2.MatrNr and h2.VorlNr=p2.VorlNr)
  ) anwesend
```

Hausaufgabe 4

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

1. Erstellen Sie in SQL eine Abfrage nach dem jährlichen Produktionsvolumen pro Jahr und Land. Verwenden Sie diese Abfrage als Hilfstabelle (`with`-Statement) in den folgenden Abfragen, orienteren Sie sich an TPC-H Anfrage 7.

   ```
   with volume as (
     select n_name, l_year, sum(volume) as revenue
     from (
        select n_name, extract(year from l_shipdate) as l_year,
           l_extendedprice * (1 - l_discount) as volume
        from supplier,lineitem,orders,nation
        where s_suppkey = l_suppkey
           and o_orderkey = l_orderkey and s_nationkey = n_nationkey
         ) as shipping
     group by n_name, l_year
   )
   ```

2. Ranken Sie Länder anhand ihres jährlichen Produktionsvolumens, auf Platz eins ist das Land mit dem höchsten Volumen, das je in einem Jahr getätigt worden ist.

   ```
   select *, rank() over (order by revenue desc) from volume;
   ```

3. Küren Sie nun die Jahressieger. Ranken Sie dazu die Länder partitioniert nach Jahr.

   ```
   select * from (
     select *, rank() over (partition by l_year order by revenue desc)
     from volume)
   where rank=1;
   ```

4. Ermitteln Sie nun das laufende Produktionsmittel der Länge drei (Vorjahr, Nachjahr, falls erfasst).

   ```
   select *, avg(revenue) over (
     partition by n_name order by l_year
     range between 1 preceding and 1 following)
     from volume
   ```

Hausaufgabe 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.

```
select name,kategorie,
       verkauft * preis * 100.0 / sum(verkauft * preis)
       over (partition by kategorie)
from Waren;
```

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.

```
select name, avg(verkauft) over (order by verkauft desc
           rows between 5 preceding and current row)
from Waren;
```

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

```
select *
from ( select *, rank() over (partition by kategorie
                              order by (verkauft * preis) desc)
       from waren)
where rank <= 3;
```