Since in the result relation each group is represented by exactly one tuple, in the 
**select** clause only aggregate functions can appear, or attributes that are used for 
grouping, i.e., that are also used in the **group by** clause.

**Sorting**

- frequently a sorted output is required 
  \[\rightarrow\] DBMS needs sort operator, sorting is expensive

- sorted output with the **order by** clause with respect to one or more attributes
  - **order by** \[asc | desc\] \(A_1, ..., [asc | desc] A_n\) \(A_i\) attribute
  
  - sorting order:
    - keyword **asc** = ascending (default)
    - keyword **desc** = descending

- The **order by** clause is the last clause in an SQL command.
example: Determine personell id, name and rank of all professors; sort the result tuples in descending order by rank and in ascending order by name.

`select pers-id, name, rank from professors order by rank desc, name asc`

<table>
<thead>
<tr>
<th>pers-id</th>
<th>name</th>
<th>rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>2136</td>
<td>Curie</td>
<td>C4</td>
</tr>
<tr>
<td>2137</td>
<td>Kant</td>
<td>C4</td>
</tr>
<tr>
<td>2126</td>
<td>Russel</td>
<td>C4</td>
</tr>
<tr>
<td>2125</td>
<td>Sokrates</td>
<td>C4</td>
</tr>
<tr>
<td>2134</td>
<td>Augustinus</td>
<td>C3</td>
</tr>
<tr>
<td>2127</td>
<td>Kopernikus</td>
<td>C3</td>
</tr>
<tr>
<td>2133</td>
<td>Popper</td>
<td>C3</td>
</tr>
</tbody>
</table>

- attribute `rank` main sorting condition, attribute `name` minor sorting condition
Nested queries

- In the **where** clause and in the **from** clause of an SQL statement further SQL statements can appear. This is called a nested query.

- In the **where**-clause we differentiate whether the result of a subquery yields a scalar value or a relation.

- **scalar subqueries**
  - example: Which students with a semester number less than the average are there?
    ```sql
    select name, sem
    from students
    where sem < (select avg(sem) from students)
    ```
  - Scalar subqueries in SQL92 are even allowed in the **select** clause of a query. In Oracle this feature is currently not supported.

- **scalar subqueries with exists**
  - In the **where** clause also subqueries are allowed that yield a boolean value. These are indicated by the keyword **exists**.
  - The condition “[not] exists <subquery>” is true if the subquery is not empty [empty].
Set-valued subqueries
- The keyword [not] in tests if an attribute [does not take] takes a value of a set.
- If the task is to test whether an attribute is in a certain relationship to all elements of a set, the keyword all can be used.

queries with forall quantifiers
- Mathematical law: $(\forall x : \varphi(x)) \iff (\neg\exists x : \neg\varphi(x))$. Hence, all queries containing a forall quantifier can be transformed to equivalent queries only containing existential quantifiers.
- Example: Which students attend all lectures offered by professor Curie?

```sql
select s.name from students as s
where not exists
(select id from lectures, professors
 where pers-id = held_by and name = "Curie")
except
(select l.id from attends as a, lectures as l
 where l.id = a.id and a.reg-id = s.reg-id))
```
subqueries in the from clause

- Since an SQL query creates a relation, a query can also be used in the from clause.
- example: Output the ids of those lectures that are attended by more than 20 students.

```sql
select id
from (select id, count(*) as number from attends group by id)
where number > 20
```

- possible to explicitly use a join operator in SQL92 in the from clause by means of the keywords
  + cross join for the Cartesian (cross) product,
  + natural join for the natural join,
  + join or inner join for theta join,
  + left outer join, right outer join or full outer join for outer join
    analogously to the operators of the relational algebra: also tuples of the left, the right or both relations, which do not fulfil the join, are inserted into the result relation
  + union join: some kind of full outer join where no comparison is performed. Both schemas are concatenated. Tuples are united and supplemented by null values.