- Determine the number of hours per week in which professors have given lectures.

```sql
select held_by, sum(hpw) as number
from lectures
group by held_by
```

- Determine the number of hours per week of those lectures held by professors who predominantly give long lectures (> 2 hours per week on average).

```sql
select held_by, sum(hpw) as number
from lectures
group by held_by
having avg(hpw) > 2
```

- Determine the number of hours per week of those lectures held by C4 professors who predominantly give long lectures (> 2 hours per week on average).

```sql
select held_by, name, sum(hpw) as number
from lectures, professors
where held_by = pers-id and rank = “C4”
group by held_by, name
having avg(hpw) > 2
```
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
  - 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].