Practical script Nr. 6

# Data aggregation

To select aggregated values there are aggregation functions in place. For example, here’s a script to select average grate.

SELECT AVG(Grade)

FROM Grade

The main (most common) aggregation functions are:

AVG – average value  
SUM- sum of values  
MIN – minimal value  
MAX – maximum values  
COUNT – amount

Aggregation functions ignore *NULL* values

To select count of rows you would need to use special syntax:  
SELECT COUNT(\*)

FROM Teacher

If you need to select number of mentors (non-NULL values):

SELECT COUNT(Mentor\_ID)

FROM Teacher

The select would ignore all the rows where Mentor\_ID IS NULL

In case you want to select number of different mentors:

SELECT COUNT(DISTINCT Mentor\_ID)

FROM Teacher

The previous query would ignore all non-NULL Mentor\_ID values and count only distinct ones

AVG and SUM works only for numeric values. MIN, MAX, COUNT works for all data types

When using aggregation functions, you are not limited to one function. Here an example selecting minimal, maximum and average grate for Examination 5000001

SELECT MIN(Grade), MAX(Grade), AVG(Grade)

from grade

WHERE Examination\_ID=5000001

AVG rounds the value and returns integer type. In case you need a floating-point average, you can multiply by 1.0 (This is special to Microsoft SQL Server)

SELECT AVG(Grade\*1.0)

from grade

WHERE Examination\_ID=5000001

Aggregation function accepts not only column names, but expressions as well

In case you need to calculate average only from distinct values, use the DISTINCT keyword:

SELECT COUNT(DISTINCT Grade), SUM(DISTINCT Grade), SUM(Grade)

from grade

WHERE Examination\_ID=5000001

# Results grouping

Data can be grouped in the query, for example to select average grade for each examination you would use:

SELECT Examination\_ID, AVG(Grade\*1.0)

from grade

GROUP BY Examination\_ID

You can then order the results using the aggregation (average grade)

SELECT Examination\_ID, AVG(Grade\*1.0)

from grade

GROUP BY Examination\_ID

ORDER BY AVG(Grade\*1.0)

Each column from SELECT should either be in the GROUP BY clause or presented as aggregation. It is also possible to group by multiple columns.

Example SQL, select each student grades with each teacher:

SELECT Student\_ID, Grading\_teacher\_ID, AVG(Grade\*1.0)

from Grade

GROUP BY Student\_ID, Grading\_teacher\_ID

ORDER BY AVG(Grade\*1.0)

# Filtering aggregation results

To filter out results from aggregated / group by query you can use HAVING clause. Example SQL selects average grade for each student and returns only those with grade greater than 7.

SELECT Student\_ID, AVG(Grade\*1.0)

from Grade

GROUP BY Student\_ID

HAVING AVG(Grade\*1.0)>7

Select exams, where there are more than 30 grades

SELECT Examination\_ID, AVG(Grade\*1.0), COUNT(GRADE)

from Grade

GROUP BY Examination\_ID

HAVING COUNT(Grade)>30

HAVING also works when you don’t use GROUP BY, but that noticeably increases the query execution time. In those cases, it’s better to use WHERE condition, which would filter the data before the resulting set is formed.

WHERE is applied on the table, before the results are computed. HAVING is run on resulting data, after it has been selected from the table.

# Filtering working data

In case we want to use only part of the data for our calculations, we can do so. WHERE clause still works and will help us in filtering the data.

For example, selecting number of grades for each examination, where grades are greater than 7

SELECT Examination\_ID , COUNT(\*)

FROM Grade

WHERE Grade>7

GROUP BY Examination\_ID

Select amount of grades greater than 7 selected in each course.

SELECT Examination.Course\_ID , COUNT(\*)

FROM Grade JOIN Examination

ON Grade.Examination\_ID=Examination.Examination\_ID

WHERE Grade>7

GROUP BY Examination.Course\_ID