Databases: Exercises 5 (24p/24p)

Task 1 [5p/5p]

A. The highest and lowest given tax rate from the Search student table. Enter maximum and minimum as column headings.

The SQL MIN() and MAX() Functions
The MIN() function returns the smallest value of the selected column.

The MAX() function returns the largest value of the selected column.

```
SELECT MAX(taxrate) AS maximum, MIN(taxrate) AS minimum
FROM students;
```

```
MariaDB [AD2008]> SELECT MAX(taxrate) AS maximum, MIN(taxrate) AS minimum
    -> FROM students;
+---------+---------+
| maximum | minimum |
+---------+---------+
|    9.90 |    0.00 |
+---------+---------+
1 row in set (0.033 sec)
```

B. How many credits are marked on the studentgradesboard?

The COUNT() function returns the number of rows that matches a specified criterion.

```
SELECT COUNT(grade) FROM studentgrades;
```

```
MariaDB [AD2008]> SELECT COUNT(grade) FROM studentgrades;
+--------------+
| COUNT(grade) |
+--------------+
|           18 |
+--------------+
1 row in set (0.037 sec)
```

C. The calculation is the average of the student tax rates. Enter as column headings Keskiarvo.
The AVG() function returns the average value of a numeric column

```
SELECT AVG(taxrate) AS average FROM students;
```

```
MariaDB [AD2008]> SELECT AVG(taxrate) AS average FROM students;
+----------+
| average  |
+----------+
| 5.387500 |
+----------+
1 row in set (0.038 sec)
```

D. How much is the income of all students in total , Enter the title of the result rowTulot yhteensä

The SUM() function returns the total sum of a numeric column.

```
SELECT SUM(incomes) AS totalincome FROM students;
```

```
MariaDB [AD2008]> SELECT SUM(incomes) AS totalincome FROM students;
+-------------+
| totalincome |
+-------------+
|    93061.02 |
+-------------+
1 row in set (0.042 sec)

MariaDB [AD2008]>
```

E. How many different eye colors do the students have on the personality board? Enter the title of the result lineLukumäärä

The COUNT() function returns the number of rows that matches a specified criterion.
The following SQL statement lists the number of different (distinct) eyecolor students

```
SELECT COUNT(distinct eyecolor) AS number FROM students;
```

```
MariaDB [AD2008]> SELECT COUNT(distinct eyecolor) AS number FROM students;
+--------+
| number |
+--------+
|      3 |
+--------+
1 row in set (0.055 sec)
```

Task 2 [4p/4p]

A. From the Search students table, the first character of all surnames.

- The LEFT() function extracts a number of characters from a string (starting from left).
- Also RIGHT() function.

Syntax

```
LEFT(string, number_of_chars)
```

```
SELECT left(lastname, 1) FROM students;
```

```
MariaDB [AD2008]> SELECT left(lastname, 1) FROM students;
+------------------+
| left(lastname, 1) |
+------------------+
| G                |
| S                |
| T                |
| V                |
| V                |
| R                |
| A                |
| A                |
+------------------+
8 rows in set (0.035 sec)
```

B. From the Search studentstable, surnames and first names combined according to the following model: Guru, Ken. Set the column header to Kokonimi.

The CONCAT() function adds two or more expressions together.

Syntax

```
CONCAT(expression1, expression2, expression3,...)
```

```
SELECT concat(firstname,', ', lastname) AS fullname
FROM students;
```

```
MariaDB [AD2008]> SELECT concat(firstname,', ', lastname) AS fullname
    -> FROM students;
+---------------+
| fullname      |
+---------------+
| Ken, Guru     |
| Tino, Saurus  |
| Sini, Tiainen |
| Vilja, Vainio |
| Elo, Vainio   |
| Muu, Rahainen |
| Kim, Alainen  |
| Ruut, Ana     |
+---------------+
8 rows in set (0.033 sec)
```

C. studentsboard, A username is created for the students of the C. studentsboard, which takes the first 4 characters of the last name and the first 4 characters of the first name. All characters must be in lowercase characters (gemena, lowercase). If the surname or first name does not have four characters, then the number of characters is increased to four by adding characters to the end of the partial string xso that the requirement of 4 characters must be met. Examples: Saurus Tino-> saurtino, Ana Ruut-> anaxruut. Make a SQL statement that prints all yo. user IDs according to the requirements.

The RPAD() function right-pads a string with another string, to a certain length.

Also LPAD() function.

Syntax

```
RPAD(string, length, rpad_string)
```

The LOWER() function converts a string to lower-case.

Note: The LCASE() function is equal to the LOWER() function.

Syntax

```
LOWER(text)
```

LOWER(columname)

```
SELECT CONCAT(RPAD(LOWER(lastname), 4, 'x'),
RPAD(LOWER(firstname), 4, 'x')) AS username
FROM students;
```

```
MariaDB [AD2008]> SELECT CONCAT(RPAD(LOWER(lastname), 4, 'x'),
    -> RPAD(LOWER(firstname), 4, 'x')) AS username
    -> FROM students;
+----------+
| username |
+----------+
| gurukenx |
| saurtino |
| tiaisini |
| vainvilj |
| vainelox |
| rahamuux |
| alaikimx |
| anaxruut |
+----------+
8 rows in set (0.048 sec)
```

C. Apply for students whose birthday is 11/11/2004, 11/11/2005 or 11/11/2006

```
SELECT * FROM students WHERE birthdate = '2004-11-11' OR
birthdate = '2005-11-11' or birthdate = '2006-11-11';
```

```
MariaDB [AD2008]> SELECT * FROM students WHERE birthdate = '2004-11-11' OR
    -> birthdate = '2005-11-11' or birthdate = '2006-11-11';
+-----------+----------+-----------+------------+----------+----------+---------+----------+
| studentID | lastname | firstname | birthdate  | eyecolor | incomes  | taxrate | hometown |
+-----------+----------+-----------+------------+----------+----------+---------+----------+
|      2004 | Vainio   | Vilja     | 2004-11-11 | Sininen  |     0.00 |    0.00 |        3 |
|      2005 | Vainio   | Elo       | 2005-11-11 | Sininen  |     0.00 |    0.00 |        3 |
|      2006 | Rahainen | Muu       | 2006-11-11 | Harmaa   | 13010.12 |    5.80 |        2 |
+-----------+----------+-----------+------------+----------+----------+---------+----------+
3 rows in set (0.032 sec)
```

Task 3 [5p/5p]

A. Calculate the average income of students by home municipality. Enter column headings Kotikunnan IDand KAtulot. You can only retrieve this information from the studentstable. The homeless can be a separate group. Sort the results from highest to lowest by revenue averages.

The `GROUP BY` statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

The `GROUP BY` statement is often used with aggregate functions (`COUNT()`, `MAX()`, `MIN()`, `SUM()`, `AVG()`) to group the result-set by one or more columns.

GROUP BY Syntax

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

```
SELECT hometown AS KotikunnanID, AVG(incomes) as KAtulot
FROM students
GROUP BY hometown
ORDER BY KAtulot DESC;
```

```
+--------------+--------------+
| KotikunnanID | KAtulot      |
+--------------+--------------+
|         NULL | 20010.120000 |
|            2 | 15510.120000 |
|            1 | 14010.220000 |
|            3 |     0.000000 |
+--------------+--------------+
4 rows in set (0.037 sec)
```

B. X-FactorAnother character known as the surname factor has been found to predict a student's academic success. Now calculate the average income of the students grouped by this X-Factorfactor. The result should also show the number of students per each X-Factorfactor. Sort the average income in descending order..

- The SUBSTRING() function extracts a substring from a string (starting at any position).

Note: The SUBSTR() and MID() functions equals to the SUBSTRING() function.

Syntax

```
SUBSTRING(string, start, length)
```

OR: substring(lastname, 2, 1) OR:

```
SUBSTRING(string FROM start FOR length)
```

- The COUNT() function returns the number of records returned by a select query.

Note: NULL values are not counted.

Syntax

```
COUNT(expression)
```

```
SELECT substring(lastname, 2, 1) AS 'surnamefactor',
count(substring(lastname, 2, 1)) AS 'studentnumber',
AVG(incomes) AS averageincomes
FROM students GROUP BY surnamefactor
ORDER BY averageincomes DESC;
```

```
MariaDB [AD2008]> SELECT substring(lastname, 2, 1) AS 'surnamefactor',
    -> count(substring(lastname, 2, 1)) AS 'studentnumber',
    -> AVG(incomes) AS averageincomes
    -> FROM students GROUP BY surnamefactor
    -> ORDER BY averageincomes DESC;
+---------------+---------------+----------------+
| surnamefactor | studentnumber | averageincomes |
+---------------+---------------+----------------+
| n             |             1 |   20010.120000 |
| l             |             1 |   18010.120000 |
| i             |             1 |   16010.320000 |
| u             |             1 |   12010.120000 |
| a             |             4 |    6755.085000 |
+---------------+---------------+----------------+
5 rows in set (0.060 sec)
```

C. Calculate student taxes using income and tax rate. Result columns: last name, first name, income, tax rate and tax. Sort by calculated tax in descending order.

```
SELECT lastname, firstname, incomes, taxrate,
incomes * (taxrate /100) AS studenttax
FROM students
ORDER BY studenttax DESC;
```

```
MariaDB [AD2008]> SELECT lastname, firstname, incomes, taxrate,
    -> incomes * (taxrate /100) AS studenttax
    -> FROM students
    -> ORDER BY studenttax DESC;
+----------+-----------+----------+---------+---------------+
| lastname | firstname | incomes  | taxrate | studenttax    |
+----------+-----------+----------+---------+---------------+
| Ana      | Ruut      | 20010.12 |    9.90 | 1981.00188000 |
| Alainen  | Kim       | 18010.12 |    8.80 | 1584.89056000 |
| Tiainen  | Sini      | 16010.32 |    7.30 | 1168.75336000 |
| Saurus   | Tino      | 14010.22 |    6.20 |  868.63364000 |
| Rahainen | Muu       | 13010.12 |    5.80 |  754.58696000 |
| Guru     | Ken       | 12010.12 |    5.10 |  612.51612000 |
| Vainio   | Vilja     |     0.00 |    0.00 |    0.00000000 |
| Vainio   | Elo       |     0.00 |    0.00 |    0.00000000 |
+----------+-----------+----------+---------+---------------+
8 rows in set (0.048 sec)
```

D. What is the difference between the population of the largest and the smallest city? By how many percent is the largest population greater than the smallest population? Use an SQL statement in which the largest and smallest number of residents, the difference between them as a number and the difference as a percentage are printed on one line. There is no need to print the names of the cities.

- The difference is calculated by difference between max and min population
- Percentage difference = ((max- min)/ max) * 100

```
SELECT MAX(population), MIN(population),
MAX(population)- MIN(population) AS difference,
((MAX(population)- MIN(population))/MAX(population))* 100 AS
percentagedifference
FROM cities;
```

```
MariaDB [AD2008]> SELECT MAX(population), MIN(population),
    -> MAX(population)- MIN(population) AS difference,
    -> ((MAX(population)- MIN(population))/MAX(population))* 100 AS percentagedifference
    -> FROM cities;
+-----------------+-----------------+------------+----------------------+
| MAX(population) | MIN(population) | difference | percentagedifference |
+-----------------+-----------------+------------+----------------------+
|          230000 |          120000 |     110000 |              47.8261 |
+-----------------+-----------------+------------+----------------------+
1 row in set (0.047 sec)
```

E. Find the names and population numbers of all cities whose population would exceed 200,000 after a 10 percent increase.

- Population increased by 10 % is calculated by formula population + population *10%

```
SELECT cityname, population + (population * 10/100) AS
increasedpopulation
FROM cities
HAVING increasedpopulation > 200000;
```

```
MariaDB [AD2008]> SELECT cityname, population + (population * 10/100) AS increasedpopulation
    -> FROM cities
    -> HAVING increasedpopulation > 200000;
+----------+---------------------+
| cityname | increasedpopulation |
+----------+---------------------+
| Turku    |         209000.0000 |
| Tampere  |         253000.0000 |
+----------+---------------------+
2 rows in set (0.034 sec)
```

Task 4 [4p/4p]

A. Search for the completed study courses of the student 2003 (studentID) with grades. There is no need to print the student's name. So only two boards are needed.

1. (INNER) JOIN: Returns records that have matching values in both tables

INNER JOIN	SELECT column_name(s) FROM table_name1 INNER JOIN table_name2 ON table_name1.column_name=table_name2.column_name

2. Where condition to retrieve only requried details

```
SELECT s.studentID, s.birthdate, s.eyecolor, s.incomes,
s.taxrate, s.hometown, sg.courseID, sg.date_created, sg.grade
FROM students s
JOIN studentgrades sg
ON s.studentID = sg.studentID
WHERE s.studentID= 2003;
```

```
MariaDB [AD2008]> SELECT s.studentID, s.birthdate, s.eyecolor, s.incomes,
    -> s.taxrate, s.hometown, sg.courseID, sg.date_created, sg.grade
    -> FROM students s
    -> JOIN studentgrades sg
    -> ON s.studentID = sg.studentID
    -> WHERE s.studentID= 2003;
+-----------+------------+----------+----------+---------+----------+----------+--------------+-------+
| studentID | birthdate  | eyecolor | incomes  | taxrate | hometown | courseID | date_created | grade |
+-----------+------------+----------+----------+---------+----------+----------+--------------+-------+
|      2003 | 2003-11-11 | Sininen  | 16010.32 |    7.30 |        1 | 1        | 2018-11-11   |     3 |
|      2003 | 2003-11-11 | Sininen  | 16010.32 |    7.30 |        1 | 2        | 2019-11-11   |     4 |
|      2003 | 2003-11-11 | Sininen  | 16010.32 |    7.30 |        1 | 3        | 2020-11-11   |     4 |
+-----------+------------+----------+----------+---------+----------+----------+--------------+-------+
3 rows in set (0.040 sec)
```

B. Search for the 8 most recent course completions, sorted from newest to oldest completion. The lines must show the name of the course, the completion date in Finnish format, e.g. 17.02.2021, the student ID studentIDand the grade of the course.

1. MySQL supports the LIMIT clause to select a limited number of records.

MySQL Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE condition
LIMIT number;
```

2. The DATE_FORMAT() function formats a date as specified.

Syntax

```
DATE_FORMAT(date, format)
```

Syntax: DATE_FORMAT(column name, format)

3. (INNER) JOIN: Returns records that have matching values in both tables

INNER JOIN	SELECT column_name(s) FROM table_name1 INNER JOIN table_name2 ON table_name1.column_name=table_name2.column_name

```
SELECT c.coursename, DATE_FORMAT(sg.date_created, '%d.%m.%Y') AS
datecreated,
sg.studentID, sg.grade
FROM courses c
JOIN studentgrades sg
ON c.courseID = sg.courseID
ORDER BY date_created desc
LIMIT 8;
```

```
MariaDB [AD2008]> SELECT c.coursename, DATE_FORMAT(sg.date_created, '%d.%m.%Y') AS datecreated,
    -> sg.studentID, sg.grade
    -> FROM courses c
    -> JOIN studentgrades sg
    -> ON c.courseID = sg.courseID
    -> ORDER BY date_created desc
    -> LIMIT 8;
+-------------+-------------+-----------+-------+
| coursename  | datecreated | studentID | grade |
+-------------+-------------+-----------+-------+
| Ruotsi      | 11.11.2020  |      2004 |     1 |
| Ruotsi      | 11.11.2020  |      2003 |     4 |
| Ruotsi      | 11.11.2020  |      2002 |     4 |
| Ruotsi      | 11.11.2020  |      2001 |     5 |
| Ruotsi      | 11.11.2020  |      2006 |     3 |
| Ruotsi      | 11.11.2020  |      2005 |     1 |
| Tietokannat | 11.11.2019  |      2006 |     2 |
| Tietokannat | 11.11.2019  |      2003 |     4 |
+-------------+-------------+-----------+-------+
8 rows in set (0.036 sec)
```

C. Search by student for the average of all their course completions. Print the students' last name, first name and calculated average.

1. (INNER) JOIN: Returns records that have matching values in both tables

INNER JOIN

```
SELECT column_name(s)
FROM table_name1
INNER JOIN table_name2
ON table_name1.column_name=table_name2.column_name
```

2. *AVERAGE + GROUP BY + JOIN*

```
SELECT s.lastname, s.firstname,
SUM(sg.grade) / COUNT(sg.studentID) AS keskiarvo
FROM students s
JOIN studentgrades sg
ON s.studentID = sg.studentID
GROUP BY sg.studentID;
```

```
MariaDB [AD2008]> SELECT s.lastname, s.firstname,
    -> SUM(sg.grade) / COUNT(sg.studentID) AS keskiarvo
    -> FROM students s
    -> JOIN studentgrades sg
    -> ON s.studentID = sg.studentID
    -> GROUP BY sg.studentID;
+----------+-----------+-----------+
| lastname | firstname | keskiarvo |
+----------+-----------+-----------+
| Guru     | Ken       |    5.0000 |
| Saurus   | Tino      |    4.0000 |
| Tiainen  | Sini      |    3.6667 |
| Vainio   | Vilja     |    1.0000 |
| Vainio   | Elo       |    1.0000 |
| Rahainen | Muu       |    2.3333 |
| Alainen  | Kim       |    3.5000 |
| Ana      | Ruut      |    4.5000 |
+----------+-----------+-----------+
8 rows in set (0.046 sec)
```

D. Search for the average of all course completions by study period. Print the name of the students' course and the calculated average.

1) `(INNER) JOIN`: Returns records that have matching values in both tables

INNER JOIN

```
SELECT column_name(s)
FROM table_name1
INNER JOIN table_name2
ON table_name1.column_name=table_name2.column_name
```

2) *AVERAGE + GROUP BY + JOIN*

```
SELECT c.coursename,
SUM(sg.grade) / COUNT(sg.date_created) AS keskiarvo
FROM courses c
JOIN studentgrades sg
ON c.courseID = sg.courseID
GROUP BY date_created;
```

```
MariaDB [AD2008]> SELECT c.coursename,
    -> SUM(sg.grade) / COUNT(sg.date_created) AS keskiarvo
    -> FROM courses c
    -> JOIN studentgrades sg
    -> ON c.courseID = sg.courseID
    -> GROUP BY date_created;
+-------------+-----------+
| coursename  | keskiarvo |
+-------------+-----------+
| Ohjelmointi |    3.5000 |
| Tietokannat |    4.0000 |
| Ruotsi      |    3.0000 |
+-------------+-----------+
3 rows in set (0.034 sec)
```

Task 5 [6p/6p]

A. Find the average population numbers by eye color. Include in your group also those whose eye color is unknown.

```
SELECT s.eyecolor, AVG(c.population) as KA
FROM students s
JOIN cities c ON s.hometown = c.cityID
group BY s.eyecolor;
```

```
MariaDB [AD2008]> SELECT s.eyecolor, AVG(c.population) as KA
    -> FROM students s
    -> JOIN cities c ON s.hometown = c.cityID
    -> group BY s.eyecolor;
+----------+-------------+
| eyecolor | KA          |
+----------+-------------+
| NULL     | 230000.0000 |
| Harmaa   | 230000.0000 |
| Ruskea   | 190000.0000 |
| Sininen  | 143333.3333 |
+----------+-------------+
4 rows in set (0.044 sec)
```

B. [2p] Search for the average of all study course completions by hometown and study course

1. The `GROUP BY` statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

2. The `GROUP BY` statement is often used with aggregate functions (`COUNT()`, `MAX()`, `MIN()`, `SUM()`, `AVG()`) to group the result-set by one or more columns.

GROUP BY Syntax

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

3. Using an INNER JOIN / JOIN with two, three, four, or many more tables is possible. You simply add the INNER JOIN keyword to the end of the join criteria for the previous join.

The syntax looks like this:

```
SELECT your_columnsFROM table1INNER JOIN table2 ON table1.col1 = table2.col1INNER JOIN table3 ON
table2.col2 = table3.col2;
```

```
SELECT ci.cityname, co.coursename, avg(sg.grade) AS KA
FROM cities ci
JOIN students s
ON ci.cityID = s.hometown
JOIN studentgrades sg
ON s.studentID = sg.studentID
```

```
JOIN courses co
ON sg.courseID = co.courseID
GROUP BY co.coursename,ci.cityname
ORDER BY cityname desc;
```

```
MariaDB [AD2008]> SELECT ci.cityname, co.coursename, avg(sg.grade) AS KA
    -> FROM cities ci
    -> JOIN students s
    -> ON ci.cityID = s.hometown
    -> JOIN studentgrades sg
    -> ON s.studentID = sg.studentID
    -> JOIN courses co
    -> ON sg.courseID = co.courseID
    -> GROUP BY co.coursename,ci.cityname
    -> ORDER BY cityname desc;
+----------+-------------+--------+
| cityname | coursename  | KA     |
+----------+-------------+--------+
| Turku    | Ohjelmointi | 4.0000 |
| Turku    | Tietokannat | 4.3333 |
| Turku    | Ruotsi      | 4.3333 |
| Tampere  | Ohjelmointi | 2.5000 |
| Tampere  | Tietokannat | 3.0000 |
| Tampere  | Ruotsi      | 3.0000 |
| Lahti    | Ruotsi      | 1.0000 |
+----------+-------------+--------+
7 rows in set (0.037 sec)
```

D. [2p] Search for students with their income so that the additional column contains an entry pienituloinenfor those whose income is 15,000 or less and the others enter an entry in this column isotuloinen. Sort from highest income to lowest income.

1. The IF() function returns a value if a condition is TRUE, or another value if a condition is FALSE.

Syntax

```
IF(condition, value_if_true, value_if_false)
```

```
SELECT lastname, firstname, incomes,
if(incomes <= 15000, 'pienituloinen', 'isotuloinen') AS tuloluokka
FROM students
ORDER BY incomes desc;
```

```
MariaDB [AD2008]> SELECT lastname, firstname, incomes,
    -> if(incomes <= 15000, 'pienituloinen', 'isotuloinen') AS tuloluokka
    -> FROM students
    -> ORDER BY incomes desc;
+----------+-----------+----------+---------------+
| lastname | firstname | incomes  | tuloluokka    |
+----------+-----------+----------+---------------+
| Ana      | Ruut      | 20010.12 | isotuloinen   |
| Alainen  | Kim       | 18010.12 | isotuloinen   |
| Tiainen  | Sini      | 16010.32 | isotuloinen   |
| Saurus   | Tino      | 14010.22 | pienituloinen |
| Rahainen | Muu       | 13010.12 | pienituloinen |
| Guru     | Ken       | 12010.12 | pienituloinen |
| Vainio   | Vilja     |     0.00 | pienituloinen |
| Vainio   | Elo       |     0.00 | pienituloinen |
+----------+-----------+----------+---------------+
8 rows in set (0.044 sec)
```