Databases: Exercises 7 (15p/15p)

Task 1 [2p/2p]

A. Make a view of all Turku students `Turku`from the who applies board. `students`From the columns, the following are printed in order: `firstname`, `lastname`, `eyecolor`and `incomes`. Test the functionality of the view by trying it out

ANSWER:

In SQL, a view is a virtual table based on the result-set of an SQL statement.

A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

You can add SQL statements and functions to a view and present the data as if the data were coming from one single table.

A view is created with the `CREATE VIEW` statement.

CREATE VIEW Syntax

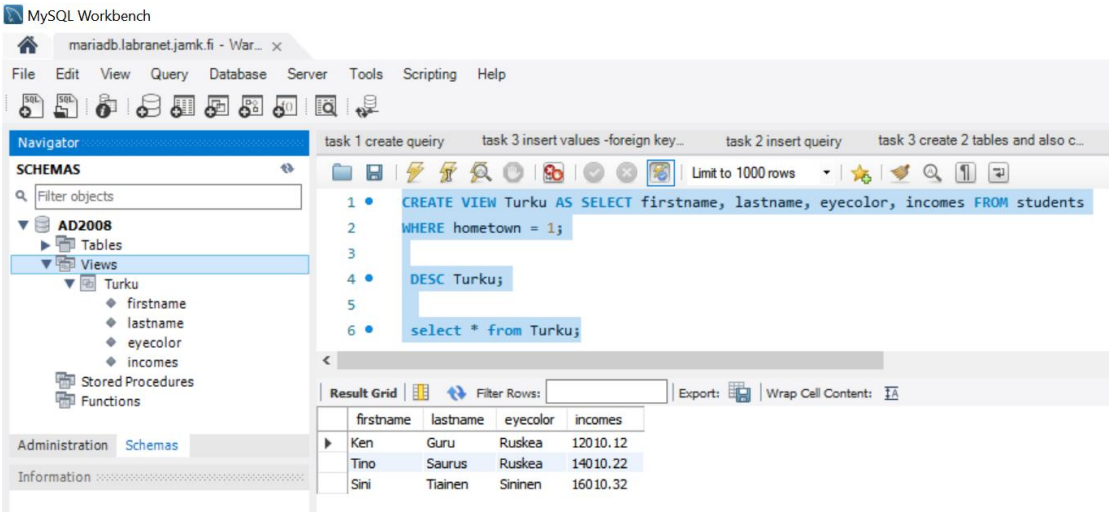
```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

- The following statement is used to create a view and the detail result information is in the screenshot.

```
CREATE VIEW Turku AS SELECT firstname, lastname, eyecolor, incomes
FROM students
WHERE hometown = 1;

 DESC Turku;

 select * from Turku;
```



B. *Create a view* `Oliot` *that prints the* `students`*names of the students in the All table in one column in the format FirstNameLastName . E.g. Kangaroo , Tinosaurus , etc. Only the first character can be printed in capitals (uppercase, uppercase). The column title should also be Oliot . When using the view, the objects should be printed in alphabetical order. Test the functionality of the view by trying it out. The creation of the view and its testing must be visible in the answer.*

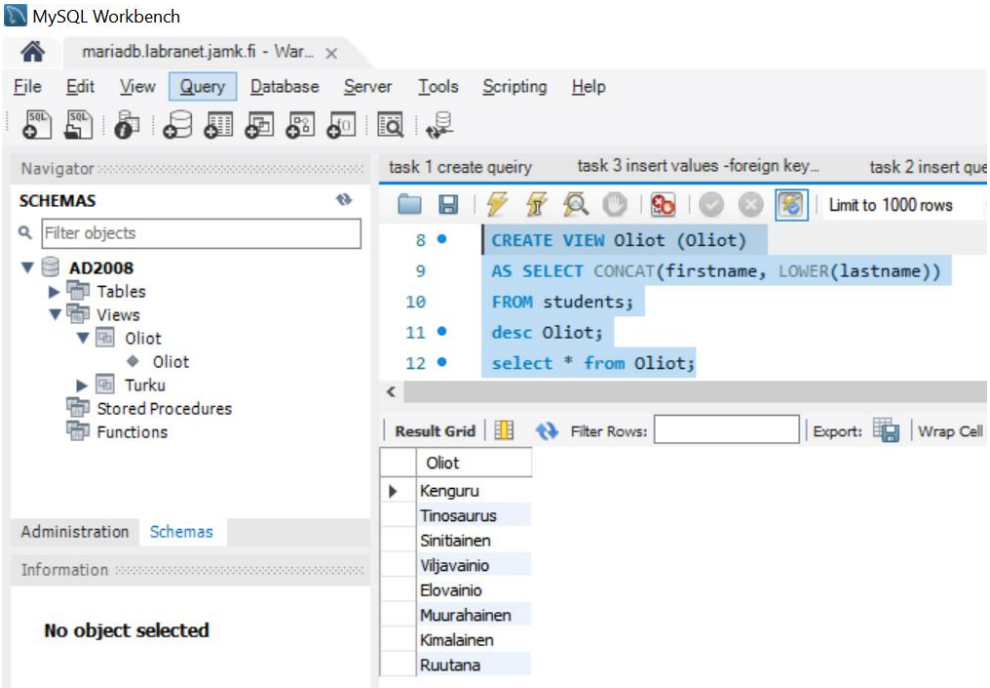
ANSWER:

- A view is created with the `CREATE VIEW` statement.
- The CONCAT() function adds two or more expressions together.
- The LOWER() function converts a string to lower-case.
- The following statement is used to create a view and the detail result information is in the screenshot

```
CREATE VIEW Oliot (Oliot)
AS SELECT CONCAT(firstname, LOWER(lastname))
FROM students;

desc Oliot;

select * from Oliot;
```



Task 2 [4p/4p]

A. *[2p] Make a view* `KaupunkiKeskiarvoTulot` *that prints the average income of students by city. The printout shows the name of each city and the city-specific average income , and the column headings should be City and KAtulot . Non-hometowners are not included in this view. When using the view, cities should be sorted based on average income in descending order. Test the functionality of the view by trying it out. The creation of the view and its testing must be visible in the answer.*

ANSWER:

- A view is created with multiple tables ,the `CREATE VIEW` statement and where condition to indicate the comman column name in multiple tables
- The `AVG()` function returns the average value of a numeric column.
- The `GROUP BY` statement groups rows that have the same values into summary rows
- The `ORDER BY` keyword sorts the records in ascending order by default. To sort the records in descending order, use the `DESC` keyword

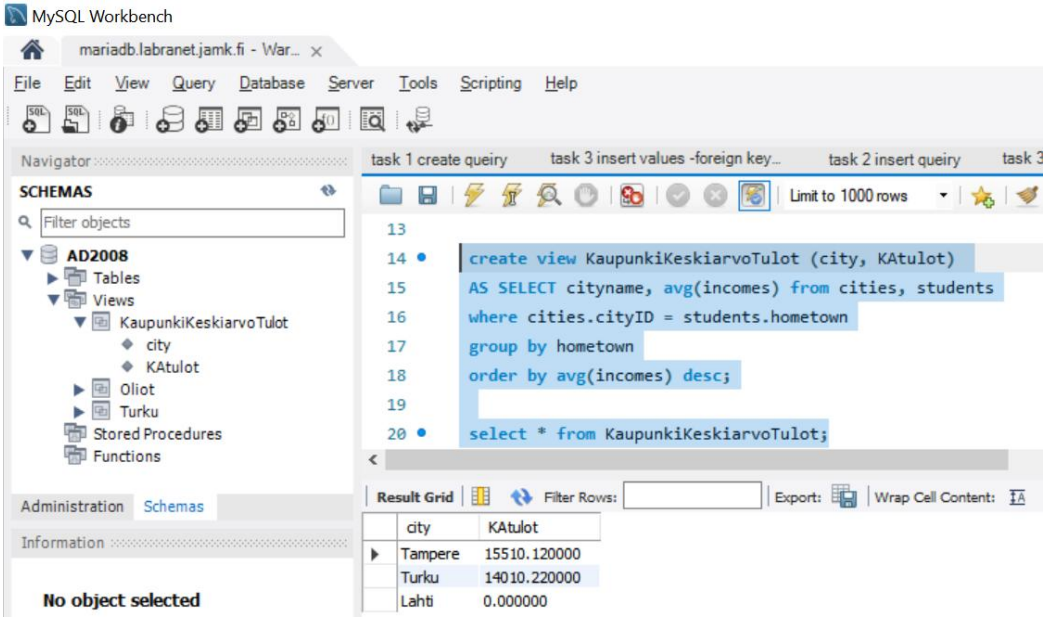
CREATE VIEW Syntax for multiple tables

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name1,table_name2,…
WHERE table_name1.columnname = table_name2.columnname
and condition
```

- The following statement is used to create a view and the detail result information is in the screenshot

```
create view KaupunkiKeskiarvoTulot (city, KAtulot)
AS SELECT cityname, avg(incomes) from cities, students
where cities.cityID = students.hometown
group by hometown
order by avg(incomes) desc;

select * from KaupunkiKeskiarvoTulot;
```



B. *[2p] Make a view `Stipendiehdokkaat` that can be used to print the 4 best transcripts. Row by row, the student's last name, first name, course and course grade are displayed, as in the model below. In addition to the grade, the sorting criteria are in order course name (ASC), last name (ASC) and first name (ASC). Test the functionality of the view by trying it out. The creation of the view and its testing must be visible in the answer.*

ANSWER: A view is created with multiple tables ,the `CREATE VIEW` statement and where condition to indicate the common column name in multiple tables

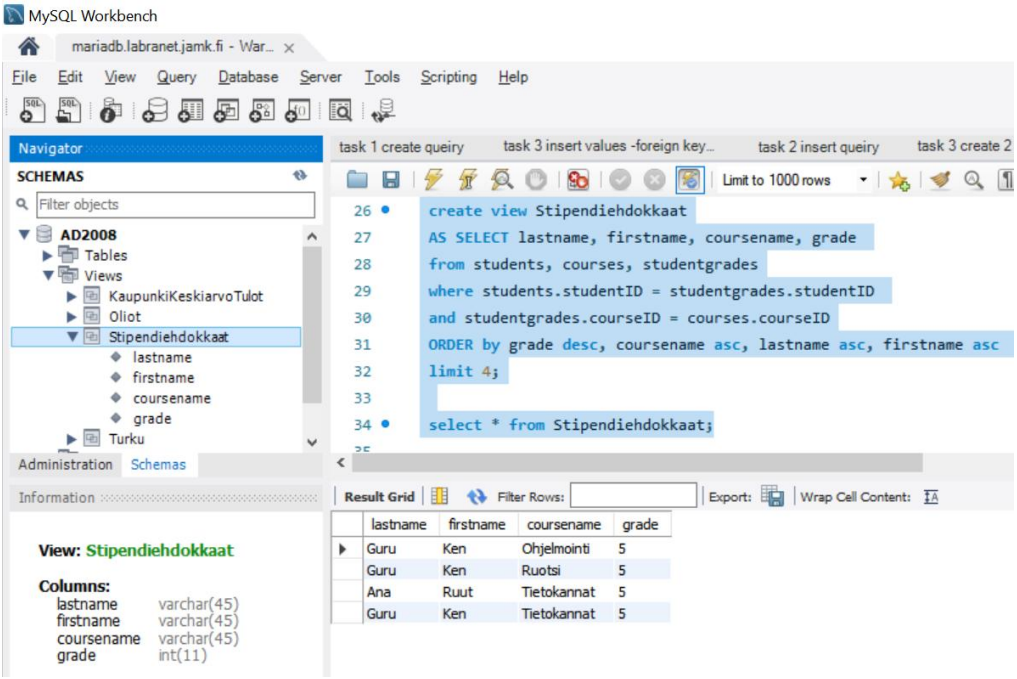
CREATE VIEW Syntax for multiple tables

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name1,table_name2,…
WHERE table_name1.columnname = table_name2.columnname
and table_name2.columnname = table_name3.columnname
And condition
```

- The `SELECT TOP` clause is used to specify the number of records to return.
- MySQL supports the `LIMIT` clause to select a limited number of records.
- The `ORDER BY` keyword sorts the records in ascending order by default. To sort the records in descending order, use the `DESC` keyword. We can set the order according to priority and mentioned for each column.
- The following statement is used to create a view and the detail result information is in the screenshot

```
create view Stipendiehdokkaat
AS SELECT lastname, firstname, coursename, grade
from students, courses, studentgrades
where students.studentID = studentgrades.studentID
and studentgrades.courseID = courses.courseID
ORDER by grade desc, coursename asc, lastname asc, firstname asc
limit 4;

select * from Stipendiehdokkaat;
```



Task 3 [3p/3p]

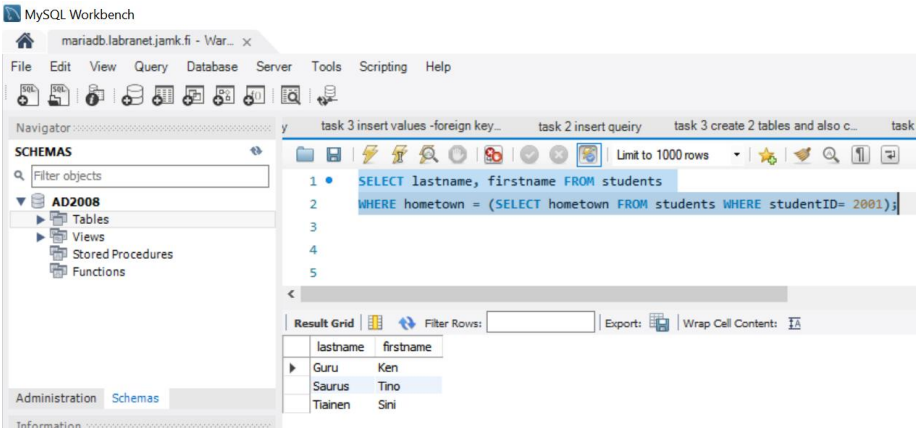
A. *Search for the students (with last and first name) who have the same hometown as Ken Guru. Ken Guru's hometown is unknown at the time of writing the survey (it could be anything). Use a subquery.*

ANSWER:

- comparison operator "=": the subquery can return only one value

The following statement is used to create a subquery with **'=' operator,** inner query gives single row output to outer query and the detail result information is in the screenshot

```
SELECT lastname, firstname FROM students
WHERE hometown = (SELECT hometown FROM students WHERE studentID=
2001);
```



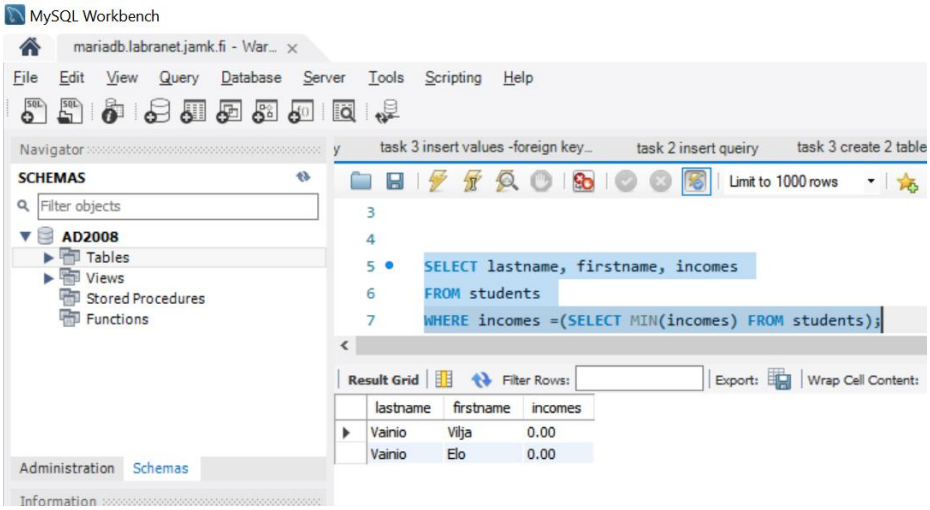
B. *Apply to the lowest income students. The minimum salary is not known at the time of writing the survey. The result columns are surname and first name and income. Use a subquery*

ANSWER:

- comparison operator "=": the subquery can return only one value

The following statement is used to create a subquery with **'=' operator,** inner query gives single row output to outer query and the detail result information is in the screenshot

```
SELECT lastname, firstname, incomes
FROM students
WHERE incomes =(SELECT MIN(incomes) FROM students);
```

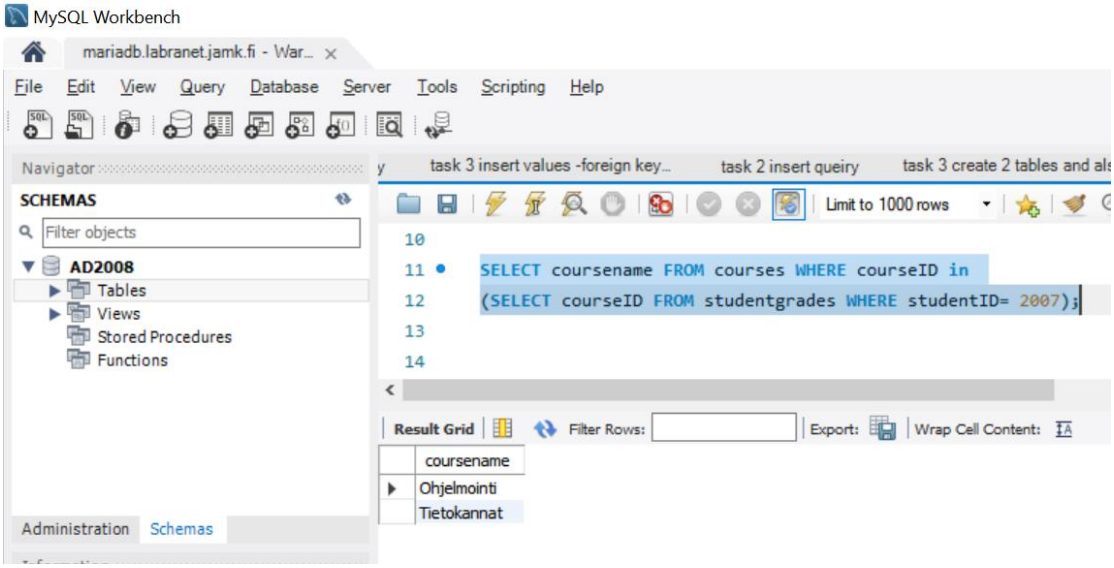


C. Search for the names of the completed study courses of student 2007 (= studentID). Use a subquery.

ANSWER:

- The IN operator or comparison operator with an ANY or ALL condition allows a subquery to return multiple values
- The following statement is used to create a subquery with **IN operator** and the detail result information is in the screenshot

```
SELECT coursename FROM courses WHERE courseID in
(SELECT courseID FROM studentgrades WHERE studentID= 2007);
```



Task 4 [4p/4p]

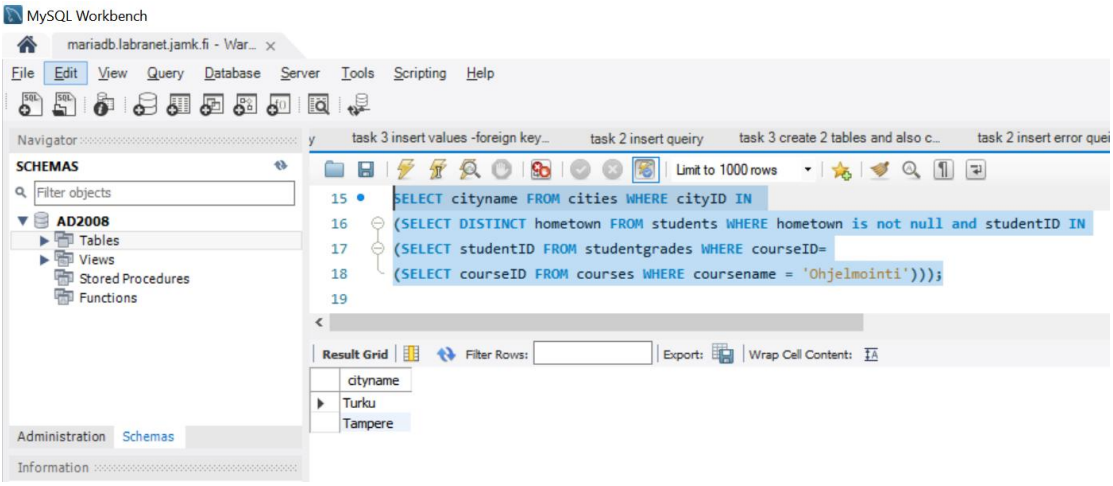
A. *[2p] Search for the names of the cities where at least one of the students has completed the Programming course.*

ANSWER:

- A subquery can be nested inside other subqueries. SQL has an ability to nest queries within one another.
- A subquery is a SELECT statement that is nested within another SELECT statement and which return intermediate results.
- SQL executes innermost subquery first, then next level.

- The following statement is used to create a nested subquery with **IN operator** and **'=' operator** the detail result information is in the screenshot.

```
SELECT cityname FROM cities WHERE cityID IN
(SELECT DISTINCT hometown FROM students WHERE hometown is not null
and studentID IN
(SELECT studentID FROM studentgrades WHERE courseID=
(SELECT courseID FROM courses WHERE coursename = 'Ohjelmointi')));
```



B. [2p] Search for the names of completed study courses of the student 2007 (= studentID) using the **`EXISTS`***subquery.*

ANSWER:

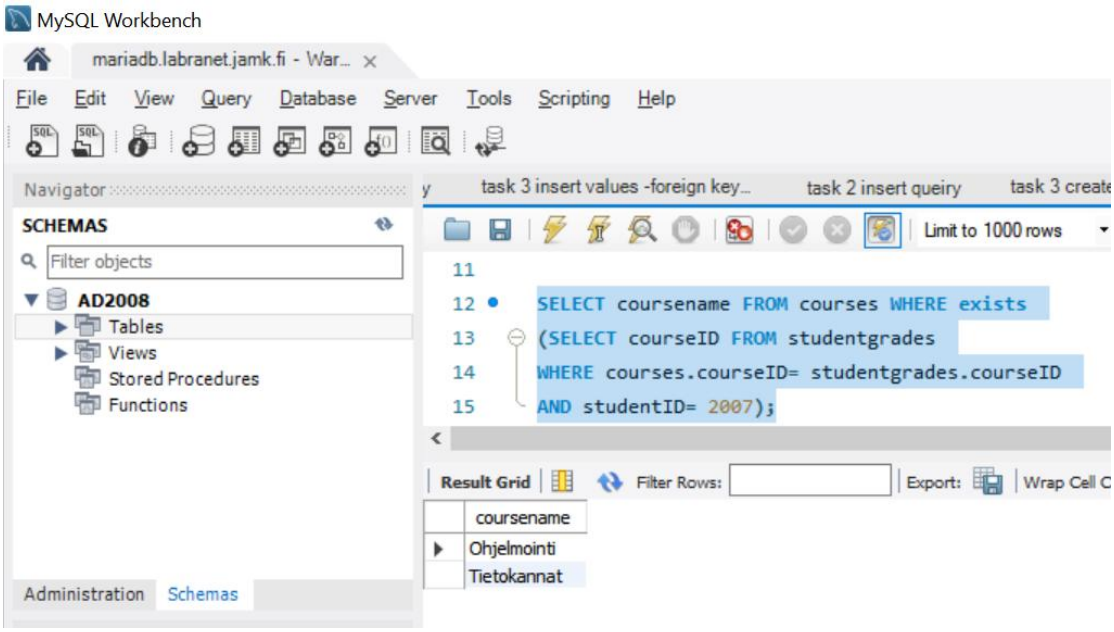
- The `EXISTS` operator is used to test for the existence of any record in a subquery.
- The `EXISTS` operator returns TRUE if the subquery returns one or more records.

EXISTS Syntax

```
SELECT column_name(s)
FROM table_name
WHERE EXISTS
(SELECT column_name FROM table_name WHERE condition);
```

- The following statement is used to create a subquery with **EXISTS operator** and the detail result information is in the screenshot

```
SELECT coursename FROM courses WHERE exists
(SELECT courseID FROM studentgrades
WHERE courses.courseID= studentgrades.courseID
AND studentID= 2007);
```



Task 5 [2p/2p]

Create a so-called 3-star thick index for a SQL query

```
SELECT lastname, firstname, incomesFROM studentsWHERE hometown = 1ORDER BY taxrate DESC;
```

Rule of thumb for creating an index (so-called 3-star index)

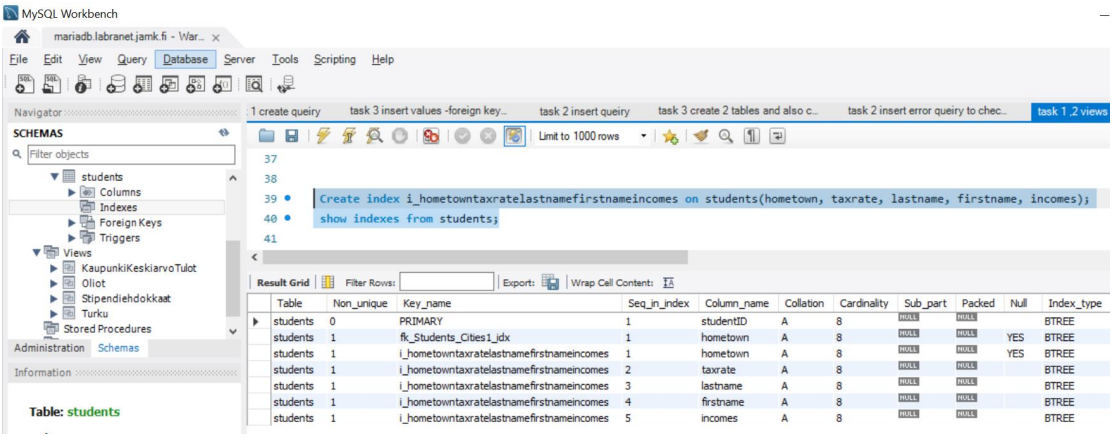
1. *Take all the columns mentioned in the WHERE clause*
2. *Add the columns mentioned in ORDER BY*
3. *Complete the column names in the SELECT statement*

If all columns mentioned in SELECT are included in the index, it is a ***thick index***

- The following statement is used to create thick index and the detail result information is in the screenshot

```
Create index i_hometowntaxratelastnamefirstnameincomes ON
students(hometown, taxrate, lastname, firstname, incomes);

show indexes from students;
```


MySQL Workbench
mariadb.labranet.jamk.fi - War...
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
students
Columns
Indexes
Foreign Keys
Triggers
Views
KaupunkiKeskiarvoTulot
Oliot
Stipendiehdokkaat
Turku
Stored Procedures
Administration Schemas
Information
Table: students
1 create queiry
task 3 insert values -foreign key...
task 2 insert queiry
task 3 create 2 tables and also c...
task 2 insert error queiry to chec...
task 1 ,2 views
Limit to 1000 rows
37
38
39 Create index i_hometowntaxratelastnamefirstnameincomes on students(hometown, taxrate, lastname, firstname, incomes);
40 show indexes from students;
41
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
Table Non_unique Key_name Seq_in_index Column_name Collation Cardinality Sub_part Packed Null Index_type
students 0 PRIMARY 1 studentID A 8 NULL NULL BTREE
students 1 fk_Students_Cities1_idx 1 hometown A 8 NULL NULL YES BTREE
students 1 i_hometowntaxratelastnamefirstnameincomes 1 hometown A 8 NULL NULL YES BTREE
students 1 i_hometowntaxratelastnamefirstnameincomes 2 taxrate A 8 NULL NULL BTREE
students 1 i_hometowntaxratelastnamefirstnameincomes 3 lastname A 8 NULL NULL BTREE
students 1 i_hometowntaxratelastnamefirstnameincomes 4 firstname A 8 NULL NULL BTREE
students 1 i_hometowntaxratelastnamefirstnameincomes 5 incomes A 8 NULL NULL BTREE