CSIT115 Data Management and Security

SELECT Statement (2)

Dr Janusz R. Getta

School of Computing and Information Technology -
University of Wollongong

SELECT statement (2)

Outline

- Queries with simple conditions
- Queries with Boolean expressions
- Set algebra queries
- Sorting
- Queries about lack of values (`NULL`s)
- Grouping
- Grouping with selections

Queries with simple conditions

Sample database

CREATE TABLE statement

```
CREATE TABLE DEPARTMENT(
 name                VARCHAR(50)        NOT NULL,
 code                CHAR(5)            NOT NULL,
 total_staff_number  DECIMAL(2)         NOT NULL,
 chair               VARCHAR(50)            NULL,
 budget              DECIMAL(9,1)       NOT NULL,
  CONSTRAINT dept_pkey PRIMARY KEY(name),
  CONSTRAINT dept_ckey1 UNIQUE(code),
  CONSTRAINT dept_ckey2 UNIQUE(chair),
  CONSTRAINT dept_check1 CHECK (total_staff_number BETWEEN 1 AND 50) );
```

CREATE TABLE statement

```
CREATE TABLE COURSE(
 cnum                CHAR(7)            NOT NULL,
 title               VARCHAR(200)       NOT NULL,
 credits             DECIMAL(2)         NOT NULL,
 offered_by          VARCHAR(50)            NULL,
  CONSTRAINT course_pkey PRIMARY KEY(cnum),
  CONSTRAINT course_check1 CHECK (credits IN (6, 12)),
  CONSTRAINT course_fkey1 FOREIGN KEY(offered_by)
                        REFERENCES DEPARTMENT(name) ON DELETE CASCADE );
```

In HTML view press 'p' to see the lecture notes

Queries with simple conditions

Find the titles of all 6 credit points courses

SELECT statement with a simple equality condition

```
SELECT title
FROM COURSE
WHERE credits = 6;
```

Find the titles of all 6 or 12 credit points courses

SELECT statement with a set membership IN operation

```
SELECT title
FROM COURSE
WHERE credits IN (6, 12);
```

Find the titles and numbers of all courses that have a word "database" in its title

SELECT statement with a pattern matching operation LIKE

```
SELECT title, cnum
FROM COURSE
WHERE UPPER(title) LIKE '%DATABASE%';
```

SELECT statement (2)

Outline

Queries with simple conditions

Queries with Boolean expressions

Set algebra queries

Sorting

Queries about lack of values (`NULL`s)

Grouping

Grouping with selections

Queries with Boolean expressions

Find the titles of all 6 credit points courses offered by a department of Physics

SELECT statement with a Boolean expression

```
SELECT title
FROM COURSE
WHERE (credits = 6) AND (offered_by ='Physics');
```

Find the titles of all 6 credit points courses or the titles of all courses offered by a department of Physics

SELECT statement with a Boolean expression

```
SELECT title
FROM COURSE
WHERE (credits = 6) OR (offered_by ='Physics');
```

Find the titles of all 6 credit points courses that are not offered by a department of Physics

SELECT statement with a Boolean expression

```
SELECT title
FROM COURSE
WHERE NOT (offered_by = 'Physics') AND (credits = 6);
```

Queries with Boolean expressions

Find the titles of all 6 credit points courses that are not offered by a department of Physics

SELECT statement with a Boolean expression

```
SELECT title
FROM COURSE
WHERE (offered_by != 'Physics') AND (credits = 6);
```

Queries with Boolean expressions

Find the titles of all courses offered by a department of Physics or offered by a department of Mathematics

SELECT statement with a Boolean expression

```
SELECT title
FROM COURSE
WHERE (offered_by = 'Physics') OR (offered_by = 'Mathematics');
```

SELECT statement with set membership operation IN (equivalent to a statement above)

```
SELECT title
FROM COURSE
WHERE offered_by IN ('Physics', 'Mathematics');
```

Find the same titles of all courses offered by a department of Physics and offered by a department of Mathematics

SELECT statement with a correct Boolean expression

```
SELECT title
FROM COURSE
WHERE(offered_by = 'Physics') AND (offered_by = 'Mathematics');
```

- WRONG !!!

SELECT statement (2)

Outline

Queries with simple conditions

Queries with Boolean expressions

Set algebra queries

Sorting

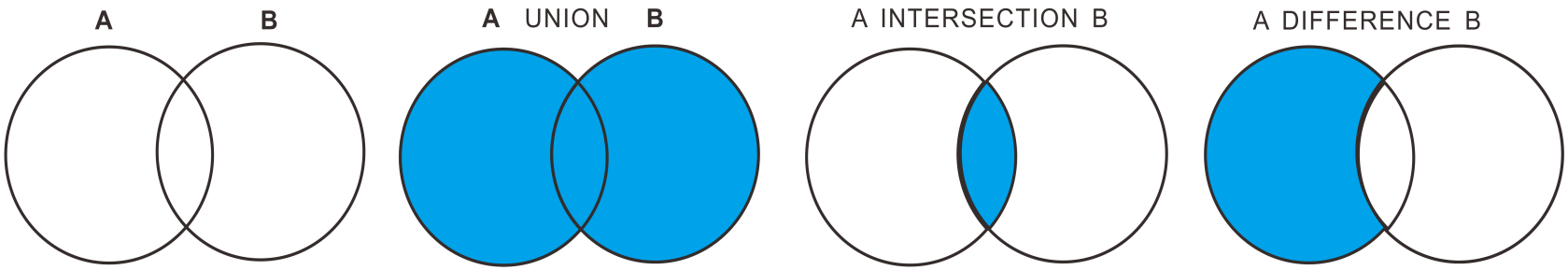
Queries about lack of values (`NULL`s)

Grouping

Grouping with selections

Set algebra queries

Set operations: `UNION`, `INTERSECTION`, and `DIFFERENCE`



A `UNION` B: all elements in A or in B

A `INTERSECTION` B: all elements in A and in B

A `DIFFERENCE` B: all elements in A and not in B

B `DIFFERENCE` A: all elements in B and not in A

Set algebra queries

UNION operation

Find the credit points of courses offered by a department of Physics or a department of Mathematics and do not list duplicated rows

SELECT statement with a Boolean expression

```
SELECT DISTINCT credits
FROM COURSE
WHERE (offered_by = 'Physics') OR (offered_by = 'Mathematics');
```

SELECT statement with a set operation UNION (equivalent to a statement above)

```
SELECT credits
FROM COURSE
WHERE offered_by = 'Physics'
UNION
SELECT credits
FROM COURSE
WHERE offered_by = 'Mathematics';
```

UNION operation automatically eliminates duplicated rows !!!

Set algebra queries

UNION ALL operation

Find the credit points of courses offered by a department of Physics or a department of Mathematics

SELECT statement with a Boolean expression

```
SELECT credits
FROM COURSE
WHERE (offered_by = 'Physics') OR (offered_by = 'Mathematics');
```

SELECT statement with a set operation UNION ALL (equivalent to a statement above)

```
SELECT credits
FROM COURSE
WHERE offered_by = 'Physics'
UNION ALL
SELECT credits
FROM COURSE
WHERE offered_by = 'Mathematics';
```

UNION operation does NOT eliminate duplicated rows !!!

Set algebra queries

What about other set operations like intersection (`INTERSECT`) and difference (`MINUS`) ?

MySQL does not implement `INTERSECT` and `MINUS` set operations :(!!!

So, how do we implement in MySQL queries with `INTERSECT` and `MINUS` set operations ?

Find the credit points that can be earned both for a course offered by Physics and for a course offered by Mathematics

Nested SELECT statement that implements INTERSECT operation

```
SELECT DISTINCT credits
FROM COURSE
WHERE offered_by = 'Physics' AND
      credits IN (SELECT credits
                  FROM COURSE
                  WHERE offered_by = 'Mathematics');
```

Set algebra queries

What about other set operations like intersection (`INTERSECT`) and difference (`MINUS`) ?

MySQL does not implement `INTERSECT` and `MINUS` set operations :(!!!

So, how do we implement in MySQL queries with `INTERSECT` and `MINUS` set operations ?

Find the credit points that can be earned for a course offered by Physics and cannot be earned for a course offered by Mathematics

Nested SELECT statement that implements MINUS operation

```
SELECT DISTINCT credits
FROM COURSE
WHERE offered_by = 'Physics' AND
      credits NOT IN (SELECT credits
                      FROM COURSE
                      WHERE offered_by = 'Mathematics');
```

SELECT statement (2)

Outline

Queries with simple conditions

Queries with Boolean expressions

Set algebra queries

Sorting

Queries about lack of values (`NULL`s)

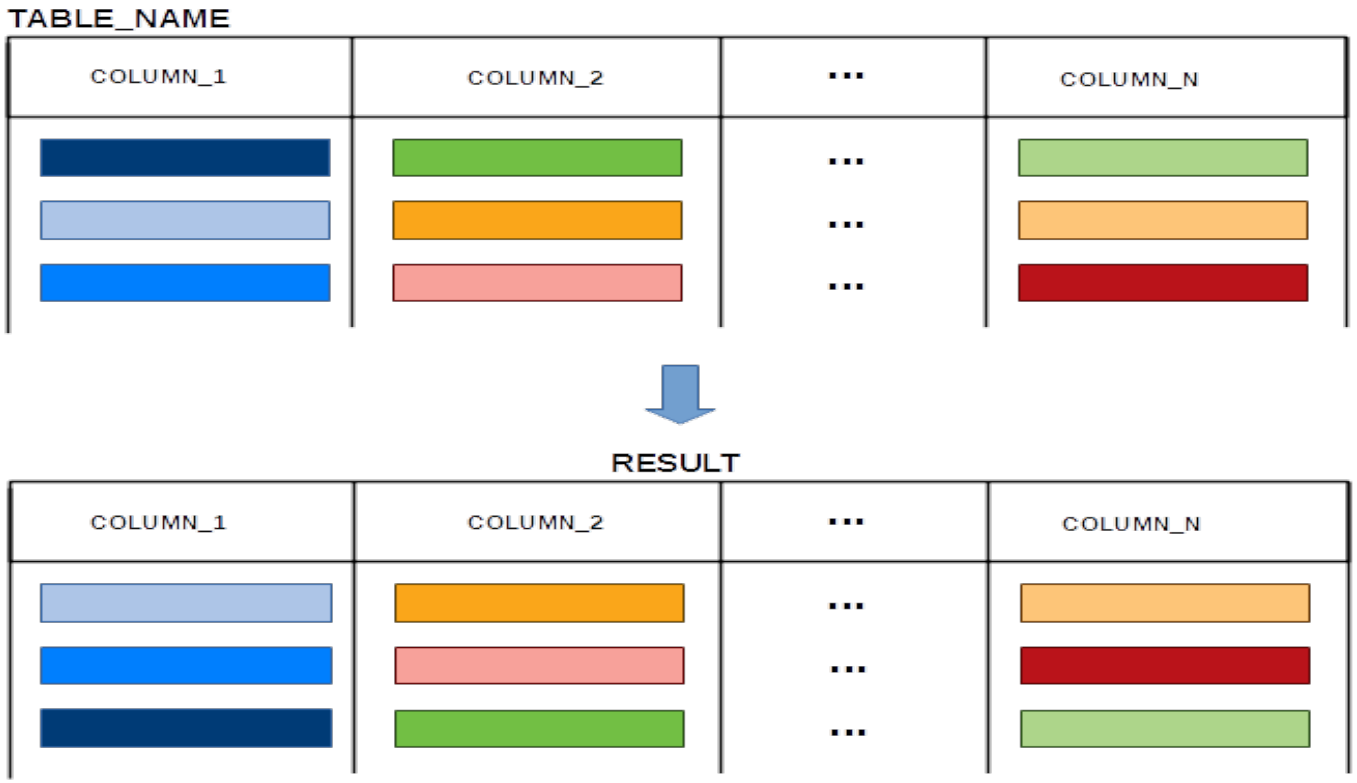
Grouping

Grouping with selections

Sorting

Sorting re-orders the rows in the results if a query

```
SELECT *
FROM TABLE_NAME
ORDER BY COLUMN_1 ASC;
```



In HTML view press 'p' to see the lecture notes

Sorting

Find the numbers and titles of all 6 credit points courses sorted in the ascending order of titles

```
SELECT statement that sorts the results in the ascending order of the values in the title column
SELECT cnum, title
FROM COURSE
WHERE credits = 6
ORDER BY title ASC;
```

```
SELECT statement that sorts the results in the ascending order of the values in the title column
SELECT cnum, title
FROM COURSE
WHERE credits = 6
ORDER BY 2 ASC;
```

Sorting

Find the numbers and titles of all 6 credit point courses sorted in a descending order of titles

```
SELECT statement that sorts the results in the descending order of the values in the title column
SELECT cnum, title
FROM COURSE
WHERE credits = 6
ORDER BY title DESC;
```

```
SELECT statement that sorts the results in the descending order of the values in the title column
SELECT cnum, title
FROM COURSE
WHERE credits = 6
ORDER BY 2 DESC;
```

Sorting

Find the numbers, titles, and credits of all courses sorted in an ascending order of credits and for all courses with the same credits sorted in descending order by titles

```
SELECT statement that sorts the results in the ascending order of the values in the credits column
and in the descending order in the title column
```

```
SELECT cnum, title, credits
FROM COURSE
ORDER BY credits ASC, title DESC;
```

```
SELECT statement that sorts the results in the ascending order of the values in the credits column
and in the descending order in the title column
```

```
SELECT cnum, title, credits
FROM COURSE
ORDER BY 3 ASC, 2 DESC;
```

SELECT statement (2)

Outline

Queries with simple conditions

Queries with Boolean expressions

Set algebra queries

Sorting

Queries about lack of values (`NULL`s)

Grouping

Grouping with selections

Queries about lack of values (NULLs)

Find the titles of all courses which are not offered now

Query about lack of value

```
SELECT title
FROM COURSE
WHERE offered_by IS NULL;
```

Incorrect query about lack of value

```
SELECT title
FROM COURSE
WHERE offered_by = NULL;
```

- WRONG !!! Lack of value cannot be compared with a value

Queries about lack of values (NULLs)

Find the titles of all courses offered now

Query about existence of value

```
SELECT title
FROM COURSE
WHERE offered_by IS NOT NULL;
```

Incorrect query about existence of value

```
SELECT title
FROM COURSE
WHERE offered_by <> NULL;
```

- WRONG !!! Lack of value cannot be compared with a value

SELECT statement (2)

Outline

Queries with simple conditions
Queries with Boolean expressions
Set algebra queries
Sorting
Queries about lack of values (`NULL`s)
Grouping
Grouping with selections

Grouping

Grouping groups the rows before application of aggregation function

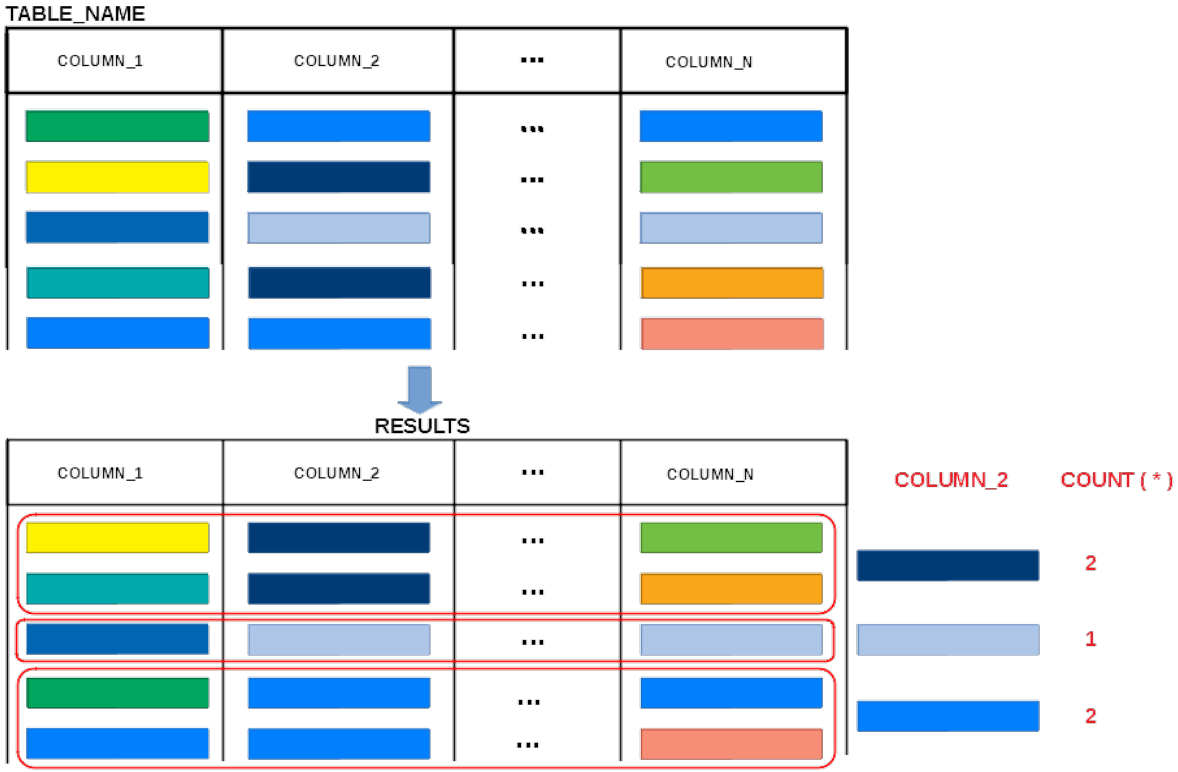
```
SELECT ...
FROM TABLE_NAME
GROUP BY COLUMN_2;
```

In HTML view press 'p' to see the lecture notes

Grouping

Next, aggregation function is applied to each group and the results are associated with the values taken from a column used for grouping

```
SELECT COLUMN_2, COUNT(*)
FROM TABLE_NAME
GROUP BY COLUMN_2;
```

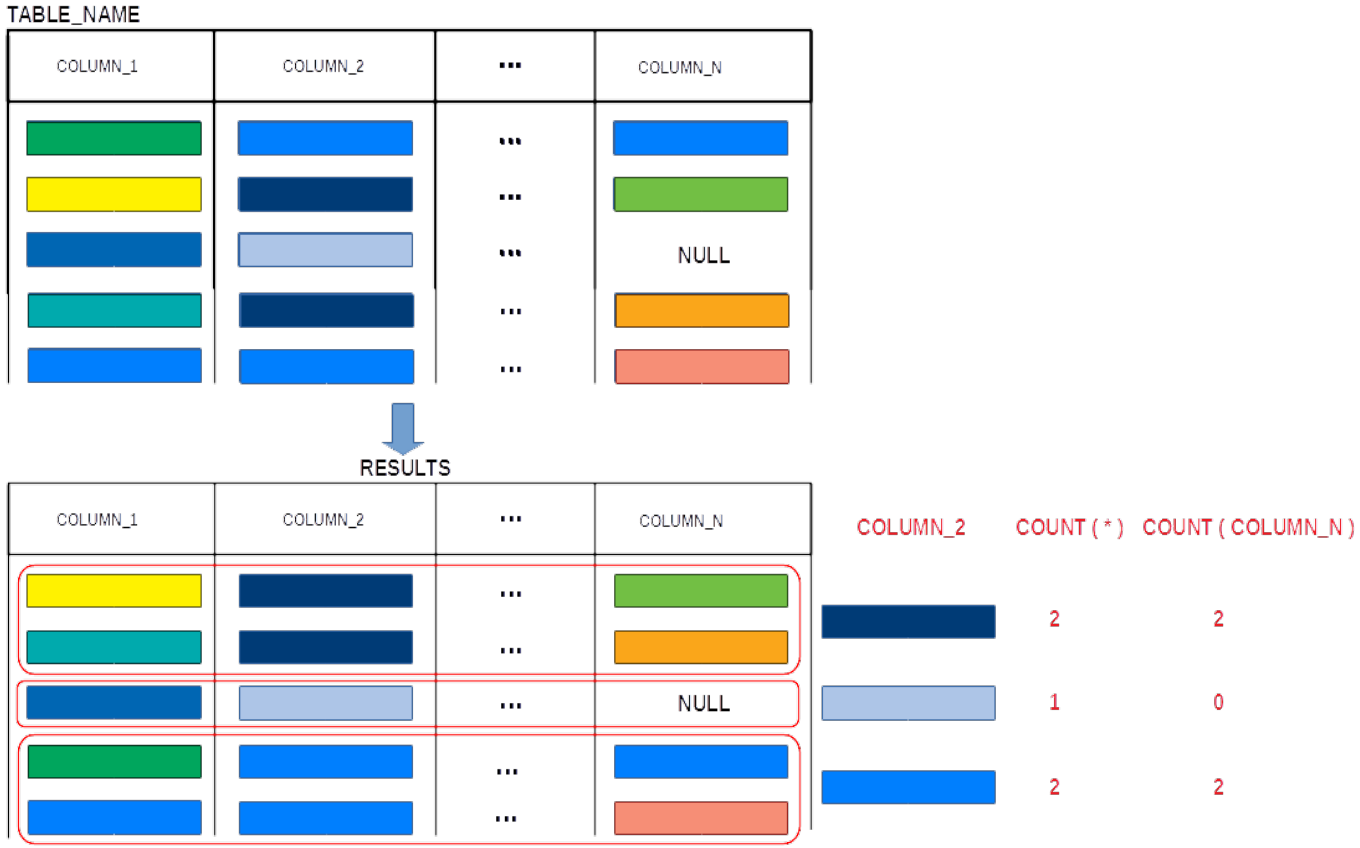


In HTML view press 'p' to see the lecture notes

Grouping

Aggregation function can be applied to an empty group of values

```
SELECT COLUMN_2, COUNT(*), COUNT(COLUMN_N)
FROM TABLE_NAME
GROUP BY COLUMN_2;
```



In HTML view press 'p' to see the lecture notes

Grouping

Find the names of all departments that offer at least one course, do not list the same names of departments

```
SELECT DISTINCT offered_by
FROM COURSE;
```

SELECT statement with DISTINCT clause

```
SELECT offered_by
FROM COURSE
GROUP BY offered_by;
```

SELECT statement with GROUP BY clause

`GROUP BY` sorts a relational table over the attributes included in `GROUP BY` list and creates the groups of rows with identical values of the attributes in `GROUP BY` list

In the example above a table `COURSE` is sorted over the values of an attribute `offered_by`

The groups of rows with the identical values of an attribute `offered_by` are created

Finally, the values of an attribute `offered_by` are selected, one value from each group

Grouping

Grouping allows for application of group functions to each group created by `GROUP BY` clause

Find the names of all departments together with the total number of all courses offered by each department

SELECT statement with GROUP BY clause and aggregation function COUNT

```
SELECT offered_by, COUNT(*)
FROM COURSE
GROUP BY offered_by;
```

- Before sorting

Temporary results after selection

cnum	title	credits	offered_by
CSCI111	C++	6	Computer Science
PHYS312	Relativity	6	Physics
MATH111	Calculus	12	Mathematics
CSCI235	Databases	6	Computer Science
MECH111	Mechanics	12	Physics

Grouping

Grouping allows for application of group functions to each group created by `GROUP BY` clause

Find the names of all departments together with the total number of all courses offered by each department

SELECT statement with GROUP BY clause and aggregation function COUNT

```
SELECT offered_by, COUNT(*)
FROM COURSE
GROUP BY offered_by;
```

- After grouping by `offered_by`

Temporary results after grouping

```
+---------+------------+---------+------------------+
| cnum    | title      | credits | offered_by       |
+---------+------------+---------+------------------+
| CSCI111 | C++        |       6 | Computer Science |
| CSCI235 | Databases  |       6 | Computer Science |
| MATH111 | Calculus   |      12 | Mathematics      |
| MECH111 | Mechanics  |      12 | Physics          |
| PHYS312 | Relativity |       6 | Physics          |
+---------+------------+---------+------------------+
```

Grouping

Grouping allows for application of group functions to each group created by `GROUP BY` clause

Find the names of all departments together with the total number of all courses offered by each department

SELECT statement with GROUP BY clause and aggregation function COUNT

```
SELECT offered_by, COUNT(*)
FROM COURSE
GROUP BY offered_by;
```

- After counting rows in each group

Final results after grouoping and application of aggregation function COUNT

```
+-------------------+-----------+
| offered_by        | count(*)  |
+-------------------+-----------+
| Computer Science  |         2 |
| Mathematics       |         1 |
| Physics           |         2 |
+-------------------+-----------+
```

Grouping

For each value of credit points find the total number of courses that have the respective credits

SELECT statement with GROUP BY clause and aggregation function COUNT

```
SELECT credits, COUNT(*)
FROM COURSE
GROUP BY credits;
```

For each department find the total number of credit points offered

SELECT statement with GROUP BY clause and aggregation function SUM

```
SELECT offered_by, SUM(credits)
FROM COURSE
GROUP BY offered_by;
```

Find the largest number of courses offered by any department

SELECT statement with GROUP BY clause and aggregation functions COUNT and MAX

```
SELECT MAX(total)
FROM (SELECT COUNT(*) total
      FROM COURSE
      GROUP BY offered_by) T;
```

In HTML view press 'p' to see the lecture notes

SELECT statement (2)

Outline

Queries with simple conditions
Queries with Boolean expressions
Set algebra queries
Sorting
Queries about lack of values (`NULL`s)
Grouping
Grouping with selections

Grouping with selections

Find the names of all departments that offer more than 1 course

SELECT statement with GROUP BY clause and aggregation function COUNT

```
SELECT offered_by, COUNT(*)
FROM COURSE
GROUP BY offered_by;
```

Final results after grouping and application of aggregation function COUNT

```
+-------------------+----------+
| offered_by        | count(*) |
+-------------------+----------+
| Computer Science  |        2 |
| Mathematics       |        1 |
| Physics           |        2 |
+-------------------+----------+
```

Grouping with selections

Find the names of all departments that offer more than 1 course

SELECT statement with GROUP BY and HAVING clauses and aggregation function COUNT

```
SELECT offered_by, count(*)
FROM COURSE
GROUP BY offered_by
HAVING count(*) > 1;
```

Final results after grouping, application of aggregation function COUNT, and HAVING clause

```
+-------------------+-----------+
| offered_by        | count(*)  |
+-------------------+-----------+
| Computer Science  |         2 |
| Physics           |         2 |
+-------------------+-----------+
```

References

C. Coronel, S. Morris, A. Basta, M. Zgola, Data Management and Security, Chapters 5, 7, Cengage Compose eBook, 2018, eBook: Data Management and Security, 1st Edition

T. Connoly, C. Begg, Database Systems, A Practical Approach to Design, Implementation, and Management, Chapters 6.3.1 - 6.3.4 Data Manipulation, Pearson Education Ltd, 2015

D. Darmawikarta, SQL for MySQL A Beginner's Tutorial, Chapters 2 - 5, Brainy Software Inc. First Edition: June 2014

How to ... ? Cookbook, How to implement queries in SQL, Recipe 5.1 How to implement SELECT statements with simple Boolean expressions ?

How to ... ? Cookbook, How to implement queries in SQL, Recipe 5.2 How to implement SELECT statements with the set algebra operations ?

How to ... ? Cookbook, How to implement queries in SQL, Recipe 5.3 How to implement SELECT statements with GROUP BY and HAVING clauses ?