```
--[Object-]Relational Databases Management Systems (MySQL/MariaDB)
-- © C.Kaldeich
--Part One: Relations & related stuff
--
--Review & exercises
--Theme: 'Stationary shop'
--db2
[..]> CREATE DATABASE db2 CHARSET latin1 COLLATE latin1_swedish_ci;
--
[..]> CREATE TABLE IF NOT EXISTS products (
productID    INT UNSIGNED  NOT NULL AUTO_INCREMENT,
productCode  CHAR(3)       NOT NULL DEFAULT '',
name         VARCHAR(30)   NOT NULL DEFAULT '',
quantity     INT UNSIGNED  NOT NULL DEFAULT 0,
price        DECIMAL(7,2)  NOT NULL DEFAULT 99.99,
PRIMARY KEY  (productID)          -- Index built automatically on
);                                -- primary-key column 'productID'
Note: There is no 'ENGINE' declaration at the end of the statement.
Check in: http://acmeextension.com/difference-between-innodb-and-myisam/
--
[..]> DES[CRIBE] products;  --Equivalent to 'DESC' or 'DESCRIBE'
--
[..]> SHOW CREATE TABLE products;
--
Try also:
[..]> SHOW CREATE TABLE products \G
--
Single insert w/ value: Note, that the table was defined with the
option 'AUTO_INCREMENT' for  'productID'.
--
[..]> INSERT INTO products VALUES (1001, 'PEN', 'Pen Red', 5000, 1.23);
--
Check table products:
[..]> SELECT * FROM products;
or
[..]> SELECT * FROM products \G
--
[..]> INSERT INTO products VALUES
(NULL, 'PEN', 'Pen Blue',  8000, 1.25),
(NULL, 'PEN', 'Pen Black', 2000, 1.25);
--
Check the table..
Note: how the system applied the 'auto_increment'?
--
```

```
Insert value to selected columns:
[..]> INSERT INTO products (productCode, name, quantity, price)
VALUES
('PEC', 'Pencil 2B', 10000, 0.48),
('PEC', 'Pencil 2H', 8000, 0.49);
Check it again: the missing value for the auto_increment column also results in
max_value + 1
--
Missing values get default values:
[..]> INSERT INTO products (productCode, name) VALUES ('PEC', 'Pencil HB');

2nd attribute (productCode) is defined to be NOT NULL; check it:
[..]> INSERT INTO products values (NULL, NULL, NULL, NULL, NULL);
--
Delete last inserted row from the table products
[..]> DELETE FROM products WHERE productID = 1006;
--

Alternativeley use SET to set the values:
[..]> INSERT INTO tableName SET column1=value1, column2=value2, ...
-
The statement below is equivalent to the statement above:
[..]> INSERT INTO tableName (column1, column2, ...) VALUES (value1, value2, ...)
--

Miscellaneous:

[..]> SELECT User, Host FROM mysql.user;
--
[..]> SELECT User, Host, Password, password_expired FROM mysql.user;
--
About 'select' without table:

[..]> SELECT NOW();
--
[..]> SELECT NOW() time;
--
[..]> SELECT NOW() time, sqrt(2) square_root;
--
Warm-up exercises:
Perform all commands above on your sql-terminal.
--
```

```
Comparison operators for numbers (INT, DECIMAL, FLOAT):
'='            (equal to),
'<>' or '!='   (not equal to),
'>'            (greater than),
'<'            (less than),
'>='           (greater than or equal to),
'<='           (less than or equal to).

For example, price > 1.0, quantity <= 500.
--Check:
[..]> SELECT name, price FROM products WHERE price < 1.0;
--
[..]> SELECT name, quantity FROM products WHERE quantity <= 2000;
--
Do not compare FLOATs (real numbers) for equality ('=' or '<>'), as
they are not precise.
On the other hand, DECIMAL are precise.
--
For strings, in addition to full matching using operators like '='
and '<>', we can perform pattern matching using operator LIKE (or NOT LIKE) with
wildcard characters.

The WILDCARDS:

'_' matches any single character;
'%' matches any number of characters (including zero).

For example:
'abc%'    matches strings beginning with 'abc';
'%xyz'    matches strings ending with 'xyz';
'%aaa%'   matches strings containing 'aaa';
'___'     attaches strings containing exactly three characters
           ( 3 x _ );
'a_b%'    matches strings beginning with 'a', followed by any single character,
          followed by 'b', followed by zero or more characters.
--
For strings, you could also use '=', '<>', '>', '<', '>=', '<=' to compare two
strings (e.g., productCode = 'PEC').
--
Important note: The ordering of string depends on the so-called "collation" chosen
(the charset and collation (e.g., "collate latin1_swedish_ci") are defined, when
the database is created, so give attention to this detail in order to get the
symbols and letters related to you language).
--
For example:
[..]> SELECT name, price FROM products WHERE productCode = 'PEN';
--
```

```
Try also:
[..]> SELECT name, price FROM products WHERE productCode > 'PEN';
--
[..]> SELECT name, price FROM products WHERE productCode < 'PEN';
--
"name" begins with 'PENCIL'
SELECT name, price FROM products WHERE name LIKE 'PENCIL%';
--
"name" begins with 'P', followed by any two characters, followed by space,
followed by zero or more characters:
[..]> SELECT name, price FROM products WHERE name LIKE 'P__ %';
--
[..]> SELECT name, price FROM products WHERE name LIKE 'P_____ %';
--
MariaDB/MySQL also support 'REGULAR EXPRESSIONS' ('regexp') matching via the
REGEXP operator.
[..]> SELECT 'Pencil' REGEXP 'Pencil';
--
[..]> SELECT 'Pencil' REGEXP 'pencil';
--
[..]> SELECT BINARY 'Pencil' REGEXP 'pencil';
--
Note that the word being matched must match the whole pattern.
Check it:
[..]> SELECT 'Pencil' REGEXP 'Penci';
--
[..]> SELECT 'Penci' REGEXP 'Pencil';
--
A match can be performed against more than one word with the '|' character (pipe).
For example:
[..]> SELECT 'Pencil' REGEXP 'Book|Pencil';
--
[..]> SELECT 'Pencil' REGEXP 'Book|Pen';
--
[..]> SELECT 'Pen' REGEXP 'Book|Pencil';
Check it.
--
Special Characters:
The above examples introduce the syntax, but are not very useful. It's the special
characters that give regular expressions their power.
--
^ matches the beginning of a string (inside square brackets it can also mean NOT.
[..]> SELECT 'Pencil' REGEXP '^Pe';
--
$ matches the end of a string:
[..]> SELECT 'Pencil' REGEXP 'il$';
--
```

```
A dot '.' matches any single character:
SELECT 'Pencil' REGEXP 'Pen.il';
--
[..]> SELECT 'Pencil' REGEXP 'Pe..il';
--
x* matches zero or more of a character 'x'. In the examples below, it's the 'n'
character:
[..]> SELECT 'Pencil' REGEXP 'Pe*cil';
--
[..]> SELECT 'Pencil' REGEXP 'Pen*cil';
--
SELECT 'Pennnncil' REGEXP 'Pen*cil';
--
x? matches zero or one of a character 'x'. In the examples below, it's the 'n'
character.
[..]> SELECT 'Pencil' REGEXP 'Pen?cil';
+---------------------------+
| 'Pencil' REGEXP 'Pen?cil' |
+---------------------------+
|                         1 |
+---------------------------+

[..]> SELECT 'Pecil' REGEXP 'Pen?cil';
+--------------------------+
| 'Pecil' REGEXP 'Pen?cil' |
+--------------------------+
|                        1 |
+--------------------------+

[..]> SELECT 'Marria' REGEXP 'Mar?ia';
+--------------------------+
| 'Marria' REGEXP 'Mar?ia' |
+--------------------------+
|                        0 |
+--------------------------+

[..]> SELECT 'Pennncil' REGEXP 'Pen?cil';
+----------------------------+
| 'Pennncil' REGEXP 'Pen?cil' |
+----------------------------+
|                          0 |
+----------------------------+
--
REGEXP is a very large subject, check the literature to get more information about
'regular expressions'.
--
```

ARITHMETIC OPERATORS

To perform arithmetic operations on numeric fields using arithmetic operators, check the table below:

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
DIV	Integer Division
%	Modulus (Remainder)
Logical Operators	AND, OR, NOT, XOR (short: 'logop' or 'LOGOP')

--

It is possible to combine multiple conditions with boolean operators AND, OR, XOR. You may also invert a condition using operator NOT.

For example:

```
[..]> SELECT * FROM products WHERE quantity >= 5000 AND name LIKE 'Pen %';
+-----------+-------------+----------+----------+-------+
| productID | productCode | name     | quantity | price |
+-----------+-------------+----------+----------+-------+
|      1001 | PEN         | Pen Red  |     5000 |  1.23 |
|      1002 | PEN         | Pen Blue |     8000 |  1.25 |
+-----------+-------------+----------+----------+-------+

[..]> SELECT * FROM products WHERE quantity >= 5000 AND price < 1.24
AND name LIKE 'Pen %';
+-----------+-------------+---------+----------+-------+
| productID | productCode | name    | quantity | price |
+-----------+-------------+---------+----------+-------+
|      1001 | PEN         | Pen Red |     5000 |  1.23 |
+-----------+-------------+---------+----------+-------+

[..]> SELECT * FROM products WHERE quantity >= 5000 AND NOT price < 1.24;
+-----------+-------------+----------+----------+-------+
| productID | productCode | name     | quantity | price |
+-----------+-------------+----------+----------+-------+
|      1002 | PEN         | Pen Blue |     8000 |  1.25 |
+-----------+-------------+----------+----------+-------+
```

```
[..]> SELECT * FROM products WHERE (quantity >= 5000 AND name LIKE 'Pen%');
+-----------+-------------+-----------+----------+-------+
| productID | productCode | name      | quantity | price |
+-----------+-------------+-----------+----------+-------+
|      1001 | PEN         | Pen Red   |     5000 |  1.23 |
|      1002 | PEN         | Pen Blue  |     8000 |  1.25 |
|      1004 | PEC         | Pencil 2B |    10000 |  0.48 |
|      1005 | PEC         | Pencil 2H |     8000 |  0.49 |
+-----------+-------------+-----------+----------+-------+

[..]> SELECT * FROM products WHERE (quantity >= 5000 AND name LIKE 'Pen %');
+-----------+-------------+----------+----------+-------+
| productID | productCode | name     | quantity | price |
+-----------+-------------+----------+----------+-------+
|      1001 | PEN         | Pen Red  |     5000 |  1.23 |
|      1002 | PEN         | Pen Blue |     8000 |  1.25 |
+-----------+-------------+----------+----------+-------+

NOT:
[..]> SELECT * FROM products WHERE NOT (quantity >= 5000 AND name LIKE 'Pen %');
+-----------+-------------+-----------+----------+-------+
| productID | productCode | name      | quantity | price |
+-----------+-------------+-----------+----------+-------+
|      1003 | PEN         | Pen Black |     2000 |  1.25 |
|      1004 | PEC         | Pencil 2B |    10000 |  0.48 |
|      1005 | PEC         | Pencil 2H |     8000 |  0.49 |
+-----------+-------------+-----------+----------+-------+

Exercise 1:
[..]> SELECT * FROM products WHERE NOT (quantity >= 5000) AND
NOT (name LIKE '.......');
+-----------+-------------+-----------+----------+-------+
| productID | productCode | name      | quantity | price |
+-----------+-------------+-----------+----------+-------+
|      1003 | PEN         | Pen Black |     2000 |  1.25 |
+-----------+-------------+-----------+----------+-------+

XOR:
[..]> SELECT * FROM products WHERE quantity >= 5000 XOR name LIKE 'Pen %';
+-----------+-------------+-----------+----------+-------+
| productID | productCode | name      | quantity | price |
+-----------+-------------+-----------+----------+-------+
|      1003 | PEN         | Pen Black |     2000 |  1.25 |
|      1004 | PEC         | Pencil 2B |    10000 |  0.48 |
|      1005 | PEC         | Pencil 2H |     8000 |  0.49 |
+-----------+-------------+-----------+----------+-------+
Why? Explain the semantic of the query above.
```

IN, NOT IN

You can select from members of a set with IN (or NOT IN) operator. This is easier and clearer than the equivalent AND-OR expression.

```
[..]> SELECT * FROM products WHERE name IN ('Pen Red', 'Pen Black');
+-----------+-------------+-----------+----------+-------+
| productID | productCode | name      | quantity | price |
+-----------+-------------+-----------+----------+-------+
|      1001 | PEN         | Pen Red   |     5000 |  1.23 |
|      1003 | PEN         | Pen Black |     2000 |  1.25 |
+-----------+-------------+-----------+----------+-------+
```

Exercise 2:
Try by yourself: include the NOT.
--

BETWEEN, NOT BETWEEN

To check if the value is within a range, use the BETWEEN ... AND ... operator. Again, this is easier and clearer than the equivalent AND-OR expression.

```
[..]> SELECT * FROM products WHERE (price BETWEEN 1.0 AND 2.0);
+-----------+-------------+-----------+----------+-------+
| productID | productCode | name      | quantity | price |
+-----------+-------------+-----------+----------+-------+
|      1001 | PEN         | Pen Red   |     5000 |  1.23 |
|      1002 | PEN         | Pen Blue  |     8000 |  1.25 |
|      1003 | PEN         | Pen Black |     2000 |  1.25 |
+-----------+-------------+-----------+----------+-------+

[..]> SELECT * FROM products WHERE NOT (price BETWEEN 1.0 AND 2.0);
+-----------+-------------+-----------+----------+-------+
| productID | productCode | name      | quantity | price |
+-----------+-------------+-----------+----------+-------+
|      1004 | PEC         | Pencil 2B |    10000 |  0.48 |
|      1005 | PEC         | Pencil 2H |     8000 |  0.49 |
+-----------+-------------+-----------+----------+-------+

[..]> SELECT * FROM products WHERE (price BETWEEN 1.0 AND 2.0) AND (quantity
BETWEEN 1000 AND 2000);
+-----------+-------------+-----------+----------+-------+
| productID | productCode | name      | quantity | price |
+-----------+-------------+-----------+----------+-------+
|      1003 | PEN         | Pen Black |     2000 |  1.25 |
+-----------+-------------+-----------+----------+-------+
```

Exercise 3:
Try it by yourself: Introduce the logical NOT and check the results.

Note: Try additional examples using the 'NOT'.

--
IS NULL, IS NOT NULL

NULL is a special value, which represent "no value", "missing value" or "unknown value". It is possible to check, if a column contains NULL by IS NULL or IS NOT NULL.
For example:
```
[..]> SELECT * FROM products WHERE productCode IS NULL;
Empty set (0.00 sec)
```
--
Note: Using comparison operators (such as = or <>) to check for NULL is a mistake - a very common mistake: NULL cannot be compared.
--

ORDER BY Clause

You can order the rows selected using ORDER BY clause, with the following syntax:
```
[..]> SELECT ... FROM tableName
WHERE criteria
ORDER BY columnA ASC|DESC, columnB ASC|DESC, ... ;
```

Note: If several rows have the same value in columnA, it will be ordered according to columnB, and so on. For strings, the ordering could be case-sensitive or case-insensitive, depending on the so-called character collating sequence used.
--
For examples:
Order the results by price in descending order:
```
[..]> SELECT * FROM products WHERE name LIKE 'Pen %' ORDER BY price DESC;
+-----------+-------------+-----------+----------+-------+
| productID | productCode | name      | quantity | price |
+-----------+-------------+-----------+----------+-------+
|      1002 | PEN         | Pen Blue  |     8000 |  1.25 |
|      1003 | PEN         | Pen Black |     2000 |  1.25 |
|      1001 | PEN         | Pen Red   |     5000 |  1.23 |
+-----------+-------------+-----------+----------+-------+
```

Order by price in descending order, followed by quantity in ascending (default) order:

```
[..]> SELECT * FROM products WHERE name LIKE 'Pen %'ORDER BY price DESC, quantity;
+-----------+-------------+-----------+----------+-------+
| productID | productCode | name      | quantity | price |
+-----------+-------------+-----------+----------+-------+
|      1003 | PEN         | Pen Black |     2000 |  1.25 |
|      1002 | PEN         | Pen Blue  |     8000 |  1.25 |
|      1001 | PEN         | Pen Red   |     5000 |  1.23 |
+-----------+-------------+-----------+----------+-------+
```

Exercises 4/5.
Check the following statements:

```
[..]> SELECT productID, productCode, quantity FROM products ORDER
BY ... ... , ...;
+-----------+-------------+----------+
| productID | productCode | quantity |
+-----------+-------------+----------+
|      1003 | PEN         |     2000 |
|      1001 | PEN         |     5000 |
|      1002 | PEN         |     8000 |
|      1005 | PEC         |     8000 |
|      1004 | PEC         |    10000 |
+-----------+-------------+----------+
--
[..]> SELECT productID,name,quantity,price,(.... .. ....)
FROM products ORDER BY ... ;
+-----------+-----------+----------+-------+--------------------+
| productID | name      | quantity | price | (                ) |
+-----------+-----------+----------+-------+--------------------+
|      1003 | Pen Black |     2000 |  1.25 |            2500.00 |
|      1005 | Pencil 2H |     8000 |  0.49 |            3920.00 |
|      1004 | Pencil 2B |    10000 |  0.48 |            4800.00 |
|      1001 | Pen Red   |     5000 |  1.23 |            6150.00 |
|      1002 | Pen Blue  |     8000 |  1.25 |           10000.00 |
+-----------+-----------+----------+-------+--------------------+
--
```

LIMIT Clause

A SELECT query on a large database may produce many rows.
It is possible to use the LIMIT clause to limit the number of rows displayed.

Examples.
Display the first two rows of a select statement:

```
[..]> SELECT * FROM products ORDER BY price LIMIT 2;
+-----------+-------------+-----------+----------+-------+
| productID | productCode | name      | quantity | price |
+-----------+-------------+-----------+----------+-------+
|      1004 | PEC         | Pencil 2B |    10000 |  0.48 |
|      1005 | PEC         | Pencil 2H |     8000 |  0.49 |
+-----------+-------------+-----------+----------+-------+
--
To continue to the following records, specify the number of rows to
be skipped, followed by the number of rows to be displayed in the
LIMIT clause.

Example:
Skip the first two rows and display the next (one) row:
[..]> SELECT * FROM products ORDER BY price LIMIT 2, 1;
+-----------+-------------+---------+----------+-------+
| productID | productCode | name    | quantity | price |
+-----------+-------------+---------+----------+-------+
|      1001 | PEN         | Pen Red |     5000 |  1.23 |
+-----------+-------------+---------+----------+-------+

How?
Step 1:
[..]> SELECT * FROM products ORDER BY price;
+-----------+-------------+-----------+----------+-------+
| productID | productCode | name      | quantity | price |
+-----------+-------------+-----------+----------+-------+
|      1004 | PEC         | Pencil 2B |    10000 |  0.48 |Step 2
|      1005 | PEC         | Pencil 2H |     8000 |  0.49 |Step 2
|      1001 | PEN         | Pen Red   |     5000 |  1.23 |Step 3
|      1002 | PEN         | Pen Blue  |     8000 |  1.25 |
|      1003 | PEN         | Pen Black |     2000 |  1.25 |
+-----------+-------------+-----------+----------+-------+

Step 2:
[..]> SELECT * FROM products ORDER BY price LIMIT 2;
+-----------+-------------+-----------+----------+-------+
| productID | productCode | name      | quantity | price |
+-----------+-------------+-----------+----------+-------+
|      1004 | PEC         | Pencil 2B |    10000 |  0.48 |
|      1005 | PEC         | Pencil 2H |     8000 |  0.49 |
+-----------+-------------+-----------+----------+-------+
```

```
Step 3:
[..]> SELECT * FROM products ORDER BY price LIMIT 2,1;
+-----------+-------------+---------+----------+-------+
| productID | productCode | name    | quantity | price |
+-----------+-------------+---------+----------+-------+
|      1001 | PEN         | Pen Red |     5000 |  1.23 |
+-----------+-------------+---------+----------+-------+

Exercises 6/7/8/9:
[..]> SELECT * FROM products ORDER BY price LIMIT ... , ... ;
+-----------+-------------+----------+----------+-------+
| productID | productCode | name     | quantity | price |
+-----------+-------------+----------+----------+-------+
|      1001 | PEN         | Pen Red  |     5000 |  1.23 |
|      1002 | PEN         | Pen Blue |     8000 |  1.25 |
+-----------+-------------+----------+----------+-------+

[..]> SELECT * FROM products .......  ....  ......... ;
+-----------+-------------+-----------+----------+-------+
| productID | productCode | name      | quantity | price |
+-----------+-------------+-----------+----------+-------+
|      1003 | PEN         | Pen Black |     2000 |  1.25 |
|      1001 | PEN         | Pen Red   |     5000 |  1.23 |
|      1002 | PEN         | Pen Blue  |     8000 |  1.25 |
|      1005 | PEC         | Pencil 2H |     8000 |  0.49 |
|      1004 | PEC         | Pencil 2B |    10000 |  0.48 |
+-----------+-------------+-----------+----------+-------+

[..]> SELECT * FROM products .......  ....  ......... LIMIT ... ;
+-----------+-------------+-----------+----------+-------+
| productID | productCode | name      | quantity | price |
+-----------+-------------+-----------+----------+-------+
|      1003 | PEN         | Pen Black |     2000 |  1.25 |
|      1001 | PEN         | Pen Red   |     5000 |  1.23 |
+-----------+-------------+-----------+----------+-------+

[..]> SELECT * FROM products .......  ....  ......... LIMIT ... , ... ;
+-----------+-------------+-----------+----------+-------+
| productID | productCode | name      | quantity | price |
+-----------+-------------+-----------+----------+-------+
|      1005 | PEC         | Pencil 2H |     8000 |  0.49 |
+-----------+-------------+-----------+----------+-------+
--
```

AS - Alias

The keyword 'AS' to define an alias, which is an identifier (such as column name, table name). The alias will be used in displaying the name. It can also be used as reference.

For example:

```
[..]> SELECT productID AS ID, productCode AS Code,name AS Description,
price AS 'Unit Price'
FROM products
ORDER BY ID;
+------+------+-------------+------------+
| ID   | Code | Description | Unit Price |
+------+------+-------------+------------+
| 1001 | PEN  | Pen Red     |       1.23 |
| 1002 | PEN  | Pen Blue    |       1.25 |
| 1003 | PEN  | Pen Black   |       1.25 |
| 1004 | PEC  | Pencil 2B   |       0.48 |
| 1005 | PEC  | Pencil 2H   |       0.49 |
+------+------+-------------+------------+
```

Note: The identifier (also called "alias") "Unit Price" contains a blank and must be (back-) quoted.

Exercise 10.

```
[..]> SELECT productID,name,quantity,price, (..... .. .....) AS .....
FROM products ORDER BY ... ;
+-----------+-----------+----------+-------+----------+
| productID | name      | quantity | price |          |
+-----------+-----------+----------+-------+----------+
|      1003 | Pen Black |     2000 |  1.25 |  2500.00 |
|      1005 | Pencil 2H |     8000 |  0.49 |  3920.00 |
|      1004 | Pencil 2B |    10000 |  0.48 |  4800.00 |
|      1001 | Pen Red   |     5000 |  1.23 |  6150.00 |
|      1002 | Pen Blue  |     8000 |  1.25 | 10000.00 |
+-----------+-----------+----------+-------+----------+
--
```

CONCAT()

To concatenate a few columns as one (e.g., joining the last name and first name) by using function CONCAT().

For example:

```
[..]> SELECT CONCAT(productCode,' - ',name) AS 'Product Description', price Price
FROM products;
```

```
+---------------------+-------+
| Product Description | Price |
+---------------------+-------+
| PEN - Pen Red       |  1.23 |
| PEN - Pen Blue      |  1.25 |
| PEN - Pen Black     |  1.25 |
| PEC - Pencil 2B     |  0.48 |
| PEC - Pencil 2H     |  0.49 |
+---------------------+-------+
```

Check below: there are 2 ways to sort the output by 'Product Description' (columns 1+2) or by 'Price'.

```
[..]> SELECT CONCAT(productCode,' - ',name) AS 'Product Description', price Price
FROM products ORDER BY price;
+---------------------+-------+
| Product Description | Price |
+---------------------+-------+
| PEC - Pencil 2B     |  0.48 |
| PEC - Pencil 2H     |  0.49 |
| PEN - Pen Red       |  1.23 |
| PEN - Pen Blue      |  1.25 |
| PEN - Pen Black     |  1.25 |
+---------------------+-------+
or,
[..]> SELECT CONCAT(productCode,' - ',name) AS 'Product Description',
price Price FROM products
ORDER BY 'Product Description';
+---------------------+-------+
| Product Description | Price |
+---------------------+-------+
| PEN - Pen Red       |  1.23 |
| PEN - Pen Blue      |  1.25 |
| PEN - Pen Black     |  1.25 |
| PEC - Pencil 2B     |  0.48 |
| PEC - Pencil 2H     |  0.49 |
+---------------------+-------+
```

Note: It is possible to use an ordinal to refers the attribute or column. That means, the ordinal refers to the position of the attribute after the SELECT.

```
[..]> SELECT CONCAT(productCode,' - ',name) AS 'Product Description', price Price
FROM products ORDER BY 2;
```

```
+---------------------+-------+
| Product Description | Price |
+---------------------+-------+
| PEC - Pencil 2B     |  0.48 |
| PEC - Pencil 2H     |  0.49 |
| PEN - Pen Red       |  1.23 |
| PEN - Pen Blue      |  1.25 |
| PEN - Pen Black     |  1.25 |
+---------------------+-------+
or,
[..]> SELECT CONCAT(productCode,' - ',name) AS 'Product Description', price Price
FROM products ORDER BY 2 desc;
+---------------------+-------+
| Product Description | Price |
+---------------------+-------+
| PEN - Pen Blue      |  1.25 |
| PEN - Pen Black     |  1.25 |
| PEN - Pen Red       |  1.23 |
| PEC - Pencil 2H     |  0.49 |
| PEC - Pencil 2B     |  0.48 |
+---------------------+-------+
or,
[..]> SELECT CONCAT(productCode,' - ',name) AS 'Product Description', price Price
FROM products ORDER BY 1;
+---------------------+-------+
| Product Description | Price |
+---------------------+-------+
| PEC - Pencil 2B     |  0.48 |
| PEC - Pencil 2H     |  0.49 |
| PEN - Pen Black     |  1.25 |
| PEN - Pen Blue      |  1.25 |
| PEN - Pen Red       |  1.23 |
+---------------------+-------+
```

Exercise 11.
Create one-column list of products with the name and ten-pack price with 10% discount.

```
[..]> SELECT CONCAT(name,'     ',((..... * ....) .. ...),'$') AS 'Product 10-pack
10% discount'
FROM products;
```

```
+------------------------------+
| Product 10-pack 10% discount |
+------------------------------+
| Pen Red     11.0700$         |
| Pen Blue     11.2500$        |
| Pen Black     11.2500$       |
| Pencil 2B     4.3200$        |
| Pencil 2H     4.4100$        |
+------------------------------+
-
```

DISTINCT

A column may have duplicate values, so use keyword DISTINCT to select only distinct values. Apply also apply DISTINCT to several columns to select distinct combinations of these columns.
For example:

```
[..]> SELECT price FROM products;
+-------+
| price |
+-------+
|  1.23 |
|  1.25 |
|  1.25 |
|  0.48 |
|  0.49 |
+-------+

[..]> SELECT DISTINCT price FROM products;
+-------+
| price |
+-------+
|  1.23 |
|  1.25 |
|  0.48 |
|  0.49 |
+-------+
```

```
Remove duplicates:
[..]> SELECT DISTINCT price AS 'Distinct Prices' FROM products;
+-----------------+
| Distinct Prices |
+-----------------+
|            1.23 |
|            1.25 |
|            0.48 |
|            0.49 |
+-----------------+

But...
[..]> SELECT DISTINCT CONCAT(price,' - ',name) AS 'Price/Product'
FROM products;
+------------------+
| Price/Product    |
+------------------+
| 1.23 - Pen Red   |
| 1.25 - Pen Blue  |
| 1.25 - Pen Black |
| 0.48 - Pencil 2B |
| 0.49 - Pencil 2H |
+------------------+
and
[..]> SELECT  price, name FROM products;
+-------+-----------+
| price | name      |
+-------+-----------+
|  1.23 | Pen Red   |
|  1.25 | Pen Blue  |
|  1.25 | Pen Black |
|  0.48 | Pencil 2B |
|  0.49 | Pencil 2H |
+-------+-----------+

Exercise 12: Why? Explain.
--
```

GROUP BY Clause

The GROUP BY clause allows you to collapse multiple records with a common value into groups.
For example:

```
[..]> SELECT * FROM products ORDER BY productCode, productID;
+-----------+-------------+-----------+----------+-------+
| productID | productCode | name      | quantity | price |
+-----------+-------------+-----------+----------+-------+
|      1004 | PEC         | Pencil 2B |    10000 |  0.48 |
|      1005 | PEC         | Pencil 2H |     8000 |  0.49 |
|      1001 | PEN         | Pen Red   |     5000 |  1.23 |
|      1002 | PEN         | Pen Blue  |     8000 |  1.25 |
|      1003 | PEN         | Pen Black |     2000 |  1.25 |
+-----------+-------------+-----------+----------+-------+

[..]> SELECT * FROM products GROUP BY productCode;
-- Only first record's name in each group is shown.
+-----------+-------------+--------------------+----------+-------+
| productID | productCode | name               | quantity | price |
+-----------+-------------+--------------------+----------+-------+
|      1001 | PEN         | Pen Red            |    5000 |   1.23 |
|      1004 | PEC         | Pencil 2B          |   10000 |   0.48 |
+-----------+-------------+--------------------+----------+-------+

[..]> SELECT * FROM products GROUP BY productCode ORDER BY 1;
-- '1' is an ordinal and refers to the first column.
+-----------+-------------+-----------+----------+-------+
| productID | productCode | name      | quantity | price |
+-----------+-------------+-----------+----------+-------+
|      1001 | PEN         | Pen Red   |     5000 |  1.23 |
|      1004 | PEC         | Pencil 2B |    10000 |  0.48 |
+-----------+-------------+-----------+----------+-------+
```

Note: It is easy to see, that the attributes 'name', 'quantity' and 'price' have no relevance to the groups. So, those attributes should be suppressed from the result table.

-

Exercise 13. Complete the code in order to get the results below:

```
[..]> SELECT productCode Code, SUM(.........) AS "......"
FROM products
GROUP BY ............;
+------+-------+
| Code | Qty   |
+------+-------+
| PEC  | 18000 |
| PEN  | 15000 |
+------+-------+
```

GROUP BY Aggregate Functions. COUNT, MAX, MIN, AVG, SUM, STD, GROUP_CONCAT:

GROUP BY used together with GROUP BY aggregate functions (such as COUNT(), AVG(), SUM()) to produce group summary.
The function COUNT(*) returns the rows selected; and COUNT(columnName) counts only the non-NULL values of the given column.

An easy example:

```
[..]> SELECT COUNT(*) AS 'Nr. of codes' FROM products;
+--------------+
| Nr. of codes |
+--------------+
|            5 |
+--------------+
```

--

Before inserting more tuples (records) into 'prodcuts' the AUTO_INCREMENT should be reseted (remeber the tuple 1006 was removed, so the AUTO_INCREMENT is set on '1007', which will be the 'productID' of the next inserted record).
The syntax is:

```
ALTER TABLE table_name AUTO_INCREMENT = value;
```

The statement is:

```
ALTER TABLE products AUTO_INCREMENT = 1006;
```

--

Now, perform the inserts:

```
INSERT INTO products (productCode,name,quantity,price)
VALUES
('NTB', 'Notebook A4L90p', 9000, 3.30),
('NTB', 'Notebook B5L60p', 8000, 2.80),
('NTB', 'Notebook A5L50p', 70000, 2.20),
('NTB', 'Notebook B6L40p', 70000, 2.00);
```

Check it:

```
[..]> SELECT * FROM products;
+-----------+-------------+-----------------+----------+-------+
| productID | productCode | name            | quantity | price |
+-----------+-------------+-----------------+----------+-------+
|      1001 | PEN         | Pen Red         |     5000 |  1.23 |
|      1002 | PEN         | Pen Blue        |     8000 |  1.25 |
|      1003 | PEN         | Pen Black       |     2000 |  1.25 |
|      1004 | PEC         | Pencil 2B       |    10000 |  0.48 |
|      1005 | PEC         | Pencil 2H       |     8000 |  0.49 |
|      1006 | NTB         | Notebook A4L90p |     9000 |  3.30 |
|      1007 | NTB         | Notebook B5L60p |     8000 |  2.80 |
|      1008 | NTB         | Notebook A5L50p |    70000 |  2.20 |
|      1009 | NTB         | Notebook B6L40p |    70000 |  2.00 |
+-----------+-------------+-----------------+----------+-------+
9 rows..
```

```
--
Check number of products by productCode:
[..]> SELECT productCode AS Code,COUNT(*) AS 'Items by Code'
FROM products
WHERE productCode='NTB';
+------+---------------+
| Code | Items by Code |
+------+---------------+
| NTB  |             4 |
+------+---------------+
--
Setting and using runtime variables:
SET @productCode='NTB';
[..]> SELECT productCode AS Code, COUNT(*) AS 'Items by Code'
FROM products WHERE productCode=@productCode;
+------+---------------+
| Code | Items by Code |
+------+---------------+
| NTB  |             4 |
+------+---------------+
--
More inserts:
[..]> INSERT INTO products (productCode, name, quantity, price) VALUES
('PAP', 'Paper A4W80/500',1100,5.00),
('PAP', 'Paper A4E80/500',1200,4.90),
('RUB', 'Rubber white Soft', 1200, 1.99),
('RUB', 'Rubber color Medium', 1500, 0.99);
--
Check the table, run:
[..]> SELECT * FROM products;
--
[..]> SELECT productCode AS Item,COUNT(*) AS 'Items by Code'
FROM products
GROUP BY productCode;
+------+---------------+
| Item | Items by Code |
+------+---------------+
| NTB  |             4 |
| PAP  |             2 |
| PEC  |             2 |
| PEN  |             3 |
| RUB  |             2 |
+------+---------------+
Or
```

```
[..]> SELECT productCode AS Item,COUNT(*) AS 'Items by Code'
FROM products
GROUP BY productCode ORDER BY 2;
+------+---------------+
| Item | Items by Code |
+------+---------------+
| PEC  |             2 |
| PAP  |             2 |
| RUB  |             2 |
| PEN  |             3 |
| NTB  |             4 |
+------+---------------+
--
GROUP BY aggregate functions such as AVG(), MAX(), MIN() and SUM().

For example: (Without GROUP BY - All rows)
[..]> SELECT MAX(price), MIN(price), AVG(price), STD(price), SUM(quantity)
FROM products;
+------------+------------+------------+------------+---------------+
| MAX(price) | MIN(price) | AVG(price) | STD(price) | SUM(quantity) |
+------------+------------+------------+------------+---------------+
|       5.00 |       0.48 |   2.136923 |   1.422211 |        195000 |
+------------+------------+------------+------------+---------------+
The values "AVG(price)" and "STD(price)" should be formated (xxx.xx).
The format of outputs, will be introduced below.

[..]> SELECT productCode AS Code, MAX(price) AS 'Highest Price',
MIN(price) AS 'Lowest Price'
FROM products
GROUP BY productCode;
+------+---------------+--------------+
| Code | Highest Price | Lowest Price |
+------+---------------+--------------+
| DIG  |        449.99 |       449.99 |
| NTB  |          3.63 |         2.20 |
| PAP  |          5.50 |         5.39 |
| PEC  |          0.54 |         0.53 |
| PEN  |          1.38 |         1.30 |
| RUB  |          2.19 |         1.09 |
+------+---------------+--------------+
```

```
Exercise 14: Complete the code, in order to get the result table below.
[..]> SELECT productCode Code, .......(..........) AS "Qty"
FROM products
GROUP BY .....
ORDER BY ..;
+------+--------+
| Code | Qty    |
+------+--------+
| DIG  |      4 |
| PAP  |   2100 |
| RUB  |   2700 |
| PEN  |  14950 |
| PEC  |  18000 |
| NTB  | 154000 |
+------+--------+

--
CAST(... AS ...) function

The CAST function is used for converting a value from one datatype to another
specific datatype, or to format floating-point numbers.

Examples:
Set the variables:
set @a=5, @b=3;

Then, work it out:
select @a+@b;
+-------+
| @a+@b |
+-------+
|     8 |
+-------+

select @a+@b "a+b";
+-------+
| a+b |
+-----+
|   8 |
+-----+

select @a a, @b b, @a+@b "a+b";
+------+------+------+
| a    | b    | a+b  |
+------+------+------+
|    5 |    3 |    8 |
+------+------+------+
```

```
select @a a, @b b, @a*@b "a*b";
+------+------+------+
| a    | b    | a*b  |
+------+------+------+
|    5 |    3 |   15 |
+------+------+------+

select @a a, @b b, sqrt(@a)*sqrt(@b) "sqrt(a)*sqrt(b)";
+------+------+-------------------+
| a    | b    | sqrt(a)*sqrt(b)   |
+------+------+-------------------+
|    5 |    3 | 3.872983346207417 |
+------+------+-------------------+

Finally:
[..]> SELECT @a a, @b b, CAST(sqrt(@a)*sqrt(@b) AS DECIMAL(7,2))
"sqrt(a)*sqrt(b)";
+------+------+-----------------+
| a    | b    | sqrt(a)*sqrt(b) |
+------+------+-----------------+
|    5 |    3 |            3.87 |
+------+------+-----------------+
--
Now, to the table 'products':
[..]> SELECT productCode,MAX(price),MIN(price),AVG(price),STD(price),
SUM(quantity)
FROM products
GROUP BY 1
ORDER BY 6 DESC;
+-------------+------------+------------+------------+------------+---------------+
| productCode | MAX(price) | MIN(price) | AVG(price) | STD(price) | SUM(quantity) |
+-------------+------------+------------+------------+------------+---------------+
| NTB         |       3.30 |       2.00 |   2.575000 |   0.511737 |        157000 |
| PEC         |       0.49 |       0.48 |   0.485000 |   0.005000 |         18000 |
| PEN         |       1.25 |       1.23 |   1.243333 |   0.009428 |         15000 |
| RUB         |       1.99 |       0.99 |   1.490000 |   0.500000 |          2700 |
| PAP         |       5.00 |       4.90 |   4.950000 |   0.050000 |          2300 |
+-------------+------------+------------+------------+------------+---------------+
--
```

Exercise 15. Use CAST(..) to format the floating-point numbers above and to get the results as below (pay attention to the order):

```
+-------------+------------+------------+---------+----------+---------------+
| productCode | MAX(price) | MIN(price) | Average | Std. Dev | SUM(quantity) |
+-------------+------------+------------+---------+----------+---------------+
| NTB         |       3.30 |       2.00 |    2.58 |     0.51 |        157000 |
| PEC         |       0.49 |       0.48 |    0.49 |     0.01 |         18000 |
| PEN         |       1.25 |       1.23 |    1.24 |     0.01 |         15000 |
| RUB         |       1.99 |       0.99 |    1.49 |     0.50 |          2700 |
| PAP         |       5.00 |       4.80 |    4.90 |     0.10 |          2300 |
+-------------+------------+------------+---------+----------+---------------+
--

HAVING clause

HAVING is similar to WHERE, but it can operate on the GROUP BY
aggregate functions; whereas WHERE operates only on columns
(attributes).

[..]> SELECT productCode AS 'Product Code', COUNT(*) AS 'Count',
CAST(AVG(price) AS DECIMAL(7,2)) AS 'Average'
FROM products
GROUP BY productCode
HAVING Count >=3;
+--------------+-------+---------+
| Product Code | Count | Average |
+--------------+-------+---------+
| NTB          |     4 |    2.58 |
| PEN          |     3 |    1.24 |
+--------------+-------+---------+
Note: Cannot use WHERE count >= 3.
--
Exercise 16.
Write the mssing code to get the output below:
[..]> SELECT productCode AS 'Product Code', COUNT(*) AS 'Count',
CAST(AVG(price) AS DECIMAL(7,2)) AS 'Average'
FROM products  GROUP BY productCode
HAVING ....... >= .. ;
+--------------+-------+---------+
| Product Code | Count | Average |
+--------------+-------+---------+
| NTB          |     4 |    2.58 |
| PAP          |     2 |    4.95 |
| PEC          |     2 |    0.49 |
| PEN          |     3 |    1.24 |
| RUB          |     2 |    1.49 |
+--------------+-------+---------+
```

Exercise 17.
Idem above, but set the right value for the HAVING clause.

```
[..]> SELECT productCode AS 'Product Code', COUNT(*) AS 'Count',
CAST(AVG(price) AS DECIMAL(7,2)) AS 'Average'
FROM products
GROUP BY productCode
HAVING ....... >= ..;
+--------------+-------+---------+
| Product Code | Count | Average |
+--------------+-------+---------+
| NTB          |     4 |    2.58 |
+--------------+-------+---------+
```

Exercise 18.
Use another criteria than 'count' to get the output below:

```
[..]> SELECT productCode AS 'Product Code', COUNT(*) AS 'Count',
CAST(AVG(price) AS DECIMAL(7,2)) AS 'Average'
FROM products
GROUP BY productCode
HAVING ......... >= ..;
+--------------+-------+---------+
| Product Code | Count | Average |
+--------------+-------+---------+
| DIG          |     2 |  304.50 |
| PAP          |     2 |    5.15 |
+--------------+-------+---------+
--
```

WITH ROLLUP

The WITH ROLLUP clause shows the summary of group values. For example:

```
[..]> SELECT productCode, MAX(price), MIN(price),
CAST(AVG(price) AS DECIMAL(7,2)) AS 'Average',
SUM(quantity)
FROM products
GROUP BY 1
WITH ROLLUP;
+-------------+------------+------------+---------+---------------+
| productCode | MAX(price) | MIN(price) | Average | SUM(quantity) |
+-------------+------------+------------+---------+---------------+
| NTB         |       3.30 |       2.00 |    2.58 |        157000 |
| PAP         |       5.00 |       4.90 |    4.95 |          2300 |
| PEC         |       0.49 |       0.48 |    0.49 |         18000 |
| PEN         |       1.25 |       1.23 |    1.24 |         15000 |
| RUB         |       1.99 |       0.99 |    1.49 |          2700 |
| NULL        |       5.00 |       0.48 |    2.14 |        195000 |
+-------------+------------+------------+---------+---------------+
```

The aggregate functions were applied to all groups.
--
Is the line starting with 'NULL' inappropriate?
Then, use COALESCE.
--

COALESCE(...., ...)

The SQL COALESCE (and ISNULL) function(s) is (are) used to handle NULL values. During the expression evaluation process the NULL values are replaced with the user-defined value.
COALESCE evaluates the arguments in order and always returns first non-null value from the defined argument list. Note: In SQL NULL is a STATE, not a value.

Example:

```
[..]> SELECT productCode Code, MAX(price), MIN(price),
CAST(AVG(price) AS DECIMAL(7,2)) AS 'Average',
SUM(quantity)
FROM products
GROUP BY productCode
WITH ROLLUP;
+------+------------+------------+---------+---------------+
| Code | MAX(price) | MIN(price) | Average | SUM(quantity) |
+------+------------+------------+---------+---------------+
| NTB  |       3.30 |       2.00 |    2.58 |        157000 |
| PAP  |       5.00 |       4.90 |    4.95 |          2300 |
| PEC  |       0.49 |       0.48 |    0.49 |         18000 |
| PEN  |       1.25 |       1.23 |    1.24 |         15000 |
| RUB  |       1.99 |       0.99 |    1.49 |          2700 |
| NULL |       5.00 |       0.48 |    2.14 |        195000 |
+------+------------+------------+---------+---------------+
```

The use of COALESCE:

```
[..]> SELECT COALESCE(NULL, NULL, 'Frist non-null argument', NULL,
'Second non-null argument');
```

Check it on your terminal.

Back to the table 'products':

```
[..]> SELECT COALESCE(productCode, 'Total') Code, MAX(price), MIN(price),
CAST(AVG(price) AS DECIMAL(7,2)) AS 'Average',
SUM(quantity)
FROM products
GROUP BY productCode
WITH ROLLUP;
```

```
+-------+------------+------------+---------+---------------+
| Code  | MAX(price) | MIN(price) | Average | SUM(quantity) |
+-------+------------+------------+---------+---------------+
| NTB   |       3.30 |       2.00 |    2.58 |        157000 |
| PAP   |       5.00 |       4.80 |    4.90 |          2300 |
| PEC   |       0.49 |       0.48 |    0.49 |         18000 |
| PEN   |       1.25 |       1.23 |    1.24 |         15000 |
| RUB   |       1.99 |       0.99 |    1.49 |          2700 |
| Total |       5.00 |       0.48 |    2.14 |        195000 |
+-------+------------+------------+---------+---------------+
```

Exercise 19. Complete the code, in order to get the results below:

```
[..]> SELECT COALESCE(productCode, '.......') ....... ,
SUM(..............) AS '........'
FROM products
GROUP BY ..........
WITH ROLLUP;
```

```
+-------+--------+
| Code  | Stock  |
+-------+--------+
| DIG   |     24 |
| NTB   | 152500 |
| PAP   |   2300 |
| PEC   |  19300 |
| PEN   |  14950 |
| RUB   |   2700 |
| Total | 191774 |
+-------+--------+
```

```
--
```

UPDATE - Modifying Data

To modify existing data, use UPDATE ... SET command, with the following syntax:

```
UPDATE tableName SET columnName = {value|NULL|DEFAULT}, ...
WHERE criteria;
```

Example. The table 'products' as-is:

```
[..]> SELECT * FROM products;
```

```
+-----------+-------------+---------------------+----------+-------+
| productID | productCode | name                | quantity | price |
+-----------+-------------+---------------------+----------+-------+
|      1001 | PEN         | Pen Red             |     5000 |  1.23 |
|      1002 | PEN         | Pen Blue            |     8000 |  1.25 |
|      1003 | PEN         | Pen Black           |     2000 |  1.25 |
|      1004 | PEC         | Pencil 2B           |    10000 |  0.48 |
...
|      1011 | PAP         | Paper A4E80/500     |     1200 |  4.90 |
```

```
|      1012 | RUB         | Rubber white Soft   |     1200 |  1.99 |
|      1013 | RUB         | Rubber color Medium |     1500 |  0.99 |
+-----------+-------------+---------------------+----------+-------+

Let's increase the  prices by 10% on all products
[..]> UPDATE products SET price = price * 1.10;
--
Check the changes:
[..]> SELECT * FROM products;
+-----------+-------------+---------------------+----------+-------+
| productID | productCode | name                | quantity | price |
+-----------+-------------+---------------------+----------+-------+
|      1001 | PEN         | Pen Red             |     5000 |  1.35 |
|      1002 | PEN         | Pen Blue            |     8000 |  1.38 |
|      1003 | PEN         | Pen Black           |     2000 |  1.38 |
|      1004 | PEC         | Pencil 2B           |    10000 |  0.53 |
|      1005 | PEC         | Pencil 2H           |     8000 |  0.54 |
|      1006 | NTB         | Notebook A4L90p     |     9000 |  3.63 |
|      1007 | NTB         | Notebook B5L60p     |     8000 |  3.08 |
|      1008 | NTB         | Notebook A5L50p     |    70000 |  2.42 |
|      1009 | NTB         | Notebook B6L40p     |    70000 |  2.20 |
|      1010 | PAP         | Paper A4W80/500     |     1100 |  5.50 |
|      1011 | PAP         | Paper A4E80/500     |     1200 |  5.39 |
|      1012 | RUB         | Rubber white Soft   |     1200 |  2.19 |
|      1013 | RUB         | Rubber color Medium |     1500 |  1.09 |
+-----------+-------------+---------------------+----------+-------+
13 rows..

Exercise 20.
Insert a new record into products:
1. The productID should be the set by the system.
2. The productCode is 'DIG' for 'digital'.
3. The name is 'iPad mini-4-WiFi-128'.
4. Quantity in stock: 4.
5. Unit price: 449.00
Note: Don't forget, that the first attribute productID is been insert by the
system. It's necessary to declare the attribute name for the values to be inserted
(2-5).
--
Result:
[..]> SELECT * FROM products;
+-----------+-------------+---------------------+----------+--------+
| productID | productCode | name                | quantity | price  |
+-----------+-------------+---------------------+----------+--------+
|      1001 | PEN         | Pen Red             |     5000 |   1.35 |
|      1002 | PEN         | Pen Blue            |     8000 |   1.38 |
...
```

```
|      1013 | RUB         | Rubber color Medium  |     1500 |   1.09 |
|      1014 | DIG         | iPad mini-4-Wi-Fi-128|        4 | 449.99 |
+-----------+-------------+----------------------+----------+--------+
14 rows..
--
[..]> UPDATE products SET quantity = quantity - 100
WHERE name = 'Pen Red';
--
Check the update:
[..]> SELECT * FROM products WHERE name = 'Pen Red';
+-----------+-------------+---------+----------+-------+
| productID | productCode | name    | quantity | price |
+-----------+-------------+---------+----------+-------+
|      1001 | PEN         | Pen Red |     4900 |  1.35 |
+-----------+-------------+---------+----------+-------+
--
Update more then one attribute value:
[..]> UPDATE products SET quantity = quantity + 50, price = 1.30
WHERE name = 'Pen Red';
-
[..]> SELECT * FROM products WHERE name = 'Pen Red';
+-----------+-------------+---------+----------+-------+
| productID | productCode | name    | quantity | price |
+-----------+-------------+---------+----------+-------+
|      1001 | PEN         | Pen Red |     4950 |  1.30 |
+-----------+-------------+---------+----------+-------+
1 row..
--
Exercise 21. Sales:
1. 100 A4-paper reams, each type, were sold: update the quantities.
2. 1500 notebooks of the sizes A5L and B6L each, were sold to schools and
highschools. Update quantities as mentioned above.

Notes: Use only 1 SQL statement for each update; 1 ream = 500 sheets.
Proceed with the above mentioned updates.
Result to be achieved:
+-----------+-------------+-----------------+----------+-------+
| productID | productCode | name            | quantity | price |
+-----------+-------------+-----------------+----------+-------+
|      1008 | NTB         | Notebook A5L50p |    68500 |  2.42 |
|      1009 | NTB         | Notebook B6L40p |    68500 |  2.20 |
|      1010 | PAP         | Paper A4W80/500 |     1000 |  5.50 |
|      1011 | PAP         | Paper A4E80/500 |     1100 |  5.39 |
+-----------+-------------+-----------------+----------+-------+
--
```

Check the example below:

```
[..]> SELECT productCode Code, name, price * quantity AS Balance
FROM products
ORDER BY 1, 3 DESC;
+------+----------------------+-----------+
| Code | name                 | Balance   |
+------+----------------------+-----------+
| DIG  | iPad mini-4-Wi-Fi-128 |   1799.96 |
| NTB  | Notebook A5L50p       | 165770.00 |
| NTB  | Notebook B6L40p       | 150700.00 |
| NTB  | Notebook A4L90p       |  32670.00 |
| NTB  | Notebook B5L60p       |  24640.00 |
| PAP  | Paper A4E80/500       |   5929.00 |
| PAP  | Paper A4W80/500       |   5500.00 |
| PEC  | Pencil 2B             |   5300.00 |
| PEC  | Pencil 2H             |   4320.00 |
| PEN  | Pen Blue              |  11040.00 |
| PEN  | Pen Red               |   6435.00 |
| PEN  | Pen Black             |   2760.00 |
| RUB  | Rubber white Soft     |   2628.00 |
| RUB  | Rubber color Medium   |   1635.00 |
+------+----------------------+-----------+
--
Exercise 22.
After all those updates, check the balance of the stocks for the accounting...
Calculate the value of all yours items grouped on 'productCode'.
[..]> SELECT COALESCE(productCode, .........) Code, .....(...... * .......)
AS Balance
FROM products
GROUP BY productCode
WITH ROLLUP;
+-------+-----------+
| Code  | Balance   |
+-------+-----------+
| DIG   |   1799.96 |
| NTB   | 351630.00 |
| PAP   |  11823.00 |
| PEC   |   9080.00 |
| PEN   |  19188.50 |
| RUB   |   4068.00 |
| Total | 397589.46 |
+-------+-----------+
Note (again..): Try using 'GROUP BY 1', an ordinal, instead 'GROUP BY
productCode', the attribute name and check the difference.
--
```

Exporting a database data to a text file.

The 'MYSQLDUMP' console utility is used to export databases to SQL text files.

```
OSX/Linux (use the 'root' account):
$ sudo su
[sudo] password for user:
# mysqldump -u root -p db > /<path>/db-bkup-$(date +%Y-%m-%d-%H%M).sql
Enter password:
# ls db-bkup*
db22-bkup-2020-06-15-1108.sql
#
Done!
--
Windows (use forward-slash (instead of back-slash) as path separator):
C:..> MYSQLDUMP -u root -p db-name > db-bkup-yymmdd.sql
-
C:..> mysqldump -u root db22 > C:/<path>/db-bkup-%date:~0,2%%date:~3,2%
%date:~6,4%.sql
--

Export just a table.
OS X/Linux (in this case no test was performed):
[..]> SELECT * FROM products INTO OUTFILE '/home/user/tb-bkup-yymmdd.csv'
COLUMNS TERMINATED BY ',';
--
Windows (use forward-slash as path separator):
[..]> SELECT * FROM products INTO OUTFILE 'c:/<path>/tb-bkup-yymmdd.csv'
COLUMNS TERMINATED BY ','
LINES TERMINATED BY '\r\n';
--

Importing a database (or single tables) from a text file.

OSX/Linux.
Importing the whole databse:
Create the new database:
[..]> CREATE DATABASE [db-new db21] charset latin1 collate
      latin1_swedish_ci;
-
Now use the terminal (command-line):
Check the directory to get the right file: 'db-bkup-yymmdd.sql'.
Use /home/user/ (e.g. '~')
Look for the dump file 'db-bkup-yymmdd.sql':
$ ll db-bkup-yymmdd.sql
-rw-r--r-- 1 user user 2683 May  3 13:21 db-bkup-yymmdd.sql
$ sudo [/usr/bin/]MYSQL -u root -p db21 '/<path>/db-bkup-yymmdd.sql'
```

```
or
$ sudo mysql -u root -p db21 < db-bkup-yymmdd.sql # use lower-case
--
For Windows:
C:..> mysql -u root db21 < c:/<path>/db-bkup-yymmdd.sql
--

Import comma-separated-values file ('.csv') into a table.

Insert/import tuples (in this case just 1 tuple) from a csv-file into the table
'products':
File: db-hp-calc-input.csv
\N,DIG,HP Prime,20,159.00
--
Important note:
'\N' stands for the entry of the 'productID', which will be set by the system, due
to the AUTO_INCREMENT option.
--
Rem.: To re-/set the AUTO_INCREMENT number (e.g. 'productID'), use:
[..]> ALTER TABLE products AUTO_INCREMENT = <last-entry + 1>;
--
Example: Import a tuple into 'products'.
[..]> LOAD DATA LOCAL INFILE '/<path>/db-hp-calc-input.csv'
INTO TABLE products COLUMNS TERMINATED BY ',';
--
Windows:
[..]> LOAD DATA LOCAL INFILE 'C:/<path>/db-hp-calc-input.csv'
INTO TABLE products COLUMNS TERMINATED BY ',';
--
Result:
[..]> SELECT * FROM products;
+-----------+-------------+----------------------+----------+--------+
| productID | productCode | name                 | quantity | price  |
+-----------+-------------+----------------------+----------+--------+
|      1001 | PEN         | Pen Red              |     4950 |   1.23 |
..
|      1014 | DIG         | iPad mini-4-Wi-Fi-128|        4 | 449.99 |
|      1015 | DIG         | HP Prime             |       20 | 159.00 |
+-----------+-------------+----------------------+----------+--------+
--
```

```
Import tab-separated-values file ('.tsv') into a table.

Load into the table 'products'.
Example using tsv-file (db-input-pencil.tsv).
OSX/Linux:
$ LOAD DATA LOCAL INFILE '/<path>/db-input-pencil.tsv' INTO TABLE
products COLUMNS TERMINATED BY '\t';
--
Windows:
[..]> LOAD DATA LOCAL INFILE 'C:/<path>/db-input-pencil.tsv'
INTO TABLE products COLUMNS TERMINATED BY '\t';
--
Exercise 23:
Create a tab-separeted-value-file (db-input-pencil.tsv) with following lines (no
empty-lines allowed):
PEC   Pencil  3B   500   0.52
PEC   Pencil  4B   200   0.62
PEC   Pencil  5B   100   0.73
PEC   Pencil  6B   500   0.47
--
Results:
[..]> SELECT * FROM products;
+-----------+-------------+---------------------+----------+--------+
| productID | productCode | name                | quantity | price  |
+-----------+-------------+---------------------+----------+--------+
|      1001 | PEN         | Pen Red             |     4950 |   1.23 |
|      1002 | PEN         | Pen Blue            |     8000 |   1.31 |
..
|      1017 | PEC         | Pencil 4B           |      200 |   0.62 |
|      1018 | PEC         | Pencil 5B           |      100 |   0.73 |
|      1019 | PEC         | Pencil 6B           |      500 |   0.47 |
+-----------+-------------+---------------------+----------+--------+
19 rows
--

LOAD - Loading records into a database

Example:
Use another database, e.g.'db[1..,2..,3..,..]', or other to preservethe database
'db2' used for the examples above (before exercise 23).
E.g., use 'db21'...
Now, check the status-quo of 'db2' (you will get the tuples imported before
(example above)):
```

```
[db2]> SELECT * FROM products;
+-----------+-------------+----------------------+----------+--------+
| productID | productCode | name                 | quantity | price  |
+-----------+-------------+----------------------+----------+--------+
|      1001 | PEN         | Pen Red              |     4950 |   1.23 |
|      1002 | PEN         | Pen Blue             |     8000 |   1.31 |
|      1003 | PEN         | Pen Black            |     2000 |   1.31 |
|      1004 | PEC         | Pencil 2B            |    10000 |   0.50 |
|      1005 | PEC         | Pencil 2H            |     8000 |   0.51 |
|      1006 | NTB         | Notebook A4L90p      |     7500 |   3.47 |
|      1007 | NTB         | Notebook B5L 60p     |     8000 |   2.94 |
|      1008 | NTB         | Notebook A5L 50p     |    68500 |   2.31 |
|      1009 | NTB         | Notebook B6L 40p     |    68500 |   2.10 |
|      1010 | PAP         | Paper A4 white 80/500|     1100 |   5.25 |
|      1011 | PAP         | Paper A4 eco 80/500  |     1200 |   5.04 |
|      1012 | RUB         | Rubber white Soft    |     1200 |   2.09 |
|      1013 | RUB         | Rubber color Medium  |     1500 |   1.04 |
|      1014 | DIG         | iPad mini-4-Wi-Fi-128|        4 | 449.99 |
|      1015 | DIG         | HP Prime             |       20 | 159.00 |
+-----------+-------------+----------------------+----------+--------+

Using a sql-script and SOURCE to reset the table 'products' and importthe tuples.
From the dump into 'db21'.

AS-IS status.
First of all, check 'db21':
[db21]> SELECT * FROM products;
+-----------+-------------+----------------------+----------+--------+
| productID | productCode | name                 | quantity | price  |
+-----------+-------------+----------------------+----------+--------+
|      1001 | PEN         | Pen Red              |     4950 |   1.23 |
|      1002 | PEN         | Pen Blue             |     8000 |   1.31 |
|      1003 | PEN         | Pen Black            |     2000 |   1.31 |
|      1004 | PEC         | Pencil 2B            |    10000 |   0.50 |
|      1005 | PEC         | Pencil 2H            |     8000 |   0.51 |
|      1006 | NTB         | Notebook A4L90p      |     7500 |   3.47 |
|      1007 | NTB         | Notebook B5L 60p     |     8000 |   2.94 |
|      1008 | NTB         | Notebook A5L 50p     |    68500 |   2.31 |
|      1009 | NTB         | Notebook B6L 40p     |    68500 |   2.10 |
|      1010 | PAP         | Paper A4 white 80/500|     1100 |   5.25 |
|      1011 | PAP         | Paper A4 eco 80/500  |     1200 |   5.04 |
|      1012 | RUB         | Rubber white Soft    |     1200 |   2.09 |
|      1013 | RUB         | Rubber color Medium  |     1500 |   1.04 |
|      1014 | DIG         | iPad mini-4-Wi-Fi-128|        4 | 449.99 |
|      1015 | DIG         | HP Prime             |       20 | 159.00 |
+-----------+-------------+----------------------+----------+--------+
15 rows.. (it is a dump of db2 before last insertions.)
```

```
Now, create the sql-script 'load_items.sql'.
Linux:
~$ nano load_items.sql         # you may use another basic text editor
DELETE FROM products;
INSERT INTO products VALUES (2001, 'PEC', 'Pencil 3B', 500, 0.52),
                            (NULL, 'PEC', 'Pencil 4B', 200, 0.62),
                            (NULL, 'PEC', 'Pencil 5B', 100, 0.73),
                            (NULL, 'PEC', 'Pencil 6B', 500, 0.47),
                            (NULL, 'PEC', 'Pencil AA', 500, 0.45),
                            (NULL, 'PEC', 'Pencil AB', 500, 0.45);
SELECT * FROM products;
~$
--
Back to 'db21' enter:
[db21]> source /home/ck/load_items.sql
Query OK, 15 rows affected (0.003 sec)

Query OK, 6 rows affected (0.004 sec)
Records: 6  Duplicates: 0  Warnings: 0

+-----------+-------------+-----------+----------+-------+
| productID | productCode | name      | quantity | price |
+-----------+-------------+-----------+----------+-------+
|      2001 | PEC         | Pencil 3B |      500 |  0.52 |
|      2002 | PEC         | Pencil 4B |      200 |  0.62 |
|      2003 | PEC         | Pencil 5B |      100 |  0.73 |
|      2004 | PEC         | Pencil 6B |      500 |  0.47 |
|      2005 | PEC         | Pencil AA |      500 |  0.45 |
|      2006 | PEC         | Pencil AB |      500 |  0.45 |
+-----------+-------------+-----------+----------+-------+
6 rows..
Done!! This is the replaced 'db21'.
-

Windows:
1. Use 'notepad', or 'notepad++' to create the script shown above and save it on
'Documents'.
2. Start or return the DBMS and enter:
[db21]> source C:/Users/<user-name>/Documents/load_products.sql
Note: Use Unix-style forward slash (/) as directory separator.
Rem.: Some Windows systems use the following syntax:
[db21]> source C:\Users\<user-name>\Documents\load_products.sql

The results should be the same as above. Check it...
-
```

Exercise 24:
Perform the task above and replace the tuples of your database 'db21' with the tuples above.
--
Related issues, the 'batch mode':
Using the 'batch mode' of the DBMS client program, is possible to re-direct the input from the script into the database:
-
Linux:
$ cd /usr/local/mysql/bin
$ ./mysql -u root -p db21 < /home/<user-name>/Documents/load_items.sql
or
Windows:
> cd 'path-to-mysql-bin'
> mysql -u root -p db21 < C:\Users\<user-name>\load_items.sql
--

Improving the DBMS: The table 'suppliers'.

The One-To-Many Relationship.

Foreign Key

A foreign key of a child table is used to reference the parent table. The foreign key constraint can be imposed to ensure so-called referential integrity - values in the child table must be valid values in the parent table.
--
Child table definition:
The foreign key will be defined when defining the child table, which references a parent table, as follows:

```
CREATE TABLE tableName (
......
CONSTRAINT constraintName FOREIGN KEY (columName)
   REFERENCES parentTableName (columnName)
-- On DELETE reference action:
[ON DELETE RESTRICT | CASCADE | SET NULL | NO ACTION]
-- On UPDATE reference action
[ON UPDATE RESTRICT | CASCADE | SET NULL | NO ACTION]
);
```

The reference action for UPDATE and DELETE via the optional ON UPDATE and ON DELETE clauses can be specified:
RESTRICT (default): disallow DELETE or UPDATE of the parent's row, if there are matching rows in child table.
-
CASCADE: cascade the DELETE or UPDATE action to the matching rows inthe child table.

-
SET NULL: set the foreign key value in the child table to NULL(if NULL is allowed).
-
NO ACTION: a SQL term which means no action on the parent's row.
Same as RESTRICT in MySQL, which disallows DELETE or UPDATE (donothing).
--

Suppose that each product has one supplier, and each supplier suppliesone or more products. To do...
Exercise 25:
1. Draw the Entity-Relationship-Model (ERM) of 'products' and 'suppliers'.
2. Create a table 'suppliers' to store suppliers data (e.g., supplierID, name [, address] and phone number).
Note: Use the database 'db21' (or your your equivalent database); The attribute 'address' is optional and will stay commented for this example.
3. Define a key-attribute with unique values, 'supplierID' (or 'suppID') to identify every supplier.
4. Set 'suppID' as the primary key for the table 'suppliers' (to ensure uniqueness and facilitate fast search).
5. Define the relationship between 'suppliers' and 'products': a new column (attribute) should be added into 'product' - the supp[lier]ID.
6. Set the 'supplierID' (or 'suppID') attribute of the table 'products' as a foreign key, to reference the 'supp[lier]ID' attributeof the table 'suppliers' to ensure the so-called 'referential integrity'.
--
Check your work:
```
[..]> DESC suppliers;
+--------+-----------------+------+-----+---------+----------------+
| Field  | Type            | Null | Key | Default | Extra          |
+--------+-----------------+------+-----+---------+----------------+
| suppID | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| name   | char(30)        | NO   |     |         |                |
| phone  | char(9)         | NO   |     |         |                |
+--------+-----------------+------+-----+---------+----------------+
```
--
Exercise 26:
Insert into 'suppliers' the following suppliers:
Paper & Office Supplies Co., phone: 123456789;
Books & School Ltd, phone: 234567890;
Electronics Corp., 345678901.
--

```
Check it:
[..]> SELECT * FROM suppliers;
+--------+----------------------------+-----------+
| suppID | name                       | phone     |
+--------+----------------------------+-----------+
|    501 | Paper & Office Supplies Co. | 123456789 |
|    502 | Books & School Ltd          | 234567890 |
|    503 | Electronics Corp.           | 345678901 |
+--------+----------------------------+-----------+
--

Relationships in the (Object-)Relational Model.

Define and establish the relationship between the table 'products' and the table
'suppliers'.

ALTER TABLE

Use 'ALTER TABLE' to add a new column 'suppID' into the table 'products':
[..]> ALTER TABLE products ADD COLUMN suppID INT UNSIGNED NOT NULL;
Query OK, 4 rows affected (0.13 sec)
Records: 4
-
[..]> DESC products;
+-------------+------------------+------+-----+---------+----------------+
| Field       | Type             | Null | Key | Default | Extra          |
+-------------+------------------+------+-----+---------+----------------+
| productID   | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| productCode | char(3)          | NO   |     |         |                |
| name        | varchar(30)      | NO   |     |         |                |
| quantity    | int(10) unsigned | NO   |     | 0       |                |
| price       | decimal(7,2)     | NO   |     | 99.99   |                |
| suppID      | int(10) unsigned | NO   |     | NULL    |                |
+-------------+------------------+------+-----+---------+----------------+
6 rows..
--
Referential Integrity:
Next, add a foreign key constraint on the 'suppID' column of the 'products' child
table to the 'suppliers' parent table, to ensure thatevery 'suppID' in the
'products' always refers to a valid 'suppID' in 'suppliers' - this is called
REFERENTIAL INTEGRITY.

Note: The attribute names in both tables are not the same, since thedatabase
system use the syntax 'database-name.table-name' to reference the tables.
```

```
Setting the FOREIGN KEY:
Before adding the foreign key in 'products', set the 'suppID' of  the existing
records in 'products' to a VALID 'suppID' of 'suppliers':
Check it:
[..]> SELECT * FROM products;
+-----------+-------------+-----------+----------+-------+--------+
| productID | productCode | name      | quantity | price | suppID |
+-----------+-------------+-----------+----------+-------+--------+
|      2001 | PEC         | Pencil 3B |      500 |  0.52 |      0 |
|      2002 | PEC         | Pencil 4B |      200 |  0.62 |      0 |
|      2003 | PEC         | Pencil 5B |      100 |  0.73 |      0 |
|      2004 | PEC         | Pencil 6B |      500 |  0.47 |      0 |
|      2005 | PEC         | Pencil AA |      500 |  0.45 |      0 |
|      2006 | PEC         | Pencil AB |      500 |  0.45 |      0 |
+-----------+-------------+-----------+----------+-------+--------+
6 rows..

[..]> UPDATE products SET suppID = 501;
Query OK, 6 rows affected (0.011 sec)
Rows matched: 6  Changed: 6  Warnings: 0

[..]> UPDATE products SET suppID = 501;
Query OK, 6 rows affected (0.010 sec)
Rows matched: 6  Changed: 6  Warnings: 0

[..]> SELECT * FROM products;
+-----------+-------------+-----------+----------+-------+--------+
| productID | productCode | name      | quantity | price | suppID |
+-----------+-------------+-----------+----------+-------+--------+
|      2001 | PEC         | Pencil 3B |      500 |  0.52 |    501 |
|      2002 | PEC         | Pencil 4B |      200 |  0.62 |    501 |
|      2003 | PEC         | Pencil 5B |      100 |  0.73 |    501 |
|      2004 | PEC         | Pencil 6B |      500 |  0.47 |    501 |
|      2005 | PEC         | Pencil AA |      500 |  0.45 |    501 |
|      2006 | PEC         | Pencil AB |      500 |  0.45 |    501 |
+-----------+-------------+-----------+----------+-------+--------+
6 rows..
--
Add a foreign key constraint:
[..]> ALTER TABLE products ADD FOREIGN KEY (suppID)
      REFERENCES suppliers(suppID);
Query OK..
```

```
[..]> DESCRIBE products; (or, [..]> DESC products;)
+-------------+------------------+------+-----+---------+------------+
| Field       | Type             | Null | Key | Default | Extra      |
+-------------+------------------+------+-----+---------+------------+
...
| suppID      | int(10) unsigned | NO   | MUL | NULL    |            |
+-------------+------------------+------+-----+---------+------------+
Note: 'MUL' comes from "MULtiple" because multiple occurrences of the same value
are allowed.
-
[..]> ALTER TABLE products ADD FOREIGN KEY (suppID)
      REFERENCES suppliers(suppID);
Query OK, 6 rows affected (0.047 sec)
Records: 6  Duplicates: 0  Warnings: 0

Exercise 27:
Update suppliers: For the productID 2004, set the supplier
'Books & School Ltd'.
-
Check it:
[..]> SELECT * FROM products;
+-----------+-------------+-----------+----------+-------+--------+
| productID | productCode | name      | quantity | price | suppID |
+-----------+-------------+-----------+----------+-------+--------+
...
|      2004 | PEC         | Pencil 6B |      500 |  0.47 |    502 |
...
+-----------+-------------+-----------+----------+-------+--------+
--

SELECT with JOIN

SELECT command can be used to query and join data from two related tables.
For example, to list the product's name (from 'products') and supplier's name
(from 'suppliers'), a JOIN operation via the two common attributes, namelly
'products.suppID' and 'suplliers.suppID':

The ANSI style: 'JOIN ... ON ...'
[..]> SELECT products.name, price, suppliers.name
FROM products
JOIN suppliers ON products.suppID = suppliers.suppID
WHERE price > 0.45;
```

```
+-----------+-------+----------------------------+
| name      | price | name                       |
+-----------+-------+----------------------------+
| Pencil 3B |  0.52 | Paper & Office Supplies Co. |
| Pencil 4B |  0.62 | Paper & Office Supplies Co. |
| Pencil 5B |  0.73 | Paper & Office Supplies Co. |
| Pencil 6B |  0.47 | Books & School Ltd         |
+-----------+-------+----------------------------+
4 rows..
Note: Use 'products.name' and 'suppliers.name' to differentiate the two names.

Join via WHERE clause (LEGACY FORMAT and not recommended by some authors):
[..]> SELECT products.name, price, suppliers.name
FROM products, suppliers
WHERE products.suppID = suppliers.suppID AND price < 0.6;
+-----------+-------+----------------------------+
| name      | price | name                       |
+-----------+-------+----------------------------+
| Pencil 3B |  0.52 | Paper & Office Supplies Co. |
| Pencil 6B |  0.47 | Books & School Ltd         |
| Pencil AA |  0.45 | Paper & Office Supplies Co. |
| Pencil AB |  0.45 | Paper & Office Supplies Co. |
+-----------+-------+----------------------------+
4 rows..

Use ALIASES (AS) to avoid ambiguity:
[..]> SELECT products.name AS 'Product Name', price AS Price,
suppliers.name AS 'Supplier Name'
FROM products
JOIN suppliers ON products.suppID = suppliers.suppID
WHERE price < 0.6;
+--------------+-------+----------------------------+
| Product Name | Price | Supplier Name              |
+--------------+-------+----------------------------+
| Pencil 3B    |  0.52 | Paper & Office Supplies Co. |
| Pencil 6B    |  0.47 | Books & School Ltd         |
| Pencil AA    |  0.45 | Paper & Office Supplies Co. |
| Pencil AB    |  0.45 | Paper & Office Supplies Co. |
+--------------+-------+----------------------------+
4 rows..
--
Use aliases for table names too:
[..]> SELECT p.name AS 'Product Name', p.price, s.name AS 'Supplier Name'
FROM products AS p
JOIN suppliers AS s ON p.suppID = s.suppID
WHERE p.price < 0.50;
```

```
+--------------+-------+-----------------------------+
| Product Name | price | Supplier Name               |
+--------------+-------+-----------------------------+
| Pencil 6B    |  0.47 | Books & School Ltd          |
| Pencil AA    |  0.45 | Paper & Office Supplies Co. |
| Pencil AB    |  0.45 | Paper & Office Supplies Co. |
+--------------+-------+-----------------------------+
3 rows..
--
```

The Many-To-Many Relationship

Suppose that a product has many suppliers; and one supplier may supplies many products in a so-called MANY-TO-MANY RELATIONSHIP. In this case the above shown solution is useless...

It is not possible to include more 'suppID's for the same product inthe 'products' table , without creating reduncy and compromising thedefined primary-key 'products.suppID'.
Similarly, no inclusion of the 'productID' in the 'suppliers' tables possible, the number of products by suppliers is not definable.

To solve this problem, a new table will be created, known as a JUNCTION TABLE (or JOINT TABLE), to provide the linkage.
Let's call the joint table 'products_supppliers':
1. The primary key of the table consists of two columns: 'productID' and 'suppID', as their combination uniquely identifies each rows.
2. Two foreign keys will be defined to set the constraint to the two parent tables.
--
Exercise 28:
Draw the ERM for these 3 relations.
--
Define the JOINT TABLE:

```
[..]> CREATE TABLE products_suppliers (
productID   INT UNSIGNED  NOT NULL,
suppID  INT UNSIGNED  NOT NULL,
/*Same data types as the parent tables */
PRIMARY KEY (productID, suppID), /* uniqueness */
FOREIGN KEY (productID)  REFERENCES products  (productID),
FOREIGN KEY (suppID) REFERENCES suppliers (suppID)
);
```

```
Check it:
[..]> DESC products_suppliers;
+-----------+------------------+------+-----+---------+-------+
| Field     | Type             | Null | Key | Default | Extra |
+-----------+------------------+------+-----+---------+-------+
| productID | int(10) unsigned | NO   | PRI | NULL    |       |
| suppID    | int(10) unsigned | NO   | PRI | NULL    |       |
+-----------+------------------+------+-----+---------+-------+
2 rows
--
Exercise 29. Now, establish the relationships, as defined bellow :
1. Supplier 'Paper & Office Supplies Co.', supplies product(s):
Pencil 3B, Pencil 4B, Pencil 5B.
2. Supplier 'Books & School Ltd', supplies product(s):
Pencil 6B, Pencil AA, Pencil AB.
3. Supplier 'Electronics Corp.', supplies product(s):
Pencil 3B, Pencil AA.
-
Check it:
[..]> SELECT * FROM products_suppliers;
+-----------+--------+
| productID | suppID |
+-----------+--------+
|      2001 |    501 |
|      2001 |    503 |
|      2002 |    501 |
|      2003 |    501 |
|      2004 |    502 |
|      2005 |    502 |
|      2005 |    503 |
|      2006 |    502 |
+-----------+--------+
8 rows..
-
Note: The values in the foreign-key columns (of the child table) must match valid
values in the columns they reference (of the parent table).
--
Important note:
Remove the 'suppID' column from the table 'products': This attribute was added to
establish the 1:n (one-to-many) relationship between 'products' and suppliers'.
The attribute 'products.suppID' is no longer needed due to the n:m (many-to-many)
relationship establish thru the new table 'products_suppliers'.
First of all remove the foreign key defined on 'products.suppID'.
To remove a key in MariaDB/MySQL the constraint's name, generated bythe system, is
needed.

To get the constraint name, issue a 'SHOW CREATE TABLE products \G'.
```

```
The foreign key's constraint name is in the clause
"CONSTRAINT constraint_name FOREIGN KEY ....".
Check it:
[..]> SHOW CREATE TABLE products \G
*************************** 1. row ***************************
       Table: products
Create Table: CREATE TABLE products (
  productID int(10) unsigned NOT NULL AUTO_INCREMENT,
  productCode char(3) NOT NULL DEFAULT '',
  name varchar(30) NOT NULL DEFAULT '',
  quantity int(10) unsigned NOT NULL DEFAULT 0,
  price decimal(7,2) NOT NULL DEFAULT 99.99,
  suppID int(10) unsigned DEFAULT NULL,
  PRIMARY KEY (productID),
  KEY supplierID (suppID),
  CONSTRAINT products_ibfk_1 FOREIGN KEY (suppID) REFERENCES suppliers
  (suppID)
) ENGINE=InnoDB AUTO_INCREMENT=2008 DEFAULT CHARSET=latin1
1 row..
--
So, drop the foreign key using:
[..]> ALTER TABLE products DROP FOREIGN KEY products_ibfk_1;
Query OK, 0 rows affected (0.006 sec)
Records: 0  Duplicates: 0  Warnings: 0
--
Check it:
[..]> SHOW CREATE TABLE products \G
*************************** 1. row ***************************
       Table: products
Create Table: CREATE TABLE `products` (
  `productID` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `productCode` char(3) NOT NULL DEFAULT '',
  `name` varchar(30) NOT NULL DEFAULT '',
  `quantity` int(10) unsigned NOT NULL DEFAULT 0,
  `price` decimal(7,2) NOT NULL DEFAULT 99.99,
  `suppID` int(10) unsigned NOT NULL,
  PRIMARY KEY (`productID`),
  KEY `suppID` (`suppID`)
) ENGINE=InnoDB AUTO_INCREMENT=2008 DEFAULT CHARSET=latin1
1 row in set (0.000 sec)
-
Done! The constraint is gone...
--
Now, remove the column redundant 'suppID' column in 'products':
[..]> ALTER TABLE products DROP suppID;
Query OK, 0 rows affected (0.015 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
--
Check it:
[..]> DESC products;
+-------------+------------------+------+-----+---------+----------------+
| Field       | Type             | Null | Key | Default | Extra          |
+-------------+------------------+------+-----+---------+----------------+
| productID   | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| productCode | char(3)          | NO   |     |         |                |
| name        | varchar(30)      | NO   |     |         |                |
| quantity    | int(10) unsigned | NO   |     | 0       |                |
| price       | decimal(7,2)     | NO   |     | 99.99   |                |
+-------------+------------------+------+-----+---------+----------------+
5 rows..
-
Done!
--

Important notes related to 'foreign key'.

Example:
Try deleting a record in the 'parent table' suppliers that is referenced by
products_suppliers ('child table') table, e.g.:
[..]> DELETE FROM suppliers WHERE suppID = 501;
ERROR 1451 (23000): Cannot delete or update a parent row:
a foreign key constraint fails (`db22`.`products_suppliers`,
CONSTRAINT `products_suppliers_ibfk_2` FOREIGN KEY (`suppID`)
REFERENCES `suppliers` (`suppID`))
-
The record cannot be deleted as the default 'ON DELETE RESTRICT' constraint was
imposed.
--

Querying

As seen before, use SELECT with JOIN to query data from the 3 tables, for
examples:
[..]> SELECT products.name AS 'Product Name', price,
suppliers.name AS 'Supplier Name'
FROM products_suppliers
JOIN products  ON products_suppliers.productID = products.productID
JOIN suppliers ON products_suppliers.suppID = suppliers.suppID
WHERE price < 0.6;
```

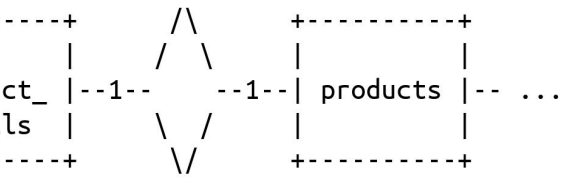
```
+--------------+-------+----------------------------+
| Product Name | price | Supplier Name              |
+--------------+-------+----------------------------+
| Pencil 3B    |  0.52 | Paper & Office Supplies Co. |
| Pencil 6B    |  0.47 | Books & School Ltd         |
| Pencil AA    |  0.45 | Books & School Ltd         |
| Pencil AB    |  0.45 | Books & School Ltd         |
| Pencil 3B    |  0.52 | Electronics Corp.          |
| Pencil AA    |  0.45 | Electronics Corp.          |
+--------------+-------+----------------------------+
6 rows..

Or using WHERE clause to join (STANDARD SQL TYPE):
[..]> SELECT products.name AS 'Product Name', price,
suppliers.name AS 'Supplier Name'
FROM products, suppliers, products_suppliers
WHERE products_suppliers.productID = products.productID
AND products_suppliers.suppID = suppliers.suppID
AND price < 0.6;
+--------------+-------+----------------------------+
| Product Name | price | Supplier Name              |
+--------------+-------+----------------------------+
| Pencil 3B    |  0.52 | Paper & Office Supplies Co. |
..
| Pencil AA    |  0.45 | Electronics Corp.          |
+--------------+-------+----------------------------+
6 rows..
--
Define aliases for table names too:
[..]> SELECT p.name AS 'Product Name', s.name AS 'Supplier Name'
FROM products_suppliers AS ps
JOIN products AS p ON ps.productID = p.productID
JOIN suppliers AS s ON ps.suppID = s.suppID
WHERE p.name = 'Pencil 3B';
+--------------+----------------------------+
| Product Name | Supplier Name              |
+--------------+----------------------------+
| Pencil 3B    | Paper & Office Supplies Co. |
| Pencil 3B    | Electronics Corp.          |
+--------------+----------------------------+
2 rows..
--
Exercise 30:
Write the example above in STANDARD SQL STYLE and check the results.
Note: The same results are expected.
--
```

Exercise 31. Translate from the STANDARD SQL TYPE to the ANSI STYLE:

```
[..]> SELECT p.name AS 'Product Name', s.name AS 'Supplier Name'
FROM products AS p, products_suppliers AS ps, suppliers AS s
WHERE p.productID = ps.productID
AND ps.suppID = s.suppID
AND s.name = 'Paper & Office Supplies Co.';
+--------------+-----------------------------+
| Product Name | Supplier Name               |
+--------------+-----------------------------+
| Pencil 3B    | Paper & Office Supplies Co. |
| Pencil 4B    | Paper & Office Supplies Co. |
| Pencil 5B    | Paper & Office Supplies Co. |
+--------------+-----------------------------+
3 rows..
--
```

One-To-One Relationship

Suppose that some products have optional data (e.g., type,comment,..).
Instead of keeping these optional data in the products table, it is more efficient to create another table called product_details, and link it to products with a one-to-one relationship, as illustrated:




```
[..]> CREATE TABLE product_details (
productID  INT UNSIGNED NOT NULL,
comment    TINYTEXT  NULL,
PRIMARY KEY (productID),
FOREIGN KEY (productID) REFERENCES products (productID)
);
```

Note: The TINYTEXT get a maximum length of 255 (= 2^8 - 1) characters.
It is possible to use TEXT in order to get longer textes:
65,535 (= 2^{16} - 1) characters.

```
Check the new table:
[..]> DESCRIBE product_details;
+-----------+-----------------+------+-----+---------+-------+
| Field     | Type            | Null | Key | Default | Extra |
+-----------+-----------------+------+-----+---------+-------+
| productID | int(10) unsigned | NO   | PRI | NULL    |       |
| comment   | tinytext         | YES  |     | NULL    |       |
+-----------+-----------------+------+-----+---------+-------+
2 rows..
-
and
[..]> SHOW CREATE TABLE product_details \G
*************************** 1. row ***************************
       Table: product_details
Create Table: CREATE TABLE product_details (
  productID int(10) unsigned NOT NULL,
  comment tinytext DEFAULT NULL,
  PRIMARY KEY (productID),
  CONSTRAINT product_details_ibfk_1 FOREIGN KEY (productID)
  REFERENCES products (productID)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
1 row..
--
Exercise 32:
1. Create a 'csv' file (e.g. 'product_details.csv') with all needed details of the
tuples in 'products'.
2. From the SQL terminal import the data into 'product_details'.
--

Further important issues related to Object-Relational DBMS (MySQL and MariaDB).

Check in the Net the definitions of ENGINE=INNODB and ENGINE=MYISAM related to
'CREATE DATABASE/TABLE ...' and their and differences.
-
Example. Check some of your database system's information:
[..]> SELECT TABLE_SCHEMA,TABLE_NAME,TABLE_TYPE, ENGINE
FROM INFORMATION_SCHEMA.TABLES
WHERE table_schema = '<the-name-of-your-dbms>';
or
[..]> SELECT ENGINE,SUPPORT,TRANSACTIONS,XA,SAVEPOINTS
FROM INFORMATION_SCHEMA.ENGINES;
--------------------------------------------------------------------------------
```