EI1020/MT1020. Databases   Academic Year 2020-21

Session 4. Simple SQL Queries with `SELECT`, `FROM`, and `WHERE`

Document version: 2021-04-12

1 Objectives

The objectives of this session are the following:

- To be able to formulate basic `SELECT` commands (on a single table) that employ the `SELECT`, `FROM`, `WHERE`, `ORDER BY`, and `FETCH FIRST` clauses.
- To be able to determine when the `DISTINCT` clause is needed.
- To be able to handle null values with the `IS NULL` operator and the `COALESCE` function.

2 Some advice before starting this session

Create a working directory with a proper name for this session (such as `s04`) inside your `ei1020` or `mt1020` directory). Jump into it before running `psql` so that all files are created inside it. Save each SQL query in a different file with the `.sql` extension. Use a naming system for the files containing queries that can be employed in all sessions. For instance, the `s04_ej01.sql` file will contain the exercise 1 of session 4.

The database to work on in this session (and next ones, unless otherwise stated) is `ei1020`. This database contains several tables already populated with up to thousands of rows.

To connect to it from both the GNU/Linux classroom computers and `lynx`, you must run the command:

```
alxxxxxx@:∼/EI1020$ psql -h db-aules.uji.es ei1020
```

Attention !!! **It is very important that you execute all the commands inside the `s04` subfolder of the `ei1020` or `mt1020` folder**. If you are not inside it, execute the next command before the `psql` command.

```
alxxxxxx@:∼/EI1020$ cd ei1020/s04
```

Before writing each query, you can check in the course website the results your query should obtain, so that you can find out if you have understood the problem statement correctly.

3 In the previous session . . .

As known, the SELECT command consists of several parts, some of which have already been introduced. The format of the SELECT sentence is the following:

```
SELECT [DISTINCT] { * | column [, column]}
FROM   table
[WHERE search_condition]
[ORDER BY column [ASC|DESC] [,column [ASC|DESC] ]
[FETCH FIRST n ROWS ONLY ];
```

Now we are going to review them in the same order as they are processed during the execution:

- The FROM clause specifies the table the query must work on.
- The WHERE clause specifies a condition that the rows must satisfy to be displayed. This clause is very useful to restrict the result to a subset of the table rows. This condition is a boolean predicate that may include AND and OR operators.
- The SELECT clause specifies the columns to be displayed in the result. To show all columns, employ *.
- The DISTINCT modifier can be employed after the SELECT word to not display duplicate rows in the result. This can only happen when the SELECT query does not include the table's primary key (or a unique key).

4 In this session . . .

You will practice the SELECT command to query the ei1020 database. You should know its structure in depth, and you should always keep the document explaining it on hand.

4.1 ORDER BY clause

If this clause is included, it is the last one (or the second-to-last one if the FETCH clause is included) in the SELECT command. It is used to sort the results. The result can be sorted in ascending or descending way, and the sorting can be based on either a single column or multiple columns.

In the following example the query shows the data of all the customers sorted by postal code (descending) and all of them which have the same postal code are sorted by name (ascending). It is not necessary to execute the example.

```
SELECT *
FROM   clientes
ORDER  BY codpostal DESC, nombre;
```

4.2 FETCH FIRST n ROWS ONLY clause

If included in a sentence, this clause limits the number of rows shown in the result. Though this clause belongs to the SQL:2008 standard, some RDBMS do not include it. Fortunately, most of them include a similar clause with the same function called `LIMIT`, `TOP`, etc.

For example, the following sentence shows the five highest prices of the articles in our database.

```sql
SELECT precio
FROM   articulos
ORDER  BY precio DESC
FETCH  FIRST 5 ROWS ONLY;
```

4.3 Expressions in `SELECT` and `WHERE` clauses

In addition to columns, you can also include expressions containing columns and constants in the `SELECT` and `WHERE` clauses. The columns and expressions specified in the `SELECT` clause can be labelled in the result by using the `AS` keyword.

If the result of a SQL query has to be sorted by the value of an expression already in the `SELECT` clause, it can be specified in the `ORDER BY` clause by using its position in the `SELECT` clause (or even by its alias).

```sql
 SELECT precio, ROUND(precio * 0.8, 2) AS rebajado
 FROM   articulos
 ORDER  BY 2;
```

4.4 Null values

When a cell has no value, its value is null. A null is not a value: its means no value. To find out if a column or an expression is null, the `IS NULL` operator must be used. Accordingly, to find out if a column or an expression is not null, the `IS NOT NULL` operator must be used. Logical expressions such as `... = NULL` or `... != NULL` are wrong.

When running a SQL query, null values can be converted into usual values by using the `COALESCE( expression, value_if_null )` function. This function returns `value_if_null` if `expression` is null; otherwise, it returns the value of `expression`. That is, it returns the first non-null argument.

⋆ **Exercise 1**: Read the following code. What are the differences between the `iva` column and the `iva_null` column? Write your answer before running the query. Then, execute the query and check your answer.

```sql
SELECT codfac, fecha, codcli,
       COALESCE( iva, 0 ) AS iva, iva AS iva_null,
       COALESCE( dto, 0 ) AS dto
FROM   facturas
WHERE  codcli < 50
AND    ( iva = 0 OR iva IS NULL );
```

Note that the `( iva = 0 OR iva IS NULL )` expression can be shortened to `COALESCE( iva, 0 ) = 0`.

Solution: ....

....

⋆ **Exercise 2**: Rewrite the following example to display the invoice code and the client code, showing a -1 if the invoice has no client.

```sql
SELECT codfac, codcli
FROM   facturas;
```

Solution: ....

....

> **Important note:** All previous exercises must be done at home before going to the lab class. The remaining exercises will be done during the lab class.

5 Session Exercises

The exercises in this section must be done individually. You can check if your SQL sentences obtain the expected result by comparing your results with those in this course's website.

⋆ **Exercise 3**: Write a sentence that shows the article code and the invoice code of the invoice lines whose amounts requested are one and whose unit prices are over 100 €. Sort the result by the article code and then by the invoice code.

In the result there is a row for each .... that meets the restrictions imposed.

⋆ **Exercise 4**: Show the code and description of those articles with stocks below their minimum stock. Besides, show the number of units needed to reach that minimum. Sort the result by the article description.

In the result there is a row for each .... that meets the restriction imposed.

⋆ **Exercise 5**: What types of VAT (IVA) have been applied to the invoices? You have to remember that a null VAT should be interpreted as zero. Additionally, note that no specific ordering of the results is required for this query.
Do you need the `ORDER BY` clause if you want the results sorted ascending? ....
In the result there is a row for each ....

⋆ **Exercise 6**: The `facturas` table contains a foreign key to the `clientes` table: `facturas.codcli`. This foreign key refers to the customer the invoice belongs to. Look at the table structure (execute `\d facturas`), and answer the following question: Does this foreign key accept nulls? ....

A null in this foreign key means that the client the invoice belongs to is unknown.

Write a SQL query to show the codes of the invoices whose customers are unknown.

⋆ **Exercise 7**: What does the following statement do?

```sql
SELECT *
FROM   articulos
WHERE  stock > stock_min * 3
```

```
AND     precio > 6
ORDER   BY codart;
```

Solution: . . . .

⋆ **Exercise 8**: Do you always know when the `DISTINCT` clause is required?

With this exercise we will show an easy strategy you can apply. Let us suppose we want to show the article codes on those invoice lines with invoice codes between 5776 and 5781. Answer the following questions and you will learn how to plan the query:

(a) What table must the sentence work on? . . . .

(b) The working table has one row for each . . . .

(c) In the result each row represents a . . . .

(d) If the rows in (b) do not represent the same as the rows in (c), we have to use `DISTINCT` clause. Which is the case for this exercise? . . . .

(e) Execute the query now and check your answer.

(f) Remove the `DISTINCT` clause and execute again the query. Is the number of rows different? . .

⋆ **Exercise 9**: Show all the data of articles whose prices are less than 5 cents. For each article, instead of the original price, display as new price twice its value. Sort the results by article code.

⋆ **Exercise 10**: Show the invoice code, line number, article code, quantity, price, total price (of the line), discount, and total price of the invoice lines after applying the line discount. The invoice lines without a discount (or with zero discount) should not be included in the results. Recall that a null value in the line discount means no discount (or zero discount). Sort the results by invoice code descending, and then by number line ascending.

You can use the `TRUNC( value, precision )` function to truncate a number to the decimal places indicated in the precision argument.

⋆ **Exercise 11**: Show the code, description, and minimum stock of those articles whose prices are higher than 1 € and whose stocks or whose minimum stocks are unknown. Sort the results by article description.

6 What you don't have to forget

- **Be careful if you have to work with null values**: If a column allows nulls, treat this column carefully when you use it to set a restriction (`WHERE`).
- You must be able to determine *a priori* if you need to use the `DISTINCT` clause (without running the query).
- Some DBMSs implement the `DISTINCT` clause by using sorting algorithms, although nowadays it is not always the case. If the `DISTINCT` clause employs sorting algorithms, when a sentence using it must also sort the result, the `ORDER BY` clause can be obviously omitted (to save the high cost of the second sorting). To do that, you have to write the columns properly in the `SELECT` clause.

You can see if the DISTINCT clause is performing sorting by executing and comparing the next two queries:

```sql
SELECT DISTINCT COALESCE(iva,0) AS iva,
       COALESCE(dto,0) AS dto
FROM   facturas;
```

and by executing this one below:

```sql
SELECT DISTINCT COALESCE(dto,0) AS dto,
       COALESCE(iva,0) AS iva
FROM   facturas;
```

Are the data sorted according to some criterion?