Order by, Group by, Having e Distinct

Com os operadores **ORDER BY**, **GROUP BY**, **HAVING e DISTINCT** podemos organizar nossas consultas mais dinamicamente.

TABELA EXEMPLO

EX_ITENS

```
| CODIGO | DESCRICAO      | TIPO     | QUANTIDADE |
-------------------------------------------------
|      1 | 'Biscoito'  | 'Comida' |         20 |
|      2 | 'Água'         | 'Bebida' |      47,95 |
|      3 | 'Suco'         | 'Bebida' |       23,4 |
|      4 | 'Pão'          | 'Comida' |         10 |
|      5 | 'Refrigerante' | 'Bebida' |      13,75 |
```

SQL de CRIAÇÃO/INSERÇÃO:

```
CREATE TABLE ex_itens (
  codigo INTEGER,
  descricao VARCHAR2(200),
  tipo VARCHAR(20),
  quantidade DECIMAL(15,2));
INSERT INTO ex_itens VALUES (1, 'Biscoito', 'Comida', 20);
INSERT INTO ex_itens VALUES (2, 'Água', 'Bebida', 47.95);
INSERT INTO ex_itens VALUES (3, 'Suco', 'Bebida', 23.4);
INSERT INTO ex_itens VALUES (4, 'Pão', 'Comida', 10);
INSERT INTO ex_itens VALUES (5, 'Refrigerante', 'Bebida', 13.75);
COMMIT;
```

ORDER BY

Ordenar **crescente** pela coluna **TIPO**.

```
SELECT *
```

```
  FROM EX_ITENS
 ORDER BY TIPO
```

Retorno:

```
| CODIGO | DESCRICAO      | TIPO     | QUANTIDADE |
-------------------------------------------------
|      3 | 'Suco'         | 'Bebida' |       23,4 |
|      5 | 'Refrigerante' | 'Bebida' |      13,75 |
|      2 | 'Água'         | 'Bebida' |      47,95 |
|      4 | 'Pão'          | 'Comida' |         10 |
|      1 | 'Biscoito'     | 'Comida' |         20 |
```

Ordenar **crescente** pela coluna **TIPO** e **decrescente** pela coluna **QUANTIDADE**. (a ordenação respeita a sequência inserida)

```
SELECT *
  FROM EX_ITENS
 ORDER BY TIPO, QUANTIDADE DESC
```

Retorno:

```
| CODIGO | DESCRICAO      | TIPO     | QUANTIDADE |
-------------------------------------------------
|      2 | 'Água'         | 'Bebida' |      47,95 |
|      3 | 'Suco'         | 'Bebida' |       23,4 |
|      5 | 'Refrigerante' | 'Bebida' |      13,75 |
|      1 | 'Biscoito'     | 'Comida' |         20 |
|      4 | 'Pão'          | 'Comida' |         10 |
```

GROUP BY

Agrupar dados pela coluna **TIPO** contando a quantidade de **itens** (linhas) e **volumes** (coluna **QUANTIDADE**) para cada tipo de item.

```
SELECT TIPO,
       COUNT(1) AS QTD_ITEM,
       SUM(QUANTIDADE) AS QTD_VOLUME
  FROM EX_ITENS GROUP BY TIPO
```

Retorno:

```
```

```
| TIPO     | QTD_ITEM |  QTD_VOLUME |
-------------------------------------
| 'Bebida' |        3 |        85,1 |
| 'Comida' |        2 |          30 |
```

HAVING

Agrupar pela coluna **TIPO** mostrando somente quando a **soma** da coluna **QUANTIDADE** é **menor do que 40**.

```
SELECT TIPO, SUM(QUANTIDADE) AS QTD_VOLUME
  FROM EX_ITENS
 GROUP BY TIPO HAVING SUM(QUANTIDADE) < 40
```

Retorno:

```
| TIPO     | QTD_VOLUME |
-------------------------
| 'Comida' |          30 |
```

Agrupar pela coluna **TIPO** mostrando somente quando a contagens de linhas é **maior ou igual à 3**.

```
SELECT TIPO, COUNT(1) AS QTD_ITEM
  FROM EX_ITENS
 GROUP BY TIPO HAVING COUNT(1) >= 3
```

Retorno:

```
| TIPO     | QTD_ITEM |
-----------------------
| 'Bebida' |        3 |
```

DISTINCT

É uma clausula SQL a qual pega os valores da tabelas e traz somente os valores "diferentes". Caso haja valores duplicados, com esta função o retorno é apenas um valor.

Sem o DISTINCT:

```
SELECT TIPO
  FROM EX_ITENS
```

```
WHERE TIPO LIKE '%Comida%'
```

Retorno:

```
| TIPO     |
------------
| 'Comida' |
| 'Comida' |
```

Com o DISTINCT:

```
SELECT DISTINCT TIPO
  FROM EX_ITENS
 WHERE TIPO LIKE '%Comida%'
```

Retorno: Ou seja, caso houver mais de um valor correspondente com o filtro, o distinct apenas traz um, sem duplicá-los na consulta.

```
| TIPO     |
------------
| 'Comida' |
```

Em questão de performance entre **DISTINCT** ou **GROUP BY** são iguais. Entretanto o uso do **GROUP BY** são mais usados para agrupar valores de funções analítica tais como **SUM, AVG, COUNT, ETC**...

Revisão #2
Criado 9 March 2022 14:41:39 por Nicolly Andrielly
Atualizado 10 March 2022 14:23:40 por Nicolly Andrielly