ORDER BY

por Rafael SP

ORDER BY: Ordena os resultados

O comando **ORDER BY** é utilizado para **ordenar os resultados** da consulta com base em uma ou mais colunas. Você pode ordenar em ordem crescente (ASC) ou decrescente (DESC).

Sintaxe básica:

```
SELECT coluna1, coluna2, ...
FROM tabela
ORDER BY coluna1 [ASC|DESC];
```

- **Exemplo**: Ordenando os clientes por nome em ordem alfabética crescente:

```
SELECT nome, email FROM clientes
ORDER BY nome ASC;
```

- **Ordenando em ordem decrescente**:

```
SELECT nome, idade FROM clientes
ORDER BY idade DESC;
```

- **Ordenando por múltiplas colunas**: Você também pode ordenar por mais de uma coluna. Por exemplo, primeiro pela cidade e depois pelo nome dos clientes.

```
SELECT nome, cidade, idade FROM clientes
ORDER BY cidade ASC, nome ASC;
```

Exercícios

1. Ordenação Crescente (Clientes - Nome A-Z)

Dado o código abaixo, ordene alfabeticamente os clientes.

```
CREATE TABLE clientes (
    id SERIAL PRIMARY KEY,
    nome VARCHAR(100),
    email VARCHAR(100),
    cidade VARCHAR(100)
);

INSERT INTO clientes (nome, email, cidade) VALUES
('Carlos Silva', 'carlos@email.com', 'São Paulo'),
('Ana Souza', 'ana@email.com', 'Rio de Janeiro'),
('Bruno Mendes', 'bruno@email.com', 'Belo Horizonte'),
('Mariana Lima', 'mariana@email.com', 'Curitiba');
```

2. Ordenação Decrescente (Produtos - Maior Preço Primeiro)

Dado o código abaixo, ordene de Z a A os clientes

```
CREATE TABLE produtos (
    id SERIAL PRIMARY KEY,
    nome VARCHAR(100),
    preco DECIMAL(10,2)
);

INSERT INTO produtos (nome, preco) VALUES
('Notebook', 3500.00),
('Smartphone', 2500.00),
('Tablet', 1500.00),
('Fone de Ouvido', 200.00);
```

3. Ordenação por Múltiplas Colunas (Pedidos - Data Recente Primeiro, Maior Valor Primeiro)

Dado o código abaixo, ordene os pedidos com maior valor e com a data mais recente.

```
CREATE TABLE pedidos (
    id SERIAL PRIMARY KEY,
    cliente_id INT,
    data_pedido DATE,
    valor_total DECIMAL(10,2),
    FOREIGN KEY (cliente_id) REFERENCES clientes(id)
);

INSERT INTO pedidos (cliente_id, data_pedido, valor_total) VALUES
(1, '2024-02-15', 500.00),
(2, '2024-02-18', 1500.00),
(3, '2024-02-18', 1200.00),
(4, '2024-02-10', 800.00);
```

4. Ordenar funcionários por cargo em ordem alfabética e, dentro do cargo, pelo salário do maior para o menor

```
CREATE TABLE funcionarios (
    id SERIAL PRIMARY KEY,
    nome VARCHAR(100),
    cargo VARCHAR(50),
    salario DECIMAL(10,2)
);

INSERT INTO funcionarios (nome, cargo, salario) VALUES
('João Pedro', 'Gerente', 7500.00),
('Larissa Martins', 'Desenvolvedor', 5500.00),
('Fernando Costa', 'Designer', 4800.00),
('Clara Oliveira', 'Desenvolvedor', 6000.00),
('Lucas Rocha', 'Designer', 5200.00),
('Ana Paula', 'Gerente', 8200.00);
```

5. Ordenar pedidos pelo valor total do menor para o maior, e em caso de empate, pela data mais recente primeiro

```
CREATE TABLE pedidos (
    id SERIAL PRIMARY KEY,
    cliente_id INT,
    data_pedido DATE,
    valor_total DECIMAL(10,2)
);

INSERT INTO pedidos (cliente_id, data_pedido, valor_total) VALUES
(1, '2024-02-15', 500.00),
(2, '2024-02-18', 1500.00),
(3, '2024-02-18', 1200.00),
(4, '2024-02-10', 500.00),
(5, '2024-02-20', 800.00);
```