FMX Academy

INFORMAÇÕES		
MÓDULO	**AULA**	**INSTRUTOR**
01 Fundamentos de Banco de Dados	**03 – Joins e Relacionamentos**	**Matheus Laureto**

CONTEÚDO

1. Revisão da Aula Anterior

WHERE – Filtragem

- Usado para **limitar os resultados.**
- Operadores:
  - = (igual), <> ou != (diferente)
  - >, <, >=, <=
  - LIKE (padrão de texto)
  - BETWEEN (faixa)
  - IN (conjunto)
  - IS NULL / IS NOT NULL
- Pode ser combinado com **NOT** para negação.

ORDER BY – Ordenação

- Ordena resultados em **ASC** (padrão) ou **DESC** (decrescente).
- ASC é implícito, DESC precisa ser escrito.

Funções de Agregação

- **COUNT()** → Conta registros.
- **SUM()** → Soma valores.
- **AVG()** → Média.
- **MIN()** → Menor valor.
- **MAX()** → Maior valor.

Agrupamento

- **GROUP BY** → Agrupa linhas com valores iguais para aplicar funções agregadas.
- **HAVING** → Filtra resultados **após** o agrupamento (diferente do WHERE, que filtra antes).

2. JOINS

O que são JOINS:

- São comandos utilizados para combinar registros de duas ou mais tabelas, com base em uma condição de comparação;
- Normalmente esses relacionamentos são feitos através de campos chave;
- Entender bem os tipos de JOINs e suas funcionalidades é crucial para uma boa implementação de código no banco.

Tipos de Joins:

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- FULL JOIN
- CROSS JOIN
- SELF JOIN

- **INNER JOIN**
  É o comando "básico", usado para buscar apenas os registros que têm correspondência em todas as tabelas envolvidas na consulta.
  Caso não tenha correspondência, o registro não será apresentado.

  Sintaxe:

```sql
SELECT nome_colunas
FROM tabela1
INNER JOIN tabela2
    ON tabela1.coluna_em_comum = tabela2.coluna_em_comum;
```

  Exemplo:

```sql
SELECT clientes.nome, pedidos.id, pedidos.valor
  FROM clientes
 INNER JOIN pedidos
    ON clientes.id = pedidos.id;
```

- **LEFT JOIN**
  A partir desse comando, começamos a filtrar informações a serem apresentadas no resultado.
  O **LEFT JOIN** retorna todos os registros da tabela da esquerda (tabela que vem antes do **JOIN**) e os registros correspondentes da tabela da direita.
  O que não tiver correspondência, vem com NULL nas colunas da direita.
  Ou seja: o **LEFT JOIN** busca todos os registros da tabela à esquerda (**LEFT**) e os registros que sejam correspondentes à direita.

  Sintaxe:

```sql
SELECT nome_colunas
  FROM tabela1
 LEFT JOIN tabela2
    ON tabela1.coluna_em_comum = tabela2.coluna_em_comum;
```

  Exemplo:

```sql
SELECT clientes.nome, pedidos.id, pedidos.valor
  FROM clientes
  LEFT JOIN pedidos
    ON clientes.id = pedidos.id;
```

- **RIGHT JOIN**
  Basicamente tem a mesma funcionalidade do **LEFT JOIN**, mas dessa vez a tabela que trás todos os dados é a tabela da direita (**RIGHT**).
  Os registros que não tiverem correspondência na tabela da esquerda, a coluna aparecerá com **NULL.**

  Sintaxe:

```sql
SELECT nome_colunas
  FROM tabela1
 RIGHT JOIN tabela2
    ON tabela1.coluna_em_comum = tabela2.coluna_em_comum;
```

  Exemplo:

```sql
SELECT clientes.nome, pedidos.id, pedidos.valor
  FROM clientes
  RIGHT JOIN pedidos
    ON clientes.id = pedidos.id;
```

- **FULL JOIN**
  Esse comando funciona para retornar todos os registros das duas tabelas, combinando os registros correspondentes e, quando não haver correspondência, é preenchido com NULL.

  Sintaxe:

```
SELECT nome_colunas
  FROM tabela1
 FULL JOIN tabela2
    ON tabela1.coluna_em_comum = tabela2.coluna_em_comum;
```

  Exemplo:

```
SELECT clientes.nome, pedidos.id, pedidos.valor
  FROM clientes
  FULL JOIN pedidos
    ON clientes.id = pedidos.id;
```

- **CROSS JOIN (os DBAs não gostam desse comando)**
  Esse comando é simples, mas perigoso na sua utilização.
  O **CROSS JOIN** gera um produto cartesiano, ou seja, todas as combinações possíveis entre as linhas da tabela da esquerda e da tabela da direita.
  Ex: se na tabela CLIENTES tem 10 registros e na tabela PEDIDOS tem 10 registros, o resultado será 10 x 10 = 100.
  No **CROSS JOIN** não se faz utilização do comando ON, já que o cruzamento é feito em todas as linhas.

  Sintaxe:

```
SELECT nome_colunas
  FROM tabela1
 CROSS JOIN tabela2
```

  Exemplo:

```
SELECT clientes.nome, pedidos.id, pedidos.valor
  FROM clientes
  CROSS JOIN pedidos
```

- **SELF JOIN**
  Esse comando é utilizado para fazer um JOIN de uma tabela com ela mesma.
  Isso pode ser útil quando precisa comparar o relacionar linhas da mesma tabela, geralmente usando apelidos diferentes para distingui-las.

  Sintaxe:

```sql
SELECT a.coluna, b.coluna
FROM tabela a
JOIN tabela b
    ON condição_de_relacionamento;
```

  Exemplo:

```sql
SELECT f.nome AS funcionario, g.nome AS gerente
  FROM funcionarios f
  LEFT JOIN funcionarios g
    ON f.id_gerente = g.id_funcionario;
```

3. EXERCICIOS

1. Liste o nome do cliente e o status de cada pedido realizado.
   Use as tabelas CLIENTES e PEDIDO
   Mostre apenas os clientes que realmente tenham feito pedidos.

2. Liste todos os clientes e, se houver, mostre também o valor total de seus pedidos.
   Clientes sem pedidos devem aparecer como null no valor.

3. Liste todos os pedidos, mostrando também o nome do cliente (se houver).
   Pedidos sem clientes devem aparecer com null no nome.

4. Liste todos os clientes e pedidos, combinando onde houver correspondência, mas garantindo que todos os registros apareçam.
   Mostre nome do cliente, número do pedido e valor.

5. Liste todos os funcionários e seus respectivos gerentes.
   Funcionários sem gerente devem aparecer com null no campo do gerente.