FMX Academy

INFORMAÇÕES		
MÓDULO	**AULA**	**INSTRUTOR**
01 Fundamentos de Banco de Dados	**02 – Filtragem, Ordenação e Funções Básicas**	**Matheus Laureto**

CONTEÚDO

1. Revisão da Aula Anterior

SQL (Structured Query Language) é a linguagem padrão para trabalhar com bancos de dados relacionais.

Serve para:

- Consultar
- Inserir
- Atualizar
- Apagar Dados
- Criar e gerenciar estruturas do banco

Categorias de Comandos SQL

- DML – Data Manipulation Language (select, insert, update, delete)
- DDL – Data Definition Language (create, alter, drop, truncate)
- DCL – Data Control Language (grant, revoke)

Resumo dos principais comandos:

- **SELECT + FROM + WHERE**
  Consulta dados de uma (ou mais) tabelas
  Comando **WHERE** filtra os resultados, não é obrigatório, mas altamente recomendado.

- **INSERT**
  Comando para inserir registros em uma tabela já criada.

- **UPDATE**
  Altera valores de registros existes (cuidado com **UPDATE** sem **WHERE**!)

- **DELETE**
  Remove registros da tabela. Se for utilizado sem **WHERE**, pode limpar a tabela inteira.

- **CREATE**
  Comando para criar estruturas na base (tabelas, views, procedures, etc...)

- **ALTER**
  Modifica as estruturas existentes.

- **DROP**
  Elimina estruturas e seus dados (no caso das tabelas)

- **TRUNCATE**
  Apaga todos os dados de uma tabela, mantendo a estrutura (commit implícito)

2. WHERE – Filtragem de Dados

Sintaxe:

```
select coluna1, coluna2...
  from tabela
 where condicao
```

Exemplo:

```
select * from tabela_modelo where codigo = 1
select * from tabela_modelo where descricao = 'MODELO 1'
```

Operadores para Filtragem:

- **Igualdade – operador `=`**
  Ex: WHERE nome = 'João'

- **Diferente – operador `<>` ou `!=`**
  Ex: WHERE cidade <> 'SP'

- **Maior/Menor – operadores `>`, `<`, `>=`, `<=`**
  Ex: WHERE idade < 30

- **Texto – operador `LIKE`**
  Ex: WHERE empresa LIKE 'ITB%'

- **Faixa – operador `BETWEEN`**
  Ex: WHERE idade BETWEEN 20 AND 30

- **Conjunto – operador IN**
  Ex: WHERE cidade IN ('SP', 'RJ')

- **Nulos – operador IS NULL**
  Ex: WHERE data IS NULL

Obs: Para a maior parte dos operadores, também existe a operação de negação, utilizando o comando NOT
Ex: **WHERE data IS NOT NULL**

3. ORDER BY – Ordenação de Resultados

Sintaxe:

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

Exemplo:

```
select codigo, descricao from tabela_modelo order by codigo desc
select codigo, descricao from tabela_modelo order by descricao asc
```

ORDER BY é comando utilizado para ordenar os resultados das consultas, podendo ser de maneira crescente (**ASC**) ou decrescente (**DESC**).

Obs: por padrão, o **ORDER BY** sempre será executado de maneira crescente, nao tendo a necessidade de expressar o comando **ASC**, pois o comando implicitamente roda como **ORDER BY ASC.**
O contrario já não acontece, ou seja, sempre que deseja uma ordenação decrescente, é obrigatório a utilização do comando **ORDER BY DESC.**

4. FUNÇÕES DE AGREGAÇÃO

As funções de agregação são utilizadas para resumir e/ou analisar grandes volumes de dados, retornando um valor representativo.

- **COUNT()** - Utilizada para contar registros

```
SELECT COUNT(*) FROM tabela;
SELECT COUNT(coluna) FROM tabela;
SELECT COUNT(1) FROM tabela;
```

- **SUM()** – Soma valores numéricos

```
SELECT SUM(valor) FROM vendas;
```

- **AVG()** – Calcula o valor médio

```sql
SELECT AVG(salario) FROM funcionarios;
```

- **MIN()** – Retorna o menor valor

```sql
SELECT MIN(idade) FROM clientes;
```

- **MAX()** – Retorna o maior valor

```sql
SELECT MAX(preco) FROM produtos;
```

5. FUNÇÕES DE AGRUPAMENTO

Funções de agrupamento são comandos que servem para agrupar registros, e aplicar as funções de agregação para cada grupo.

- **GROUP BY**
  Utilizado para agrupar registros com o mesmo valor de uma ou mais colunas, e aplica a função de agregação ao grupo (no exemplo, conta quantos cliente existem em cada cidade):

```sql
SELECT cidade, COUNT(*) AS total_clientes
FROM clientes
GROUP BY cidade;
```

- **HAVING**
  O comando HAVING funciona como o comando WHERE, mas para as funções agrupadas/agregadas.
  No exemplo abaixo, vemos uma consulta, verificando as cidades de forma agrupada, com mais de 10 clientes:

```sql
SELECT cidade, COUNT(1) AS total
FROM clientes
GROUP BY cidade
HAVING COUNT(1) > 10;
```

6. EXERCICIOS

1. Liste da tabela clientes, os registros que:
   Tenha idade superior a 30 anos
   Moram no estado de São Paulo (SP)
   O nome da pessoa comece com a letra J

2. Liste da tabela de produtos:
   Todos os produtos com a categoria “Eletrônicos”
   Ordene em ordem alfabética

3. Na tabela de vendas, calcule:
   O valor total de vendas
   A média de valor por venda
   A quantidade de vendas realizadas

4. Na tabela de clientes:
   Mostre a quantidade de clientes por cidade

5. Na tabela de pedidos, mostre:
   Clientes que fizeram mais de dois pedidos.