Aula: GROUP BY, COUNT, SUM, AVG, MAX, MIN, HAVING e DISTINCT em MySQL

1. Estrutura das Tabelas

Vamos continuar usando as tabelas **Produtos** e **Vendas** como base.

Tabela `Produtos`

Essa tabela armazena as informações dos produtos disponíveis.

id_produto	nome_produto	preco
1	Notebook	3000
2	Teclado	150
3	Mouse	50
4	Monitor	700

Tabela `Vendas`

Essa tabela registra as vendas realizadas, indicando o produto vendido, a quantidade e a data da venda.

id_venda	id_produto	data_venda	quantidade
101	1	2023-08-01	3
102	2	2023-08-02	5
103	1	2023-08-03	2
104	3	2023-08-05	10
105	4	2023-08-05	1

---

2. Comandos de Criação das Tabelas

Para criar as tabelas no MySQL, use os seguintes comandos:

sql
Copiar código
```
CREATE TABLE Produtos (
    id_produto INT PRIMARY KEY,
```

```
    nome_produto VARCHAR(50),
    preco DECIMAL(10, 2)
);

CREATE TABLE Vendas (
    id_venda INT PRIMARY KEY,
    id_produto INT,
    data_venda DATE,
    quantidade INT,
    FOREIGN KEY (id_produto) REFERENCES Produtos(id_produto)
);
```

---

3. Comandos de Inserção de Dados

Agora vamos inserir os dados nas tabelas para realizar as consultas posteriormente.

sql
Copiar código

```
INSERT INTO Produtos (id_produto, nome_produto, preco)
VALUES
(1, 'Notebook', 3000),
(2, 'Teclado', 150),
(3, 'Mouse', 50),
(4, 'Monitor', 700);

INSERT INTO Vendas (id_venda, id_produto, data_venda, quantidade)
VALUES
(101, 1, '2023-08-01', 3),
(102, 2, '2023-08-02', 5),
(103, 1, '2023-08-03', 2),
(104, 3, '2023-08-05', 10),
(105, 4, '2023-08-05', 1);
```

---

4. O Comando `DISTINCT`

O **DISTINCT** é utilizado para eliminar duplicatas de um conjunto de resultados. Ele retorna apenas valores únicos de uma coluna ou combinação de colunas.

Exemplo: Produtos Vendidos (Sem Duplicatas)

Queremos listar os diferentes produtos que já foram vendidos, sem repetições.

sql
Copiar código
```sql
SELECT DISTINCT id_produto
FROM Vendas;
```

Resultado:

id_produto
1
2
3
4

Esse comando retorna uma lista única de produtos vendidos, eliminando registros repetidos.

Exemplo: Produtos e suas Datas de Venda (Sem Duplicatas)

Agora queremos ver todos os pares únicos de produtos e datas de venda.

sql
Copiar código
```sql
SELECT DISTINCT id_produto, data_venda
FROM Vendas;
```

Resultado:

id_produto	data_venda
1	2023-08-01
2	2023-08-02
1	2023-08-03
3	2023-08-05
4	2023-08-05

---

5. Diferença Entre `DISTINCT` e `GROUP BY`

Embora ambos **DISTINCT** e **GROUP BY** sejam usados para evitar duplicatas, eles têm diferenças importantes em seus usos:

- **DISTINCT** é usado principalmente para remover duplicatas de uma consulta sem aplicar agregações. Ele é utilizado quando você deseja ver apenas os valores únicos de uma ou mais colunas.
- **GROUP BY** é utilizado quando você deseja agrupar linhas para aplicar funções agregadas (como **SUM**, **COUNT**, **AVG**, etc.).

Exemplo de Comparação

Suponha que queremos contar o número de produtos únicos vendidos.

- Usando **DISTINCT**:

sql
Copiar código
```
SELECT COUNT(DISTINCT id_produto) AS num_produtos_unicos
FROM Vendas;
```

- Usando **GROUP BY**:

sql
Copiar código
```
SELECT id_produto, COUNT(*) AS num_vendas
FROM Vendas
GROUP BY id_produto;
```

DISTINCT apenas retorna o número de produtos distintos, enquanto **GROUP BY** agrupa as linhas por produto e calcula quantas vezes cada produto foi vendido.

---

6. Funções de Agregação com `GROUP BY`

Conforme vimos anteriormente, o **GROUP BY** agrupa as linhas com base em uma ou mais colunas e permite aplicar funções de agregação como **COUNT**, **SUM**, **AVG**, **MAX**, e **MIN**. Vamos revisar alguns exemplos.

Exemplo: Quantidade Total Vendida por Produto

sql
Copiar código
```
SELECT id_produto, SUM(quantidade) AS total_vendido
FROM Vendas
GROUP BY id_produto;
```

Resultado:

id_produto	total_vendido
1	5
2	5
3	10
4	1

---

7. Funções de Agregação com `HAVING`

A cláusula **HAVING** é usada para filtrar os resultados após a agregação, enquanto a cláusula **WHERE** é usada antes da agregação.

Exemplo: Filtrar Produtos com Total de Vendas Maior que 4

sql
Copiar código

```
SELECT id_produto, SUM(quantidade) AS total_vendido
FROM Vendas
GROUP BY id_produto
HAVING SUM(quantidade) > 4;
```

Resultado:

id_produto	total_vendido
1	5
2	5
3	10

---

8. Exemplo Completo com `DISTINCT`, `GROUP BY` e Funções Agregadas

Vamos fazer um exemplo mais complexo que combine vários conceitos.

Exemplo: Relatório Completo de Vendas por Produto

Queremos um relatório que mostre:

- O número de vendas por produto.
- O total de unidades vendidas.
- A média de quantidade vendida por pedido.

- A maior e menor quantidade vendida em um pedido. E, em seguida, mostrar apenas produtos que venderam mais de 3 unidades no total.

sql
Copiar código

```
SELECT id_produto,
       COUNT(DISTINCT id_venda) AS num_vendas,
       SUM(quantidade) AS total_vendido,
       AVG(quantidade) AS media_vendida,
       MAX(quantidade) AS max_venda,
       MIN(quantidade) AS min_venda
FROM Vendas
GROUP BY id_produto
HAVING SUM(quantidade) > 3;
```

Resultado:

id_produto	num_vendas	total_vendido	media_vendida	max_venda	min_venda
1	2	5	2.5	3	2
2	1	5	5.0	5	5
3	1	10	10.0	10	10

---

9. Conclusão

Nesta aula, abordamos:

- O uso de **GROUP BY** para agrupar dados.
- O uso do **DISTINCT** para eliminar duplicatas e sua diferença com **GROUP BY**.
- Funções de agregação como **COUNT**, **SUM**, **AVG**, **MAX**, **MIN** para realizar cálculos sobre grupos de dados.
- A utilização da cláusula **HAVING** para filtrar grupos com base em condições agregadas.

Esses conceitos são essenciais para gerar relatórios e análises em bancos de dados relacionais.