**1. Buscar a quantidade de médicos por especialidade, ordenados por especialidade**

**SELECT especialidade, COUNT(\*) AS quantidade\_medicos**

**FROM Medicos**

**GROUP BY especialidade**

**ORDER BY especialidade;**

**Explicação:**

* SELECT especialidade, COUNT(\*) AS quantidade\_medicos: Seleciona a coluna especialidade e conta quantos médicos existem para cada especialidade. O resultado será exibido na coluna quantidade\_medicos.
* FROM Medicos: A tabela de origem é Medicos.
* GROUP BY especialidade: Agrupa os registros pela coluna especialidade.
* ORDER BY especialidade: Ordena o resultado por especialidade em ordem alfabética.

**2. No exercício 2, buscar apenas aquelas especialidades com mais de 1 médico atendente**

SELECT especialidade, COUNT(\*) AS quantidade\_medicos

FROM Medicos

GROUP BY especialidade

HAVING COUNT(\*) > 1

ORDER BY especialidade;

**Explicação:**

* **Diferença do exercício anterior**: Adicionamos a cláusula HAVING para filtrar os grupos onde a quantidade de médicos (COUNT(\*)) é maior que 1.
* HAVING: Diferente do WHERE, o HAVING é usado para filtrar resultados de agregações (como a contagem aqui).

**3. Buscar a maior idade de médicos por especialidade**

SELECT especialidade, MAX(idade) AS maior\_idade

FROM Medicos

GROUP BY especialidade

ORDER BY especialidade;

**Explicação:**

* SELECT especialidade, MAX(idade) AS maior\_idade: Seleciona a especialidade e calcula a maior idade de médicos em cada especialidade.
* MAX(idade): Função de agregação que retorna o maior valor da coluna idade.
* GROUP BY especialidade: Agrupa os médicos por especialidade.
* ORDER BY especialidade: Ordena o resultado por especialidade.

**4. Buscar média de salário dos funcionários por cidade**

SELECT cidade, AVG(salario) AS media\_salario

FROM Funcionarios

GROUP BY cidade

ORDER BY cidade;

**Explicação:**

* SELECT cidade, AVG(salario) AS media\_salario: Seleciona a cidade e calcula a média de salário dos funcionários em cada cidade.
* AVG(salario): Função de agregação que retorna a média dos valores da coluna salario.
* GROUP BY cidade: Agrupa os registros por cidade.
* ORDER BY cidade: Ordena o resultado por cidade em ordem alfabética.

**5. Apresentar a quantidade de consultas por médicos, listando o código do médico e o total de consultas agendadas**

SELECT codm, COUNT(\*) AS total\_consultas

FROM Consultas

GROUP BY codm

ORDER BY total\_consultas DESC;

**Explicação:**

* SELECT codm, COUNT(\*) AS total\_consultas: Seleciona o código do médico (codm) e conta o número total de consultas feitas por cada médico.
* GROUP BY codm: Agrupa as consultas pelo código do médico.
* ORDER BY total\_consultas DESC: Ordena os resultados em ordem decrescente de total de consultas.

**6. Apresentar os andares que têm capacidade total de leitos superior a 100**

SELECT andar, SUM(capacidade) AS capacidade\_total

FROM Ambulatorio

GROUP BY andar

HAVING SUM(capacidade) > 100

ORDER BY andar;

**Explicação:**

* SELECT andar, SUM(capacidade) AS capacidade\_total: Seleciona o andar e calcula a soma da capacidade de todos os ambulatórios naquele andar.
* SUM(capacidade): Função de agregação que soma os valores da coluna capacidade.
* GROUP BY andar: Agrupa os registros por andar.
* HAVING SUM(capacidade) > 100: Filtra os grupos para mostrar apenas aqueles cuja soma da capacidade é maior que 100.
* ORDER BY andar: Ordena o resultado por número do andar.

**Resumo**

Esses scripts utilizam **funções de agregação** (como COUNT, MAX, AVG, SUM) para consolidar informações. O uso de **GROUP BY** é essencial para agrupar os dados antes de aplicar essas funções, e o **HAVING** permite filtrar os resultados das agregações.

Se você precisar de mais detalhes ou exemplos, estou aqui para ajudar! 😊

4o

**Você disse:**

GROUP BY andar: Agrupa os registros por andar. esse script agrupa todos dos andares ou só os com capacidade > 100

**O ChatGPT disse:**

ChatGPT

O script **GROUP BY andar** agrupa **todos os registros por andar**, independentemente do valor da soma da capacidade.

A filtragem para mostrar apenas os andares com capacidade total superior a 100 é feita posteriormente pela cláusula **HAVING SUM(capacidade) > 100**.

**Fluxo do script:**

1. **GROUP BY andar**: Agrupa todos os registros da tabela Ambulatorio por andar. Cada grupo contém todos os ambulatórios que pertencem ao mesmo andar.
2. **SUM(capacidade)**: Calcula a soma da capacidade de cada grupo (ou seja, de cada andar).
3. **HAVING SUM(capacidade) > 100**: Filtra os grupos criados no passo anterior, mantendo apenas aqueles cuja soma da capacidade é maior que 100.

**Exemplo com dados fictícios:**

| **nroa** | **andar** | **capacidade** |
| --- | --- | --- |
| 1 | 1 | 30 |
| 2 | 1 | 40 |
| 3 | 2 | 60 |
| 4 | 2 | 50 |
| 5 | 3 | 90 |

* **Agrupamento (GROUP BY andar)**:
  + Andar 1: Capacidade total = 30 + 40 = **70**
  + Andar 2: Capacidade total = 60 + 50 = **110**
  + Andar 3: Capacidade total = **90**
* **Filtragem (HAVING SUM(capacidade) > 100)**:
  + Apenas o andar 2 será retornado porque sua capacidade total é maior que 100.

Se o script **não** tivesse a cláusula HAVING, todos os andares seriam exibidos com suas somas respectivas.