

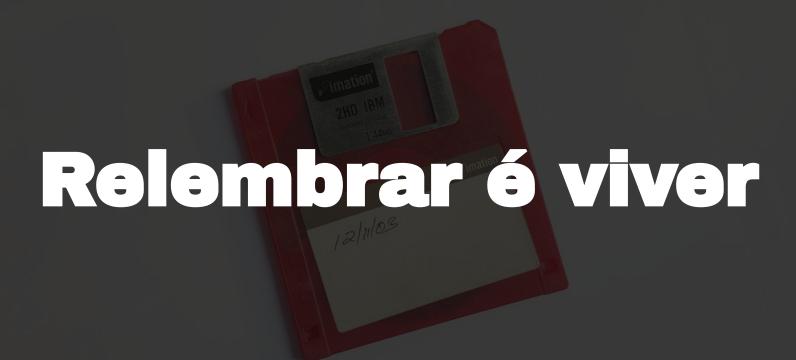
## Mais Informações

Módulo 4 - Aula 8 - Hard









#### Relembrando

- Create
- Read
- Update
- Delete

- NOT
- o AND
- o OR
- o IN
- o Max, Min
- DISTINCT
- ORDER BY

## Saki asilin

#### Aquecendo os motores

#### **Perguntas:**

- 1. Quais são os pagamentos mais caros?
- 2. Quais são os clientes inativos?
- 3. Quais são as cidades cadastradas daqui do Brasil?
- 4. Quais são as cidades cadastradas que fazem parte da América Latina?
- 5. Quais são os aluguéis feitos pelo funcionário 'Mike Hillyer'?
- 6. Quais são os filmes em que o ator 'John Suvari' participou?
- 7. Quais são os filmes em que um ator chamado 'Rip' participou?

# Agregação

#### Agregação

Em alguns cenários queremos contar, somar, agrupar os resultados.

Para realizar isso, precisaremos de mais de uma coluna e utilizar o comando group by

## Agregação - soma

```
customer_id,
sum(amount)
FROM PAYMENT
group by
customer_id
;
```

#### Agregação - contagem

```
customer_id,
count(customer_id)

FROM PAYMENT
group by
customer_id
;
```

#### **Atividade: Aprofundando!**

- → Vocês vão ficar surpresos em saber que temos mais perguntas para responder usando o nosso banco Sakila. Agora, vocês deverão criar as consultas que respondem às seguintes perguntas:
  - Qual o ranking dos DVDs que foram mais alugados?
  - Quantas vezes o filme com id = 1 na loja 1 foi alugado?
  - Qual funcionário alugou mais filmes?
  - Qual o faturamento do dia 27 de maio de 2005?
  - Quanto cada funcionário faturou no dia 27 de maio de 2005?"



#### Filtros em agregações - problema

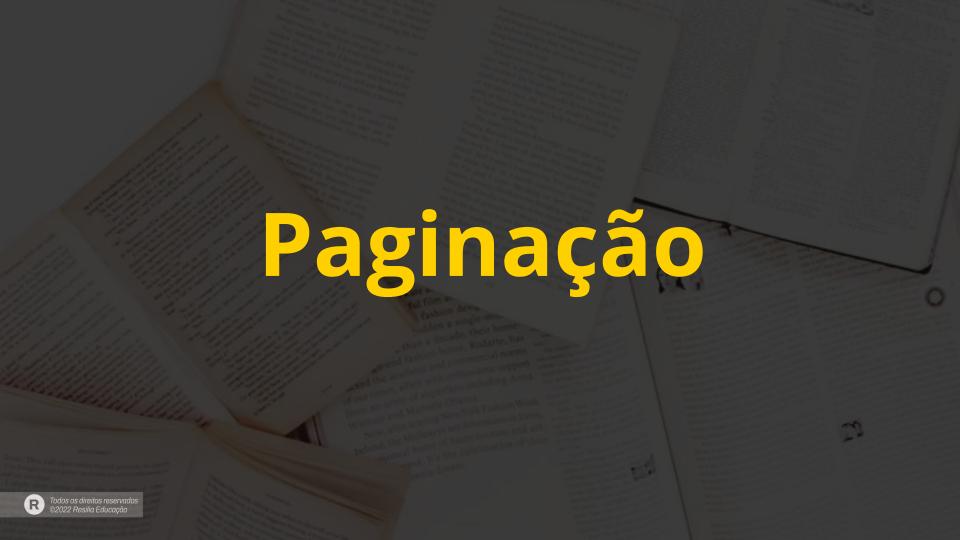
Como o WHERE acontece antes do SELECT não podemos utilizá-lo!

#### Having

A filtragem utilizando HAVING resolve este problema pois ele será executado somente após o SELECT.

```
SELECT
    customer id,
    sum(amount)
FROM
    PAYMENT
group by customer id
HAVING
    sum(amount) > 100
```

```
SELECT
    customer id,
    count(customer id)
FROM
    RENTAL
group by customer id
HAVING
    count(customer id)
```





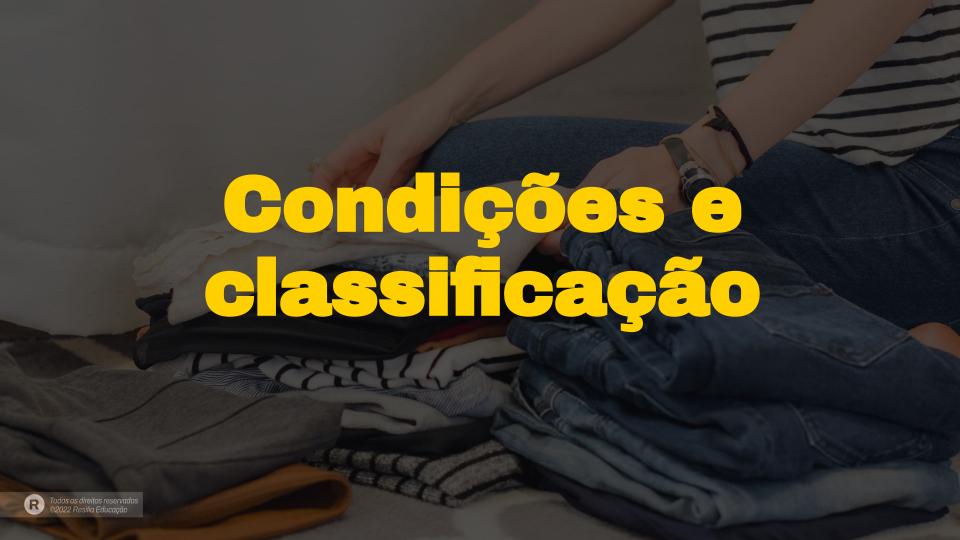
#### Limitando resultados

Para limitar a quantidade de linhas retornadas numa consulta utilizamos a palavra limit,

como no exemplo abaixo:



Nós podemos especificar também um offset, para dizer a partir de qual registro a contagem do limite começa.



#### **Condições e classificações**

Para verificar condições em SQL utilizamos a expressão CASE.

#### Condições e classificações - sintaxe

A expressão CASE é dividida nas seguintes etapas:

- 1. CASE: Inicia a verificação de condições
- 2. WHEN: Recebe a condição a ser verificada
- 3. THEN: Retorno exibido caso a condição seja verdadeira
- 4. ELSE: Resultado padrão/default
- 5. END: Finaliza a expressão de verificação

#### Condições e classificações - sintaxe

#### Exemplo prático:

```
SELECT
    customer_id,
    CASE
        WHEN
            count(customer id) > 30
        THEN
            'SUPER CLIENTE'
        WHEN
            count(customer_id) > 15
        THEN
             'CLIENTE FIEL'
        WHEN
            count(customer_id) > 0
        THEN
             'CLIENTE EVENTUAL'
    END as tipo_cliente
```



#### **Atividade: Classificados!**

- 1) Classifique os clientes por faixa de gastos:
- Acima de 100\$ 'Clientão brabo'
- ⇒ Entre 70\$ e 100\$ 'Cliente bom'
- Abaixo de 70\$ 'Cliente eventual'
- 2) Classifique os atores por número de participações:
- Acima de 40 participações 'Topa tudo'
- Acima de 30 participações 'Altamente produtivo'
- Acima de 20 participações 'Fez alguns filmes'
- ⇒ Acima de 10 participações 'Tem que fazer seu nome ainda'



### Atividade em pares