



escola
britânica de
artes criativas
& tecnologia

Desenvolvedor Full Stack Python

Agregação de Dados

AGENDA

- **Agrupando dados utilizando o Group By e Having**
- **Order By e Distinct**
- **Limitando a busca de dados no banco de dados utilizando offset e limit**

Agrupando dados utilizando o Group By e Having

A cláusula **GROUP BY** agrupa linhas baseado em semelhanças entre elas.

O **GROUP BY** é útil quando queremos agrupar e calcular informações dentro da nossa tabela:

payment
* payment_id
customer_id
staff_id
rental_id
amount
payment_date

```
SELECT  
    customer_id  
FROM  
    payment  
GROUP BY  
    customer_id;
```



	customer_id	smallint
1		184
2		87
3		477
4		273
5		550
6		51
7		394
8		272
9		70

Agrupando dados utilizando o Group By e Having

No exemplo anterior **GROUP BY** removeu os dados duplicados da nossa tabela.

Podemos utilizar o **GROUP BY** para agregar dados, neste caso podemos utilizar o **SUM()**.

Podemos calcular o valor total de cada pedido utilizando o **Group By** juntamente com a função **SUM**.

Agrupando dados utilizando o Group By e Having

```
SELECT  
    customer_id,  
    SUM (amount)  
FROM  
    payment  
GROUP BY  
    customer_id;
```

	customer_id smallint	sum numeric
1	184	80.80
2	87	137.72
3	477	106.79
4	273	130.72
5	550	151.69
6	51	123.70
7	394	77.80
8	272	65.87
9	70	75.83
10	190	102.75
11	350	63.79

Group By com JOIN

```
SELECT
    first_name || ' ' || last_name full_name,
    SUM (amount) amount
FROM
    payment
INNER JOIN customer USING (customer_id)
GROUP BY
    full_name
ORDER BY amount;
```

	full_name text	amount numeric
1	Eleanor Hunt	211.55
2	Karl Seal	208.58
3	Marion Snyder	194.61
4	Rhonda Kennedy	191.62
5	Clara Shaw	189.60
6	Tommy Collazo	183.63
7	Ana Bradley	167.67
8	Curtis Irby	167.62
9	Marcia Dean	166.61
10	Mike Way	162.67
11	Arnold Havens	161.68
12	Wesley Bull	158.65

Having

Podemos utilizar o Having para filtrar grupos ou uma agregação.

Diferente do **Where** onde utilizamos para filtrar dados individuais.

Group By e Having

```
SELECT
    customer_id,
    SUM (amount)
FROM
    payment
GROUP BY
    customer_id
HAVING
    SUM (amount) > 200;
```

	customer_id smallint	sum numeric
1	526	208.58
2	148	211.55

Removendo Dados Repetidos com Distinct

Para remover dados duplicados em SQL, utilizamos comando **DISTINCT** seguido das colunas que queremos remover dados duplicados:

```
SELECT DISTINCT customer_id FROM customer;
```


Ordenando Campos com Order By

Quando você consulta dados de uma tabela, a instrução SELECT retorna linhas em uma ordem não especificada. Para classificar as linhas do conjunto de resultados, você usa a cláusula **ORDER BY** na instrução SELECT.

```
SELECT * FROM customer ORDER BY name ASC ou DESC;
```

DESC = Decrescente

ASC = Crescente

Ordenando Campos com Order By

SELECT * FROM customer **ORDER BY** name **ASC** ou **DESC**;

DESC = Decrescente

ASC = Crescente

OBS: **ASC** é a opção default caso não especificemos o tipo de ordenação

Offset e Limit no Postgres

A cláusula LIMIT do PostgreSQL é usada para limitar a quantidade de dados retornada pela instrução SELECT.

```
SELECT * FROM customer LIMIT 4;
```

Para selecionar ou retornar dados de uma determinada posição, usamos o LIMIT junto com o OFFSET

```
SELECT * FROM customer LIMIT 4 OFFSET 2;
```

Resumo

Aprendemos um pouco como **agregações** funcionam, como ordenar e filtrar dados repetidos, além de aprender um pouco sobre o offset e limit para limitar informações no SQL.

Agora que você já sabe como agregações funcionam neste exercício vamos construir uma agregação para saber **a quantidade de itens que cada produto tem.**

- Utilize a função de Group By e Sum, Juntamente com o Join nas tabelas de Product e Stock

Você pode compartilhar os arquivos .SQL de agregação na plataforma da EBAC.