Banco de dados II

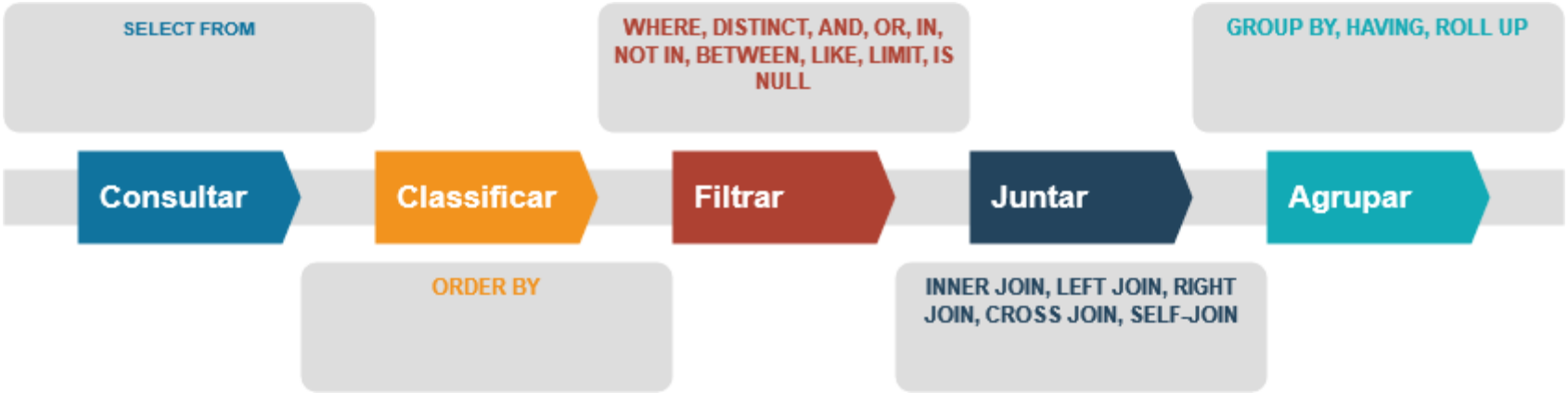
IFSP - Guarulhos

Objetivos

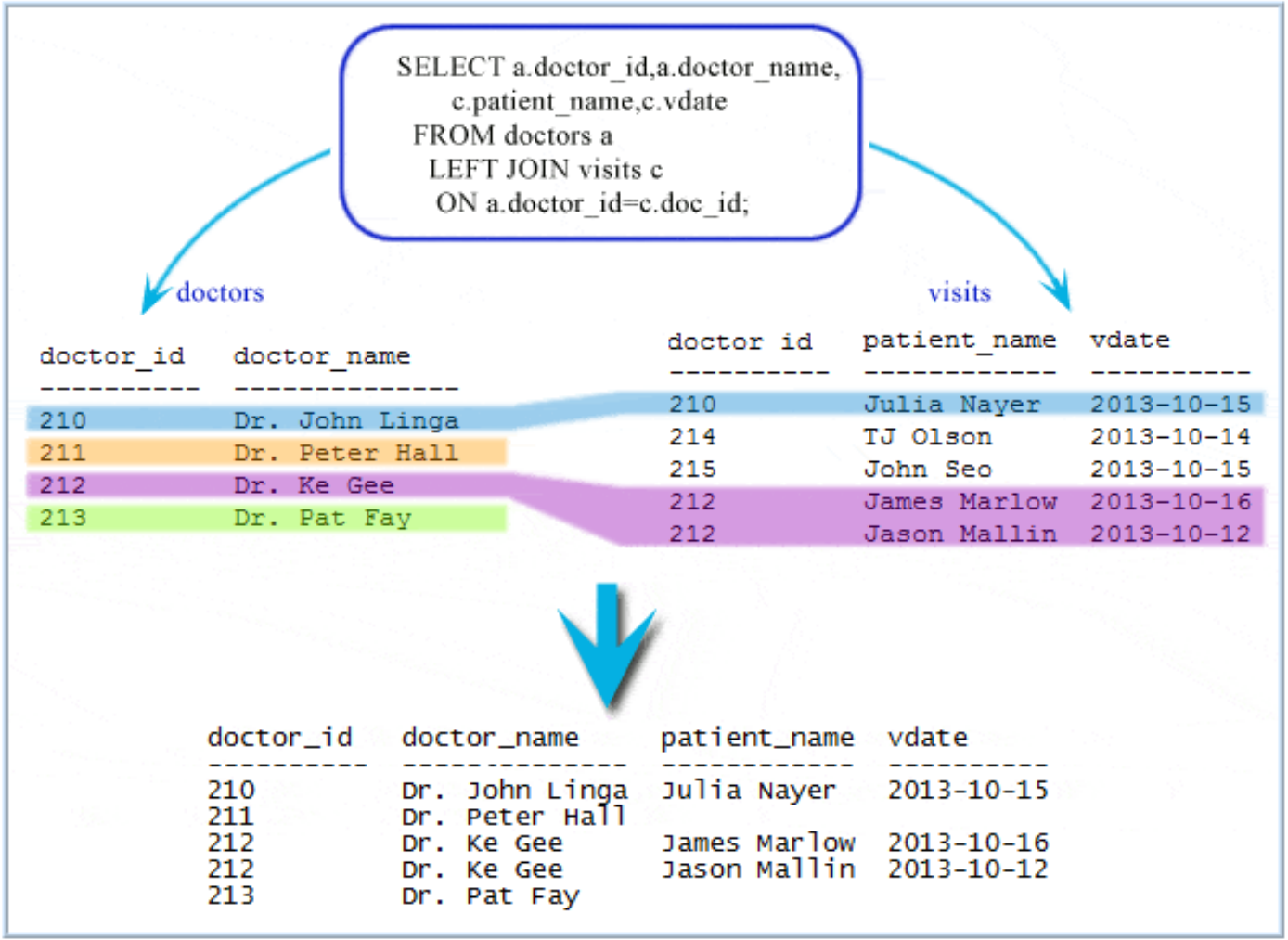
Estudar agrupamento de dados com

- GROUP BY
- HAVING
- ROLL UP

Revisando...

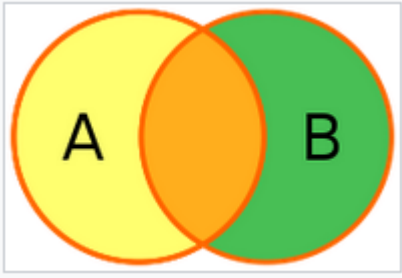


https://www.mysqltutorial.org/mysql-basics/



https://www.w3resource.com/sqlite/sqlite-left-join.php

Observação sobre o FULL OUTER JOIN no MySQL

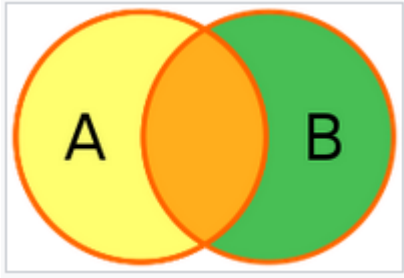



```
SELECT *
FROM empregado FULL OUTER JOIN departamento
  ON empregado.IDDepartamento = departamento.IDDepartamento;
```

Empregado.ÚltimoNome	Empregado.IDDepartamento	Departamento.NomeDepartamento	Departamento.IDDepartamento
Smith	34	Administrativo	34
Jones	33	Engenharia	33
Robinson	34	Administrativo	34
Williams	NULO	NULO	NULO
Heisenberg	33	Engenharia	33
Rafferty	31	Administrativo	31
NULO	NULO	*Marketing*	*35*

https://pt.wikipedia.org/wiki/Join_(SQL)#Jun%C3%A7%C3%A3o_interna_(inner_join)
https://dba.stackexchange.com/questions/55498/full-outer-join-does-not-work-for-two-small-tables-error-1064

Alternativa para o FULL OUTER JOIN no MySQL




```
SELECT * FROM TableA A
LEFT JOIN TableB B
ON A.name = B.name
UNION
SELECT * FROM TableA A
RIGHT JOIN TableB B
ON A.name = B.name
```

https://dba.stackexchange.com/questions/55498/full-outer-join-does-not-work-for-two-small-tables-error-1064

SELF JOIN

```
SELECT COALESCE(e.first_name, ' ', e.last_name)
        AS employee,
        COALESCE(m.first_name, ' ', m.last_name)
        AS manager
  FROM    employees e
        INNER JOIN
          employees m
        ON m.employee_id = e.manager_id
  ORDER BY manager;
```

employee	manager
Bruce Ernst	Alexander Hunold
David Austin	Alexander Hunold
Valli Pataballa	Alexander Hunold
Diana Lorentz	Alexander Hunold
Alexander Khoo	Den Raphaely
Shelli Baida	Den Raphaely
Sigal Tobias	Den Raphaely
Guy Himuro	Den Raphaely
Karen Colmenares	Den Raphaely

SELF JOIN

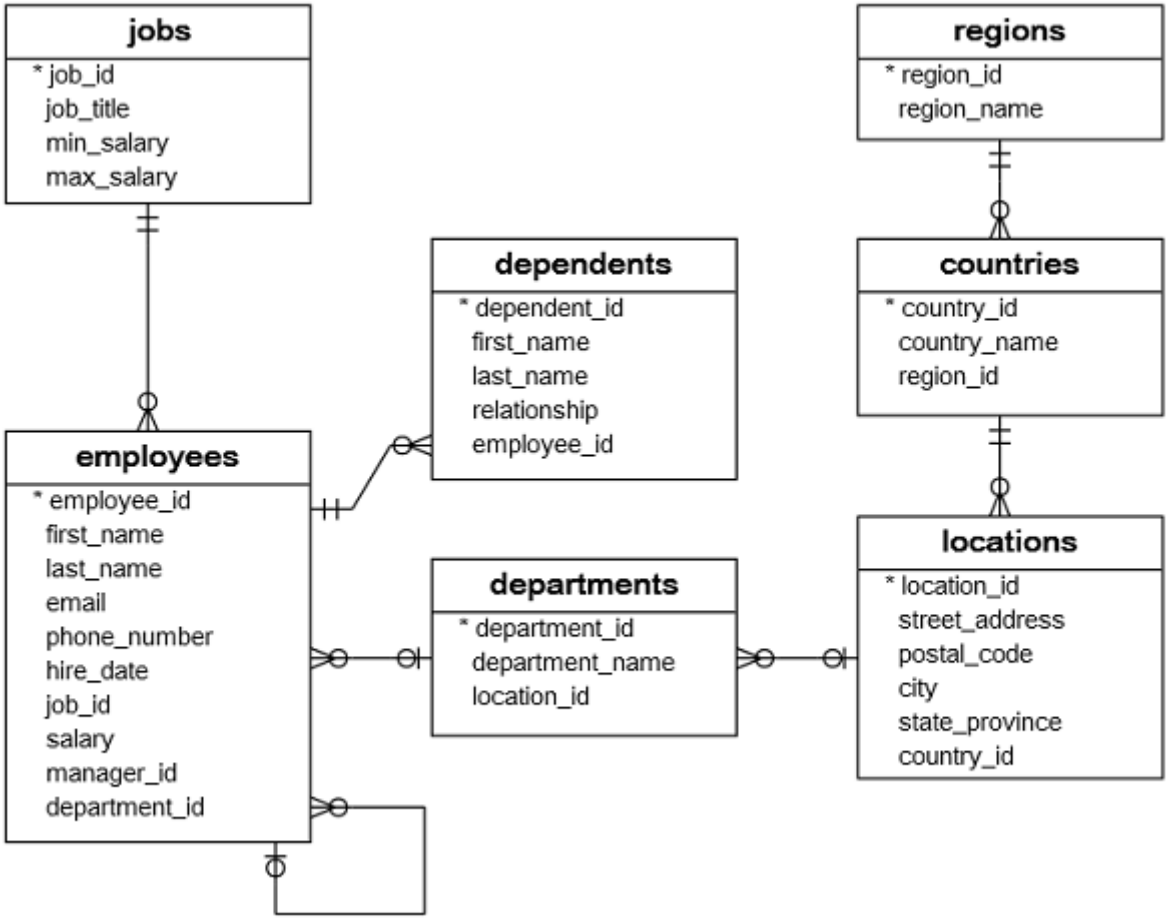
```
SELECT COALESCE(e.first_name, ' ', e.last_name)
        AS employee,
        COALESCE(m.first_name, ' ', m.last_name)
        AS manager
  FROM    employees e
        ????
           employees m
        ON m.employee_id = e.manager_id
  ORDER BY manager;
```

Quem é esse empregado?

employee	manager
Steven King	NULL
Bruce Ernst	Alexander Hunold
David Austin	Alexander Hunold
Valli Pataballa	Alexander Hunold
Diana Lorentz	Alexander Hunold
Alexander Khoo	Den Raphaely
Shelli Baida	Den Raphaely
Sigal Tobias	Den Raphaely
Guy Himuro	Den Raphaely

Quais **tipos de perguntas** alguém poderia fazer sobre os dados se fosse usá-los para o desenvolvimento de um aplicativo ou uma ideia de negócio?

- Qual é o valor médio, máximo e mínimo dos dados?
- E se agruparmos os dados por categoria (utilizando HAVING)?
- De quais maneiras podemos agrupar os valores que não existem ainda (utilizando CASE)?
- De quais maneiras interessantes podemos filtrar os dados (utilizando AND/OR)?



https://www.sqltutorial.org/sql-sample-database/
https://github.com/renatobdo/BD2/tree/main/semana2

GROUP BY

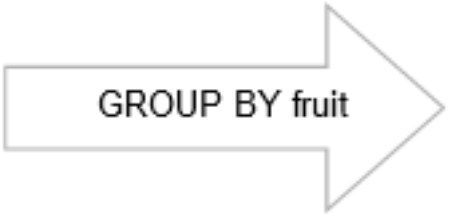
id	fruit
1	Apple
2	Orange
3	Apple
4	Banana
5	Orange

fruit
Apple
Banana
Orange

```
SELECT fruit
FROM
sample_table
GROUP BY  fruit;
```

GROUP BY com funções agregadas (MIN, MAX, AVG, SUM ou COUNT)

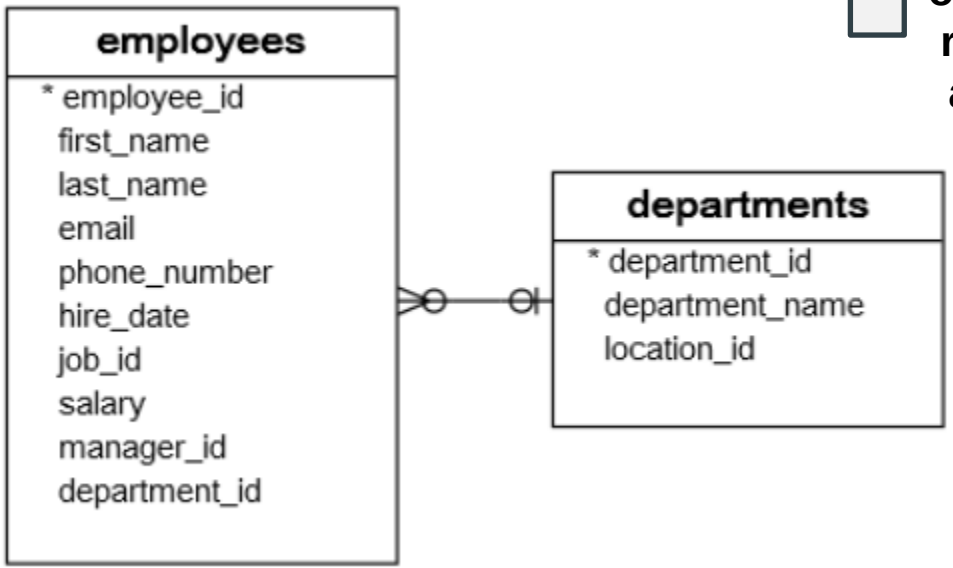
id	fruit
1	Apple
2	Orange
3	Apple
4	Banana
5	Orange




```
SELECT fruit,
COUNT(id) FROM
sample_table
GROUP BY
fruit;
```

fruit	count(id)
Apple	2
Banana	1
Orange	2

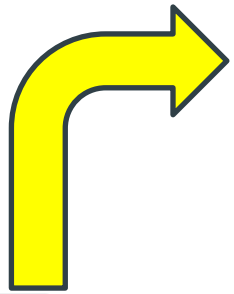
GROUP BY



Como obtemos o resultado ao lado?

```
+---------------+
| department_id |
+---------------+
|             1 |
|             2 |
|             3 |
|             4 |
|             5 |
|             6 |
|             7 |
|             8 |
|             9 |
|            10 |
|            11 |
+---------------+
11 rows in set (0.00 sec)
```

GROUP BY com a função COUNT

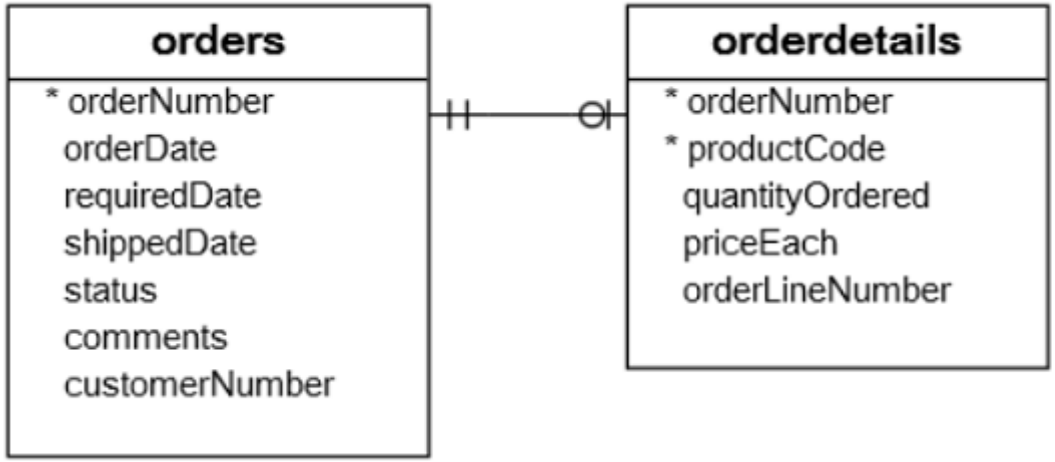


Como obtemos a quantidade de funcionários por departamento?

Primeiro obtemos as linhas da tabela employees por department_id
Depois a função **count** deve retornar o número de empregados de cada departamento

```
+---------------+-----------+
| department_id | headcount |
+---------------+-----------+
|             1 |         1 |
|             2 |         2 |
|             3 |         6 |
|             4 |         1 |
|             5 |         7 |
|             6 |         5 |
|             7 |         1 |
|             8 |         6 |
|             9 |         3 |
|            10 |         6 |
|            11 |         2 |
+---------------+-----------+
11 rows in set (0.00 sec)
```

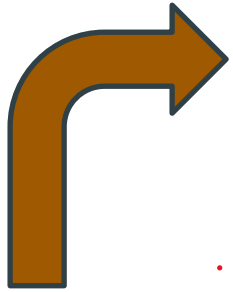
GROUP BY com a função COUNT



status	COUNT(*)
Cancelled	6
Disputed	3
In Process	6
On Hold	4
Resolved	4
Shipped	303

Primeiro obtemos as linhas da tabela orders (pedidos) por status
Depois a função **count** deve retornar o número de status de cada pedido

GROUP BY com a função COUNT e INNER JOIN



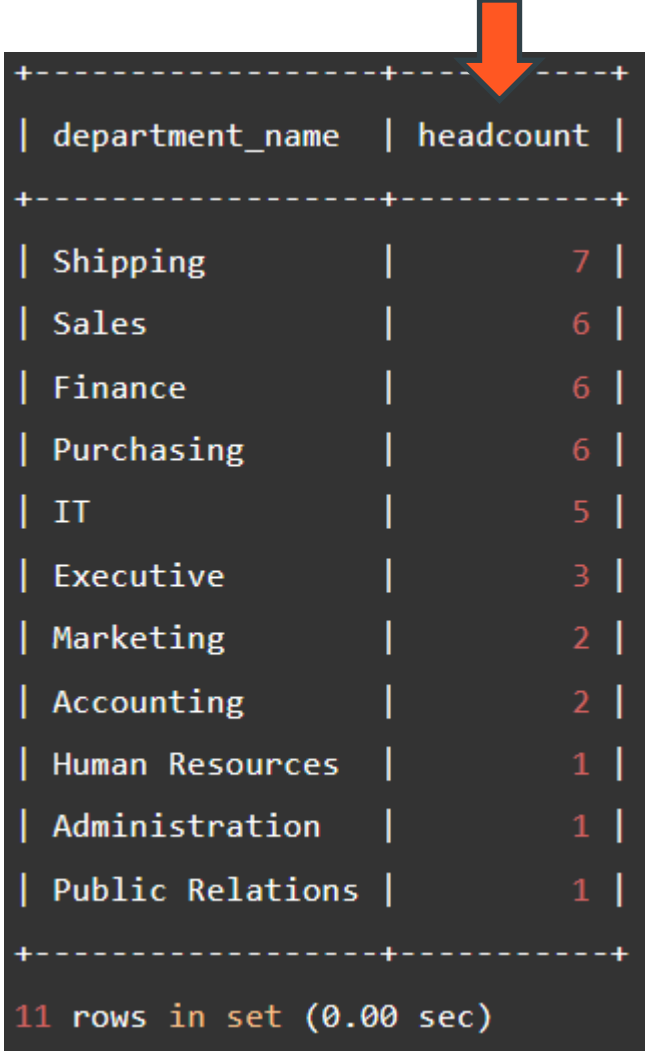
Como obtemos o nome do departamento?

department_name	headcount
Accounting	2
Administration	1
Executive	3
Finance	6
Human Resources	1
IT	5
Marketing	2
Public Relations	1
Purchasing	6
Sales	6
Shipping	7

11 rows in set (0.01 sec)

GROUP BY com a função COUNT, INNER JOIN e ORDER BY...

Como ordenamos pela quantidade de funcionários por departamento?

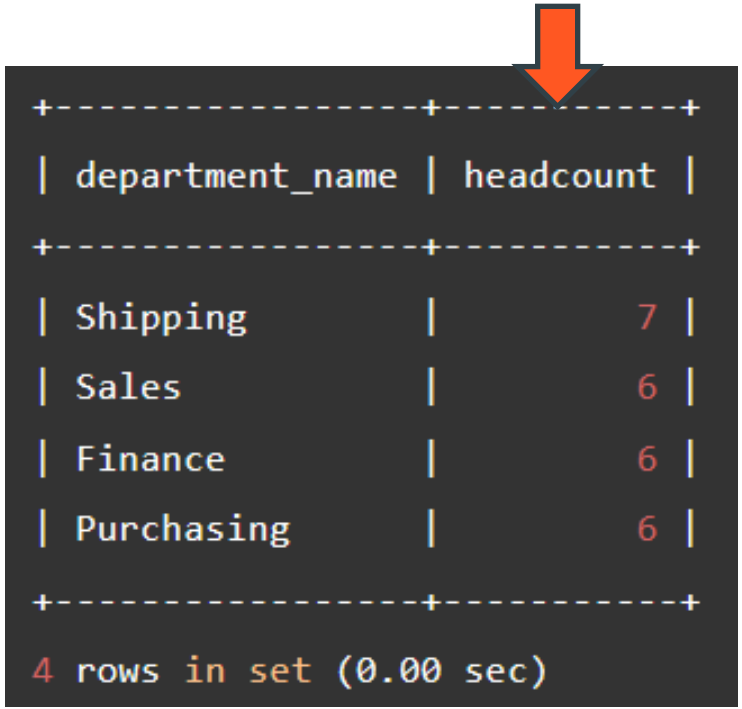


department_name	headcount
Shipping	7
Sales	6
Finance	6
Purchasing	6
IT	5
Executive	3
Marketing	2
Accounting	2
Human Resources	1
Administration	1
Public Relations	1

11 rows in set (0.00 sec)

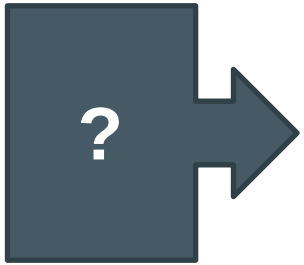
GROUP BY com a função COUNT, INNER JOIN, HAVING e ORDER BY ...

Como ordenamos pela quantidade de funcionários por departamento em que a quantidade de funcionários seja maior que 5?



```
+-----------------+-----------+
| department_name | headcount |
+-----------------+-----------+
| Shipping        |         7 |
| Sales           |         6 |
| Finance         |         6 |
| Purchasing      |         6 |
+-----------------+-----------+
4 rows in set (0.00 sec)
```

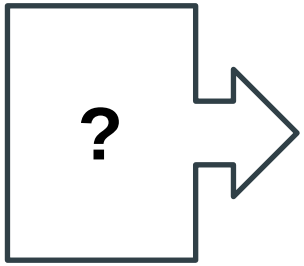
GROUP BY com MIN, MAX, AVG...



department_name	min_salary	max_salary	average_salary
Accounting	8300.00	12000.00	10150.00
Administration	4400.00	4400.00	4400.00
Executive	17000.00	24000.00	19333.33
Finance	6900.00	12000.00	8600.00
Human Resources	6500.00	6500.00	6500.00
IT	4200.00	9000.00	5760.00
Marketing	6000.00	13000.00	9500.00
Public Relations	10000.00	10000.00	10000.00
Purchasing	2500.00	11000.00	4150.00
Sales	6200.00	14000.00	9616.67
Shipping	2700.00	8200.00	5885.71

11 rows in set (0.01 sec)

GROUP BY com SUM



department_name	total_salary
Accounting	20300.00
Administration	4400.00
Executive	58000.00
Finance	51600.00
Human Resources	6500.00
IT	28800.00
Marketing	19000.00
Public Relations	10000.00
Purchasing	24900.00
Sales	57700.00
Shipping	41200.00

11 rows in set (0.01 sec)

GROUP BY com múltiplas colunas

department_name	job_title	COUNT(employee_id)
Accounting	Accounting Manager	1
Accounting	Public Accountant	1
Administration	Administration Assistant	1
Executive	Administration Vice President	2
Executive	President	1
Finance	Accountant	5
Finance	Finance Manager	1
Human Resources	Human Resources Representative	1
IT	Programmer	5
Marketing	Marketing Manager	1
Marketing	Marketing Representative	1

Mais exemplos de conjuntos de agrupamentos

- Para realizar esses exemplos devemos criar a tabela inventory e realizar os inserts. Veja os scripts em: https://github.com/renatobdo/BD2/blob/main/semana3/Agrupamentos.sql

A)

warehouse	product	qty
San Fransisco	iPhone	260
San Fransisco	Samsung	300
San Jose	iPhone	300
San Jose	Samsung	350

B)

warehouse	qty
San Fransisco	560
San Jose	650

C)

product	qty
iPhone	560
Samsung	650

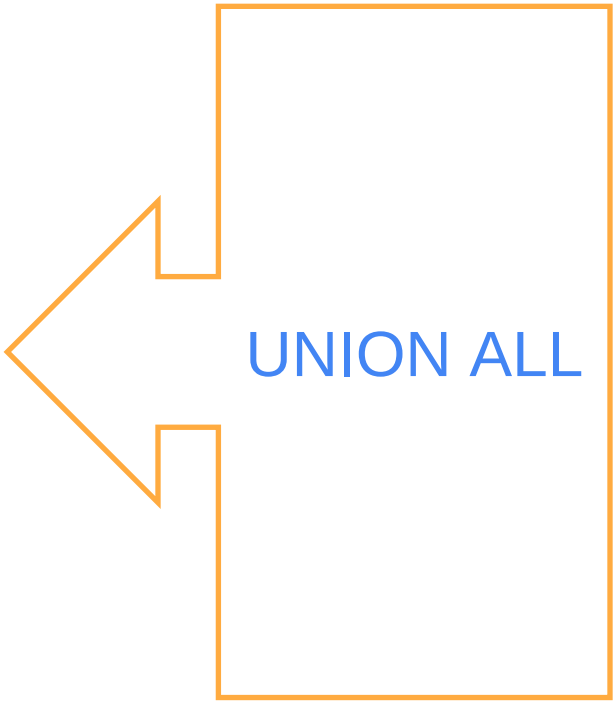
D)

qty
1210

Como obter os resultados A, B, C e D acima?

Mais exemplos de conjuntos de agrupamentos

warehouse	product	qty
San Fransisco	iPhone	260
San Fransisco	Samsung	300
San Jose	iPhone	300
San Jose	Samsung	350
San Fransisco	NULL	560
San Jose	NULL	650
NULL	iPhone	560
NULL	Samsung	650
NULL	NULL	1210



GROUP BY ROLL UP

- É uma extensão do GROUP BY
- Permite incluir linhas extras que representam subtotais
  - Linhas “super agregadas”

warehouse	product	model	quantity
San Fransisco	iPhone	6s	50
San Fransisco	iPhone	7	10
San Fransisco	iPhone	X	200
San Fransisco	Samsung	Galaxy S	200
San Fransisco	Samsung	Note 8	100
San Jose	iPhone	6s	100
San Jose	iPhone	7	50
San Jose	iPhone	X	150
San Jose	Samsung	Galaxy S	200
San Jose	Samsung	Note 8	150

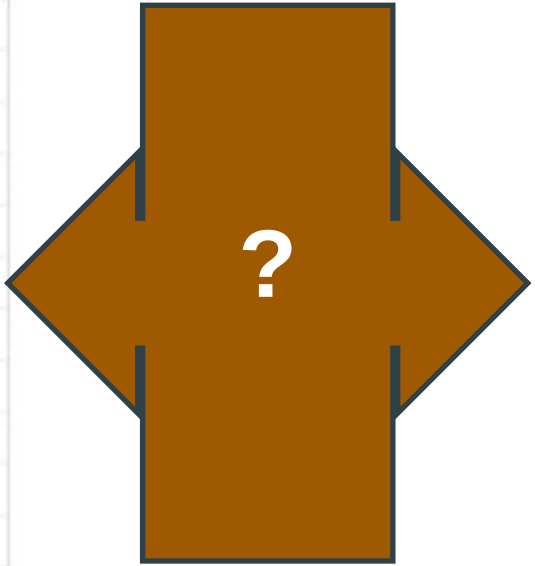
```
SELECT
warehouse,
SUM(quantity)
FROM
inventory
GROUP BY
warehouse with
ROLLUP;
```

warehouse	SUM(quantity)
San Fransisco	560
San Jose	650
NULL	1210

GROUP BY ROLL UP

- É uma extensão do GROUP BY
- Permite incluir linhas extras que representam subtotais
  - Linhas “super agregadas”

warehouse	product	model	quantity
San Fransisco	iPhone	6s	50
San Fransisco	iPhone	7	10
San Fransisco	iPhone	X	200
San Fransisco	Samsung	Galaxy S	200
San Fransisco	Samsung	Note 8	100
San Jose	iPhone	6s	100
San Jose	iPhone	7	50
San Jose	iPhone	X	150
San Jose	Samsung	Galaxy S	200
San Jose	Samsung	Note 8	150



warehouse	SUM(quantity)
San Fransisco	560
San Jose	650
All warehouses	1210

GROUP BY ROLL UP com múltiplas colunas

warehouse	product	model	quantity
San Fransisco	iPhone	6s	50
San Fransisco	iPhone	7	10
San Fransisco	iPhone	X	200
San Fransisco	Samsung	Galaxy S	200
San Fransisco	Samsung	Note 8	100
San Jose	iPhone	6s	100
San Jose	iPhone	7	50
San Jose	iPhone	X	150
San Jose	Samsung	Galaxy S	200
San Jose	Samsung	Note 8	150

```
SELECT
warehouse,
product,
SUM(quantity)
FROM  inventory
GROUP BY
warehouse,
product
```

	warehouse	product	SUM(quantity)
▶	San Jose	iPhone	300
	San Fransisco	iPhone	260
	San Jose	Samsung	350
	San Fransisco	Samsung	300

?

	warehouse	product	SUM(quantity)
▶	San Fransisco	iPhone	260
	San Fransisco	Samsung	300
	San Fransisco	NULL	560
	San Jose	iPhone	300
	San Jose	Samsung	350
	San Jose	NULL	650
	NULL	NULL	1210

Moodle

Questionário 1

SQL CASE

employees
* employee_id
first_name
last_name
email
phone_number
hire_date
job_id
salary
manager_id
department_id

first_name	last_name	hire_date	aniversary
Adam	Fripp	1997-04-10	3 years
Alexander	Hunold	1990-01-03	10 years
Alexander	Khoo	1995-05-18	5 years
Britney	Everett	1997-03-03	3 years
Bruce	Ernst	1991-05-21	NULL
Charles	Johnson	2000-01-04	NULL
Daniel	Faviet	1994-08-16	NULL

SQL CASE

employees
* employee_id
first_name
last_name
email
phone_number
hire_date
job_id
salary
manager_id
department_id

first_name	last_name	salary	evaluation
Adam	Fripp	8200.00	High
Alexander	Hunold	9000.00	High
Alexander	Khoo	3100.00	Average
Britney	Everett	3900.00	Average
Bruce	Ernst	6000.00	High
Charles	Johnson	6200.00	High
Daniel	Faviet	9000.00	High
David	Austin	4800.00	Average
Den	Raphaely	11000.00	High
Diana	Lorentz	4200.00	Average
Guy	Himuro	2600.00	Low
Hermann	Baer	10000.00	High
Irene	Mikkilineni	2700.00	Low

Dúvidas projeto

Utilizem o final da aula para esclarecimento de dúvidas e desenvolvimento da atividade

Data de entrega dia 25/08/2022

Referências Bibliográficas

- https://pt.wikipedia.org/wiki/Join_(SQL)#Jun%C3%A7%C3%A3o_interna_(inner_join)
- https://dba.stackexchange.com/questions/55498/full-outer-join-does-not-work-for-two-small-tables-error-1064
- https://www.sqltutorial.org/sql-self-join/
- https://www.sqltutorial.org/sql-group-by/
- https://www.sqltutorial.org/sql-rollup/