

# SELECT

## *Parte 4*

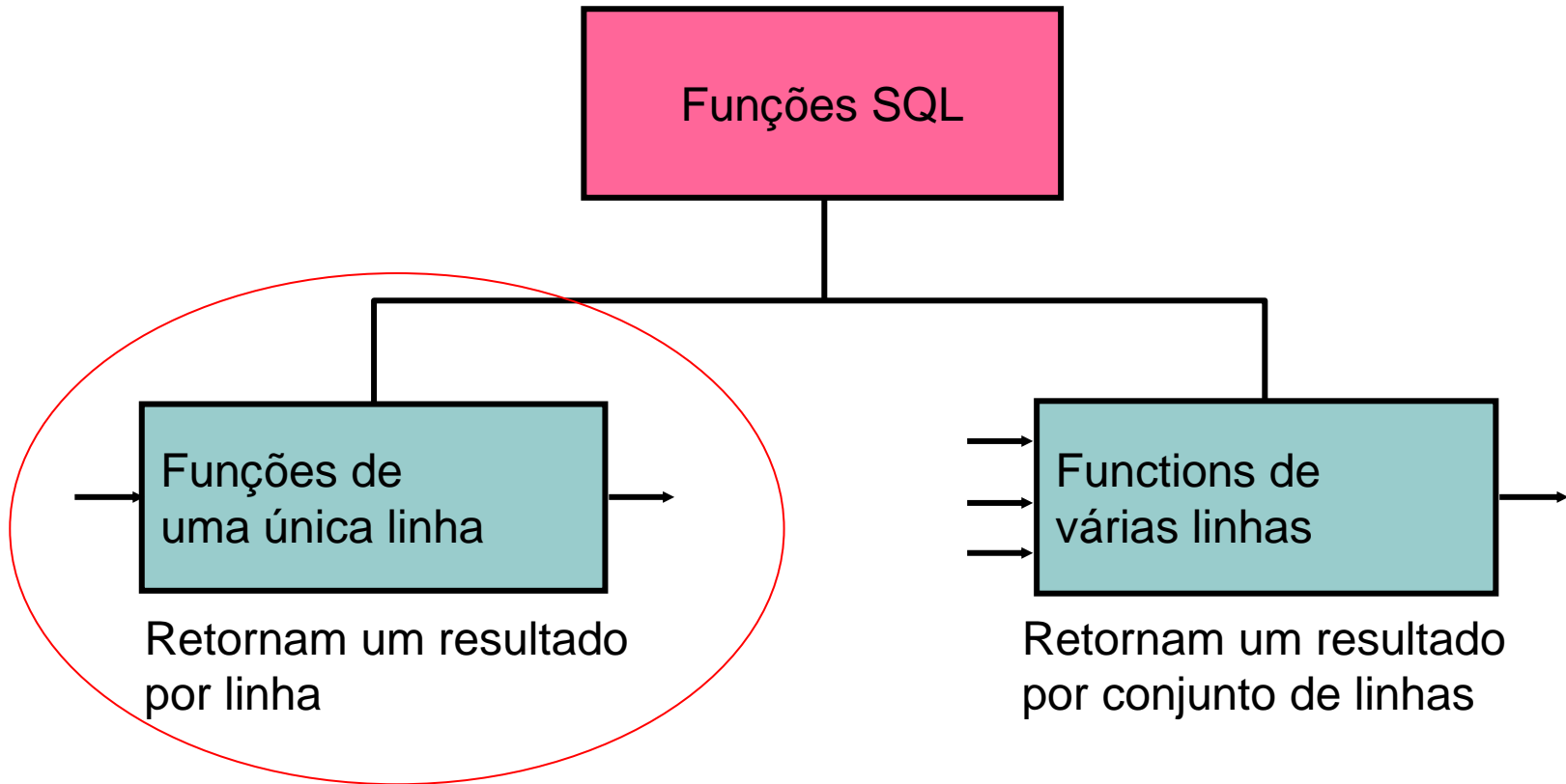
Algumas funções SQL



*Luciano Melo*  
*profluciano.melo@fiap.com.br*



## FUNÇÕES SQL



## Functions de Manipulação de Maiúsculas e Minúsculas

**Estas functions convertem letras maiúsculas em minúsculas e vice-versa em strings de caracteres:**

Function	Resultado
<code>LOWER('SQL Course')</code>	sql course
<code>UPPER('SQL Course')</code>	SQL COURSE
<code>INITCAP('SQL Course')</code>	Sql Course

## Exemplo:

Exiba o número, o nome e o número de departamento do funcionário Higgins:

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  last_name = 'higgins';
no rows selected
```

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  LOWER(last_name) = 'higgins';
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
205	Higgins	110

## Functions de Manipulação de Caracteres

**Estas functions manipulam strings de caracteres:**

Function	Resultado
<code>CONCAT('Hello', 'World')</code>	HelloWorld
<code>SUBSTR('HelloWorld',1,5)</code>	Hello
<code>LENGTH('HelloWorld')</code>	10

## Funções aplicada a números

- **ROUND:** Arredonda o valor até o decimal especificado
- **TRUNC:** Trunca o valor até o decimal especificado
- **MOD:** Retorna o resto da divisão

Function	Resultado
ROUND (45.926, 2)	45.93
TRUNC (45.926, 2)	45.92
MOD (1600, 300)	100

## A tabela DUAL

**DUAL é uma tabela fictícia que pode ser usada para exibir resultados de functions e cálculos.**

```
SELECT ROUND(45.923,2), ROUND(45.923,0)
FROM DUAL;
```

```
SELECT TRUNC(45.923), TRUNC(45.215)
FROM DUAL;
```

```
SELECT MOD(10,2), MOD(10,3)
FROM DUAL;
```

```
SELECT last_name, salary, MOD(salary, 5000)
FROM employees
WHERE job_id = 'SA_REP';
```

## Função NVL

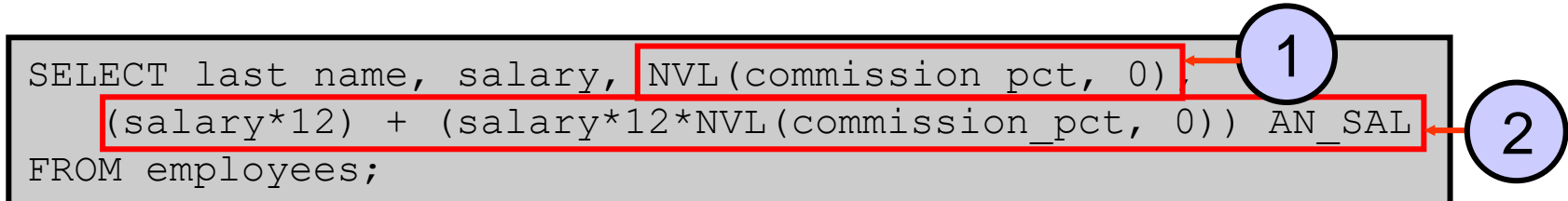
**Converte um valor nulo em um valor real:**

- **É possível usar os tipos de dados de data, caractere e número.**
- **A correspondência entre os tipos de dados é necessária:**
  - `NVL(commission_pct,0)`
  - `NVL(hire_date,'01-JAN-97')`
  - `NVL(job_id,'No Job Yet')`

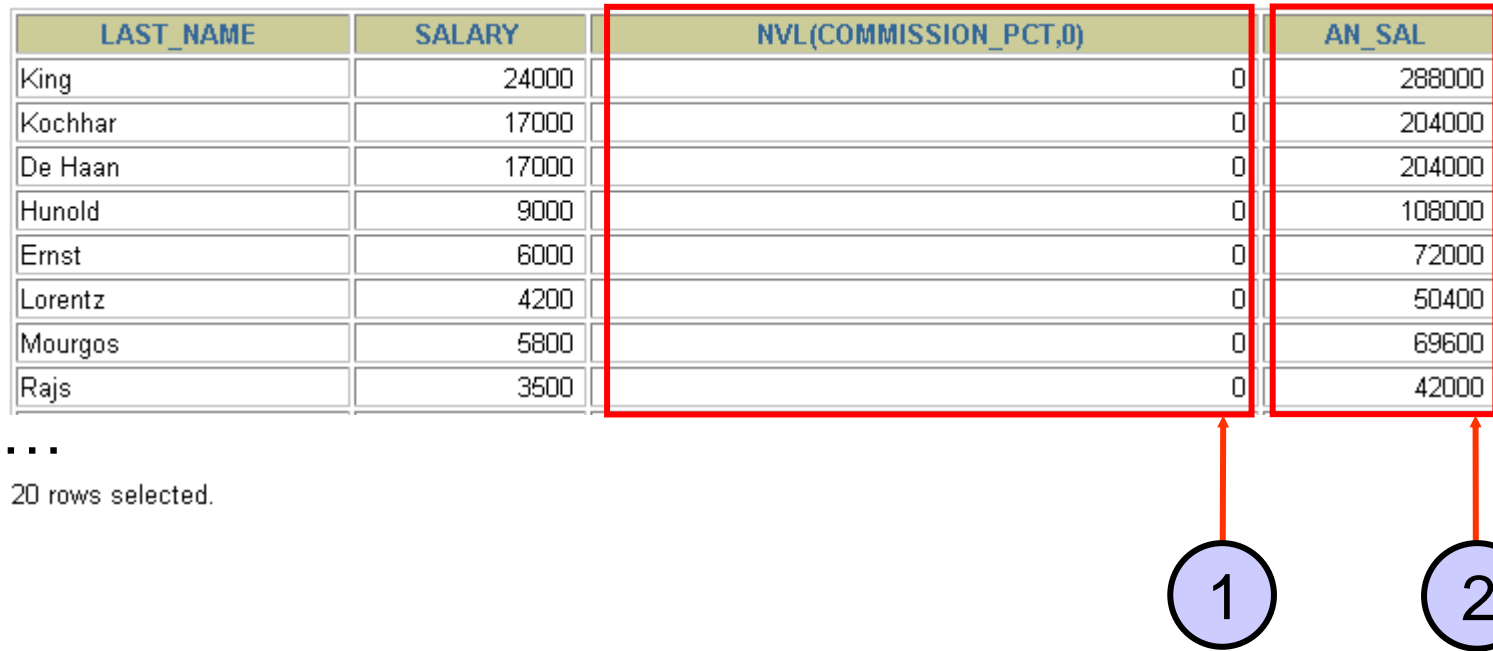


## Exemplos:

```
SELECT last name, salary, NVL(commission_pct, 0),  
       (salary*12) + (salary*12*NVL(commission_pct, 0)) AN_SAL  
FROM employees;
```



LAST_NAME	SALARY	NVL(COMMISSION_PCT,0)	AN_SAL
King	24000	0	288000
Kochhar	17000	0	204000
De Haan	17000	0	204000
Hunold	9000	0	108000
Ernst	6000	0	72000
Lorentz	4200	0	50400
Mourgos	5800	0	69600
Rajs	3500	0	42000



...

20 rows selected.

## Expressão CASE

```
SELECT e.FIRST_NAME,  
       d.DEPARTMENT_NAME,  
       CASE WHEN e.SALARY < 3000 THEN 'Nível C'  
            WHEN e.SALARY < 5000 THEN 'Nível B'  
            ELSE 'Nível A'  
       END "Nível Salarial"  
FROM EMPLOYEES e LEFT JOIN DEPARTMENTS d ON (d.department_id = e.department_id)  
order by e.first_name;
```

	FIRST_NAME	DEPARTMENT_NAME	Nível Salarial
19	David	Sales	Nível A
20	David	IT	Nível B
21	David	Sales	Nível A
22	Den	Purchasing	Nível A
23	Diana	IT	Nível B
24	Donald	Shipping	Nível C
25	Douglas	Shipping	Nível C
26	Eleni	Sales	Nível A
27	Elizabeth	Sales	Nível A
28	Ellen	Sales	Nível A
29	Gerald	Sales	Nível A
30	Girard	Shipping	Nível C
31	Guy	Purchasing	Nível C
32	Harrison	Sales	Nível A
33	Hazel	Shipping	Nível C

## Função TO\_CHAR

**Usada para converter números ou datas em caracter**

```
TO_CHAR( value [, format_mask] )
```

**Quando o valor é uma data, use o formato (mascara) de data**

```
SELECT TO_CHAR(SYSDATE, 'DD/MM/YYYY') FROM DUAL;  
SELECT TO_CHAR(SYSDATE, 'DD-Mon-YYYY HH24:MI:SS') FROM DUAL;
```

**Quando o valor é um número, use o formato para números**

```
SELECT TO_CHAR(100000/2, '$999G990D00') FROM DUAL;
```

## Função TO\_NUMBER

Usada para converter uma string em número

```
TO_NUMBER( value [, format_mask] )
```

### Exemplo

```
SELECT TO_NUMBER('$9,500.55','$999,999.99') FROM DUAL;
```

## Função TO\_DATE

Usada para converter uma string em uma data

```
TO_DATE( value [, format_mask] )
```

### Exemplos

```
SELECT TO_DATE('10-Jun-2022', 'DD-Mon-YYYY') FROM DUAL;
```

```
SELECT TO_DATE('10/06/2022 10:30', 'DD/MM/YYYY HH24:MI') FROM DUAL;
```

➔ *Lembre-se que um campo do tipo date no Oracle guarda data, hora, minutes e segundos*

## Função TO\_NUMBER

**Usada para converter uma string em número**

```
TO_NUMBER( value [, format_mask] )
```

### Exemplo

```
SELECT TO_NUMBER('$9,500.55','$999,999.99') FROM DUAL;
```

## Aprofunde seu conhecimento ...

- Existem várias outras funções Oracle para caracteres, números, datas, etc
- Aritmética de datas no Oracle



## Pesquise sobre Funções de única Linha (Single Row Functions)

### Referências

### Single-Row Functions:

<https://docs.oracle.com/en/database/oracle/oracle-database/21/sqlrf/Single-Row-Functions.html#GUID-B93F789D-B486-49FF-B0CD-0C6181C5D85C>

<https://beginner-sql-tutorial.com/oracle-functions.htm>

# SELECT

## *Parte 5*

### Funções de Grupo

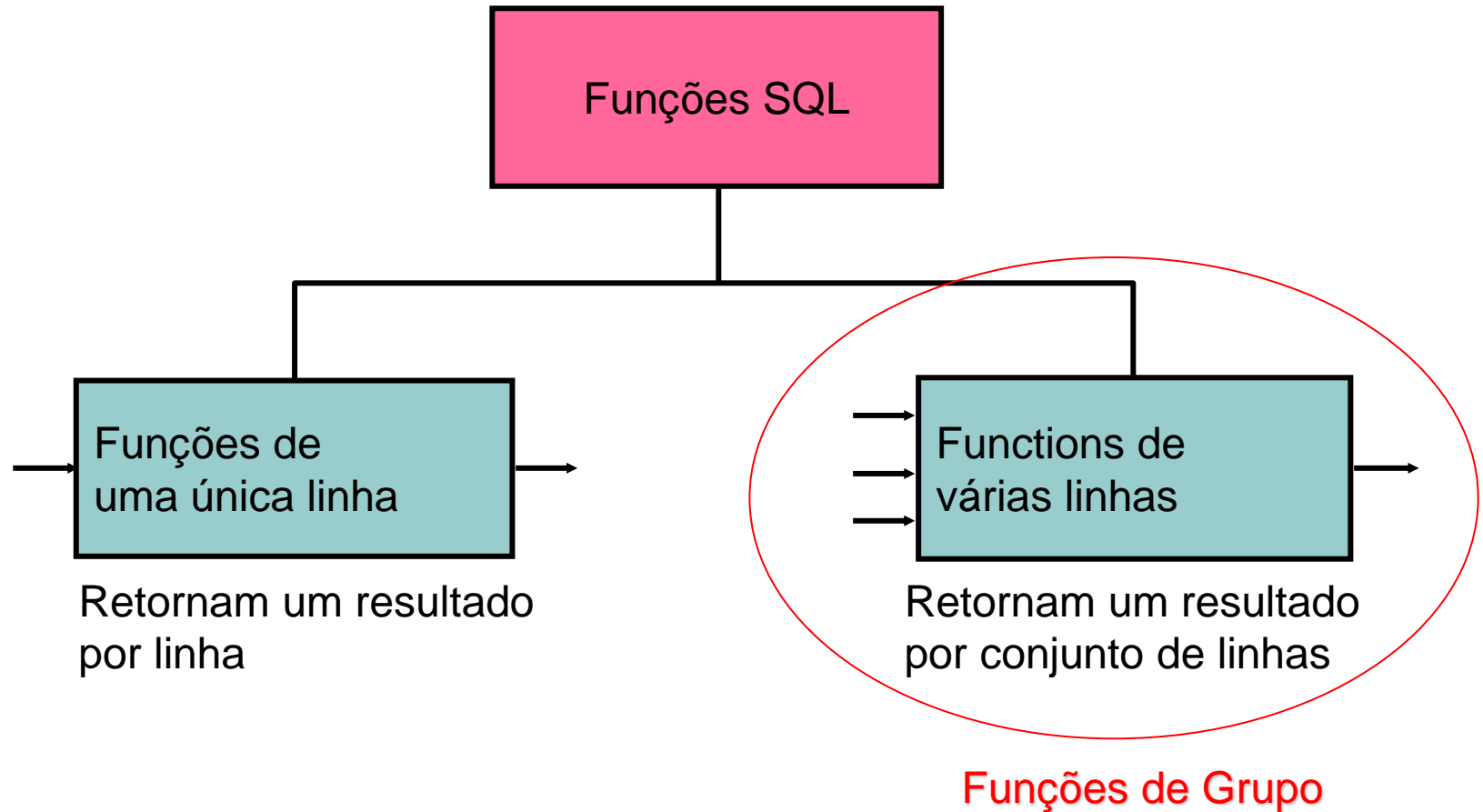


*Luciano Melo*  
*profluciano.melo@fiap.com.br*





## FUNÇÕES SQL



## ➤ Funções de Grupo

As funções de grupo operam em conjuntos de linhas para gerar um resultado por grupo!

EMPLOYEES

DEPARTMENT_ID	SALARY
90	24000
90	17000
90	17000
60	9000
60	6000
60	4200
50	5800
50	3500
50	3100
50	2600
50	2500
80	10500
80	11000
80	8600
	7000
10	4400

...  
20 rows selected.

Salário máximo na  
tabela EMPLOYEES

MAX(SALARY)
24000

## ➤ Funções de Grupo

Exemplos de funções de grupos

- **AVG** → Média
- **COUNT** → Totalizador (quantidade)
- **MAX** → Maior valor
- **MIN** → Menor valor
- **SUM** → Somatório dos valores
- **Entre outras ...**

## ➤ Funções de Grupo

- Exemplos de funções de grupos
  - **AVG** → Média
  - **COUNT** → Totalizador (quantidade)
  - **MAX** → Maior valor
  - **MIN** → Menor valor
  - **SUM** → Somatório dos valores
  - **Entre outras ...**
- Sintaxe

```
SELECT      [column,] group_function(column), ...  
FROM        table  
[WHERE      condition]  
[GROUP BY  column]  
[ORDER BY  column];
```

## ➤ Funções de Grupo

As funções AVG, MAX, MIN e SUM podem ser usadas para dados numéricos.

```
SELECT AVG(salary), MAX(salary),  
       MIN(salary), SUM(salary)  
FROM   employees  
WHERE  job_id LIKE '%REP%';
```

AVG(SALARY)	MAX(SALARY)	MIN(SALARY)	SUM(SALARY)
8150	11000	6000	32600

## ➤ Funções de Grupo

As funções `MIN` e `MAX` também podem ser usadas para datas e caracteres.

```
SELECT MIN(hire date), MAX(hire date)
FROM   employees;
```

MIN(HIRE_	MAX(HIRE_
17-JUN-87	29-JAN-00

## ➤ Funções de Grupo

### FUNÇÃO COUNT

**COUNT (\*)** retorna o número de linhas de uma tabela:

1

```
SELECT COUNT(*)  
FROM employees  
WHERE department_id = 50;
```

COUNT(\*)

5

**COUNT (expr)** retorna o número de linhas com valores não nulos para *expr*:

2

```
SELECT COUNT(commission_pct)  
FROM employees  
WHERE department_id = 80;
```

COUNT(COMMISSION\_PCT)

3

## ➤ Funções de Grupo

- Funções de grupos e valores NULOS:

As funções de grupo ignoram valores nulos. Se for necessário, use a função NVL para incluí-los.

1

```
SELECT AVG (commission_pct)
FROM   employees;
```

AVG(COMMISSION\_PCT)

.2125

- Considerando os valores nulos com a função NVL

2

```
SELECT AVG (NVL (commission_pct, 0))
FROM   employees;
```

AVG(NVL(COMMISSION\_PCT,0))

.0425



# SELECT

## ➤ Funções de Grupo

Criando mais de um grupo de resultados

EMPLOYEES

DEPARTMENT_ID	SALARY
10	4400
20	13000
20	6000
50	5800
50	3500
50	3100
50	2500
50	2600
60	9000
60	6000
60	4200
80	10500
80	8600
80	11000
90	24000
90	17000

...

20 rows selected.

4400

9500

3500

6400

10033

Salário médio  
na tabela  
EMPLOYEES  
para cada  
departamento

DEPARTMENT_ID	AVG(SALARY)
10	4400
20	9500
50	3500
60	6400
80	10033.3333
90	19333.3333
110	10150
	7000

## ➤ Funções de Grupo

- Para criar grupos, use a cláusula `GROUP BY`.
- Todas as colunas da lista `SELECT` que não são funções de grupo devem estar incluídas na cláusula `GROUP BY`.

```
SELECT  department_id, AVG(salary)
FROM    employees
GROUP BY department_id ;
```

DEPARTMENT_ID	AVG(SALARY)
10	4400
20	9500
50	3500
60	6400
80	10033.3333
90	19333.3333
110	10150
	7000

8 rows selected.

## ➤ Funções de Grupo

### Criando mais de um grupo de resultados

EMPLOYEES

DEPARTMENT_ID	JOB_ID	SALARY
90	AD_PRES	24000
90	AD_VP	17000
90	AD_VP	17000
60	IT_PROG	9000
60	IT_PROG	6000
60	IT_PROG	4200
50	ST_MAN	5800
50	ST_CLERK	3500
50	ST_CLERK	3100
50	ST_CLERK	2600
50	ST_CLERK	2500
80	SA_MAN	10500
80	SA_REP	11000
80	SA_REP	8600
...		
20	MK_REP	6000
110	AC_MGR	12000
110	AC_ACCOUNT	8300

20 rows selected.

Adicione os  
salários à tabela  
EMPLOYEES  
para cada cargo,  
agrupados por  
departamento

DEPARTMENT_ID	JOB_ID	SUM(SALARY)
10	AD_ASST	4400
20	MK_MAN	13000
20	MK_REP	6000
50	ST_CLERK	11700
50	ST_MAN	5800
60	IT_PROG	19200
80	SA_MAN	10500
80	SA_REP	19600
90	AD_PRES	24000
90	AD_VP	34000
110	AC_ACCOUNT	8300
110	AC_MGR	12000
	SA_REP	7000

13 rows selected.

## ➤ Funções de Grupo

Para criar mais de um grupo, basta colocar mais de uma coluna na cláusula group by.

```
SELECT    department_id dept_id, job_id, SUM(salary)
FROM      employees
GROUP BY  department id, job id ;
```

DEPT_ID	JOB_ID	SUM(SALARY)
10	AD_ASST	4400
20	MK_MAN	13000
20	MK_REP	6000
50	ST_CLERK	11700
50	ST_MAN	5800
60	IT_PROG	19200
80	SA_MAN	10500
80	SA_REP	19600
90	AD PRES	24000
90	AD_VP	34000
110	AC_ACCOUNT	8300
110	AC_MGR	12000
	SA_REP	7000

13 rows selected.

## ➤ Funções de Grupo

### Filtrando resultados dos grupos

EMPLOYEES

DEPARTMENT_ID	SALARY
90	24000
90	17000
90	17000
60	9000
60	6000
60	4200
50	5800
50	3500
50	3100
50	2600
50	2500
80	10500
80	11000
80	8600
...	
20	6000
110	12000
110	8300

20 rows selected.

O salário máximo por departamento quando for maior que US\$ 10.000

DEPARTMENT_ID	MAX(SALARY)
20	13000
80	11000
90	24000
110	12000

## ➤ Funções de Grupo

- Para restringir grupos de resultados, use a cláusula HAVING
- As linhas são agrupadas, a função de grupo é executada e então o filtro é aplicado.

```
SELECT    department_id, MAX(salary)
FROM      employees
GROUP BY  department_id
HAVING    MAX(salary) > 10000 ;
```

DEPARTMENT_ID	MAX(SALARY)
20	13000
80	11000
90	24000
110	12000

## Aprofunde seu conhecimento ...

- Funções de grupos são extremamente importantes e muito utilizadas.
- São fundamentais para a parte de análise de dados (analytics) e BI



### Pesquise sobre Funções de Grupo (Aggregate Functions)

#### Referências

#### Funções de Grupo:

<https://docs.oracle.com/en/database/oracle/oracle-database/21/sqlrf/Aggregate-Functions.html#GUID-62BE676B-AF18-4E63-BD14-25206FEA0848>

## ➤ Pratique: Execute os comandos SELECT abaixo

```
SELECT department_id, sum(salary) Total, avg(salary) Media  
Max(salary) Top_salary, min(salary) low_salary FROM employees  
GROUP BY department_id  
ORDER BY Total desc;
```

```
SELECT department_id, job_id, max(hire_date), min(hire_date)  
FROM employees  
GROUP BY department_id, job_id;
```

```
SELECT count(distinct department_id) from employees;
```

```
SELECT manager_id, count(*) from employees  
WHERE department_id in (10,20,30,40,80)  
FROM JOBS;
```





# Exercícios

*profluciano.melo@fiap.com.br*

Mostrar o nome do departamento, a soma dos salários, o total de empregados de cada departamento. Ordene o resultado de forma a mostra os departamentos com maior folha salarial primeiro

```
SELECT d.department_name , sum (e.salary) total_salario, count(*) total_emp
FROM EMPLOYEES e JOIN DEPARTMENTS d on (e.department_id = d.department_id)
GROUP BY d.department_name
order by total_salario desc
```

Mostrar o nome do departamento, o título da profissão, o total de salários e a media dos salários de cada título em cada departamento dos funcionários alocados em departamentos

```
SELECT d.department_name , j.job_title, sum (e.salary) total, avg (e.salary) media
FROM EMPLOYEES e JOIN DEPARTMENTS d on (e.department_id = d.department_id)
      JOIN JOBS j on (e.job_id = j.job_id)
GROUP BY d.department_name , j.job_title
```

# SELECT

Mostrar o nome do departamento, o total de salário e quantos funcionários existem em cada departamento. Para os funcionários não alocados em departamento, o nome do departamento de aparecer como “Sem Alocação”

```
SELECT NVL(d.department_name, 'Sem Alocação') department, sum(e.salary), count(*)
FROM EMPLOYEES e LEFT JOIN DEPARTMENTS d on (e.department_id = d.department_id)
GROUP BY NVL(d.department_name, 'Sem Alocação')
```

Mostrar o nome da cidade e o total de salários em cada cidade. Liste apenas as cidades cujo total de salários seja superior a 50000

```
SELECT l.city , sum (e.salary) total_salario
FROM EMPLOYEES e JOIN DEPARTMENTS d on (e.department_id = d.department_id)
                JOIN LOCATIONS l on (d.location_id = l.location_id)
GROUP BY l.city
having sum (e.salary) > 50000
```

Mostrar o nome de todas as cidades que não tem nenhum empregado trabalhando nela

```
SELECT l.city
FROM EMPLOYEES e right JOIN DEPARTMENTS d on (e.department_id = d.department_id)
                right JOIN LOCATIONS l on (d.location_id = l.location_id)
GROUP BY l.city
having count(employee_id) = 0
```