**Design e Desenvolvimento de Banco de Dados**

Comandos DDL são usados para definir objetos em um banco de dados, como: **Tabelas e índices**.

CREATE 🡪 Cria o objeto

ALTER 🡪 Altera a definição do objeto

DROP 🡪 Remove o objeto

TRUNCATE 🡪 Esvazia o conteúdo de uma tabela

Objetos básicos de um BD

Tabela 🡪 Unidade básica de armazenamento; composta de linhas

View 🡪 Representa logicamente subconjuntos de dados de uma ou mais tabelas

Sequencia 🡪 Gera valores numéricos (ORACLE)

Índice 🡪 Melhora o desempenho de algumas consultas

Sinônimo 🡪 Fornece nomes alternativos e objetos

Regras de nomeação

Os nomes de tabelas e colunas:

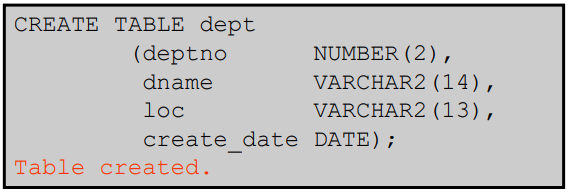
* Devem começar com uma letra
* Devem ter de 1 a 30 caracteres
* Devem conter apenas os caracteres A a Z, a a z, 0 a 9, \_, $ e #
* Não devem duplicar o nome de outro objeto pertencente ao mesmo usuário
* Não devem ser palavras reservadas do servidor Oracle

Principais Tipos de dados:

* VARCHAR2 (size) 🡪 Dados de caractere de tamanho variável
* CHAR (size) 🡪 Dados de caractere de tamanho fixo
* NUMBER (p, s) 🡪 Dados numéricos de tamanho variável
* DATE 🡪 Valores de datas e horários

**Criando Tabelas:**

Comando 🡪 CREATE TABLE nomeTabela( coluna1 tipodedado, coluna2 tipodedado);



Opções adicionais:

* DEFAULT -> coluna1 DATE DEFAULT SYSDATE
* AUTO-INCREMENTO -> coluna1 NUMBER (2) GENERATED ALWAYS AS IDENTITY

**Alterando Tabelas:**

Comando 🡪 ALTER TABLE …

* Adicionar coluna -> ALTER TABLE *nomeTabela* ADD coluna1 NUMBER (2);
* Modificando coluna -> ALTER TABLE *nomeTabela* MODIFY coluna1 NUMBER (4);
* Renomeando coluna -> ALTER TABLE *nomeTabela* RENAME COLUMN coluna1 TO col1;
* Removendo coluna -> ALTER TABLE *nomeTabela* DROP COLUMN col1;

**Removendo Tabelas:**

Comando 🡪 DROP TABLE ...

* Removendo tabela -> DROP TABLE *nomeTabela*;
* Removendo tabela sem enviar para lixeira -> DROP TABLE *nomeTabela* PURGE;
* Removendo tabela com FK -> DROP TABLE *nomeTabela* CASCADE CONSTRAINTS;
* Restaurando tabela -> FLASHBACK TABLE *nomeTabela* TO BEFORE DROP;

**Constraints:**

Tipos de constraints:

1. NOT NULL -> adiciona depois do tipo da coluna ex: coluna1 NUMBER (6) NOT NULL
2. UNIQUE -> CONSTRAINT nomeConstraint\_uk UNIQUE(coluna1);
3. PRIMARY KEY -> CONSTRAINT nomeConstraint\_pk PRIMARY KEY (coluna1);
4. FOREIGN KEY -> CONSTRAINT nomeConstraint\_fk FOREIGN KEY (coluna1) REFERENCES *nomeTabela2* (coluna1);
   1. Sempre referência uma UNIQUE ou PRIMARY KEY
   2. Pode remover os registros filhos quando for apagar a FK se quiser:
      1. ON DELETE CASCADE no final da declaração da CONSTRAINT FK
5. CHECK -> CONSTRAINT nomeConstraint\_min CHECK (salary > 0);

**Esvaziando Tabelas:**

Comando 🡪 TRUNCATE TABLE *nomeTabela*

**Índices:**

* É usado pelo BD para acelerar a recuperação de linhas
* Pode reduzir a entrada/saída de disco por um método de acesso rápido para localizar dados rapidamente
* É independente da tabela que indexa
* É usado e mantido automaticamente pelo banco de dados

Como criar índices:

1. Automaticamente -> Quando define uma CONSTRAINT PRIMARY KEY ou UNIQUE
2. Manualmente -> Criando um índice não exclusivo em colunas para acelerar o acesso as linhas:
   1. Criando um índice:
      * CREATE INDEX nomeIndice ON *nomeTabela* (coluna1)
   2. Índice composto:
      * CREATE INDEX nomeIndice ON *nomeTabela* (coluna1, coluna2)

Removendo um índice:

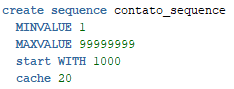
* DROP INDEX nomeIndice;

**Conteúdo extra:**

**Vídeo:** https://web.microsoftstream.com/video/1ba8c231-4bdf-4932-8487-339563b16866

* Ensina a definir valores DEFAULT já no modelo relacional antes de criar o script
* Usar Constraint CHECK para diferenciar entre conta PJ e PF
* Gerar automaticamente o ID e quais parâmetros pra geração de ID
* Configurar o ARCO (Especialização) entre a tabela pessoa física e pessoa jurídica, e definir a condição que segue para cada tabela

**Aula de java – COMO CRIAR SEQUENCE**

cache gera 20 números e mantem na memória para agilizar, dai gerando de 20 em 20

**DML – Linguagem de manipulação de dados**

Instruções DML:

**INSERINDO NA TABELA**

Comando 🡪 INSERT INTO *nomeTabela* (coluna1, coluna2) VALUES (300, ‘Teste’);

**ATUALIZANDO LINHAS NA TABELA**

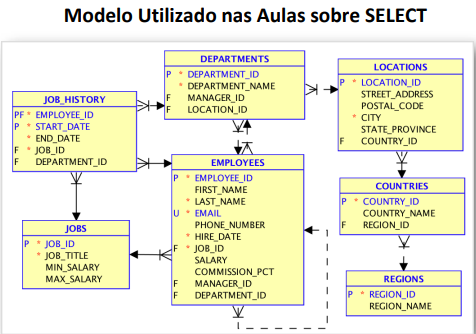
Comando 🡪 UPDATE *nomeTabela* SET coluna2 = ‘teste’ WHERE coluna1 = 300;

**APAGANDO LINHAS NA TABELA**

Comando 🡪 DELETE *nomeTabela* WHERE coluna1 = 300;

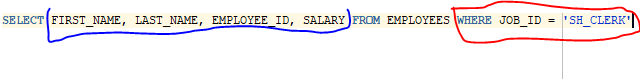
**Transações de Banco de Dados**

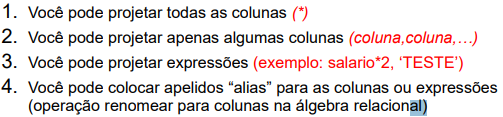
* Uma transação consiste em um conjunto de instruções DML que formam uma unidade lógica de trabalho
* Uma transação pode iniciar automaticamente, quando o primeiro comando DML é executado ou então explicitamente, como no caso do SQL Server (begin transaction)
* Uma transação termina da seguinte maneira:
  + COMMIT: Efetiva todas as alterações feitas pela transação
  + ROLLBACK: Desfaz todas as alterações feitas pela transação

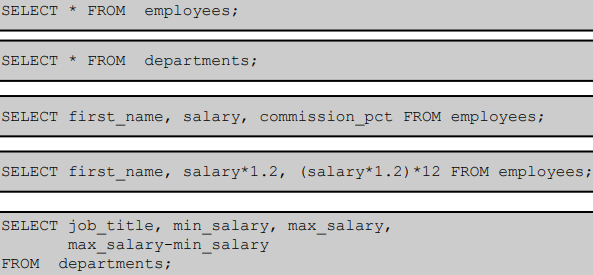


**COMANDOS DQL - Queries**

**Q** vem de QUERY, o único comando DQL é 🡪 SELECT

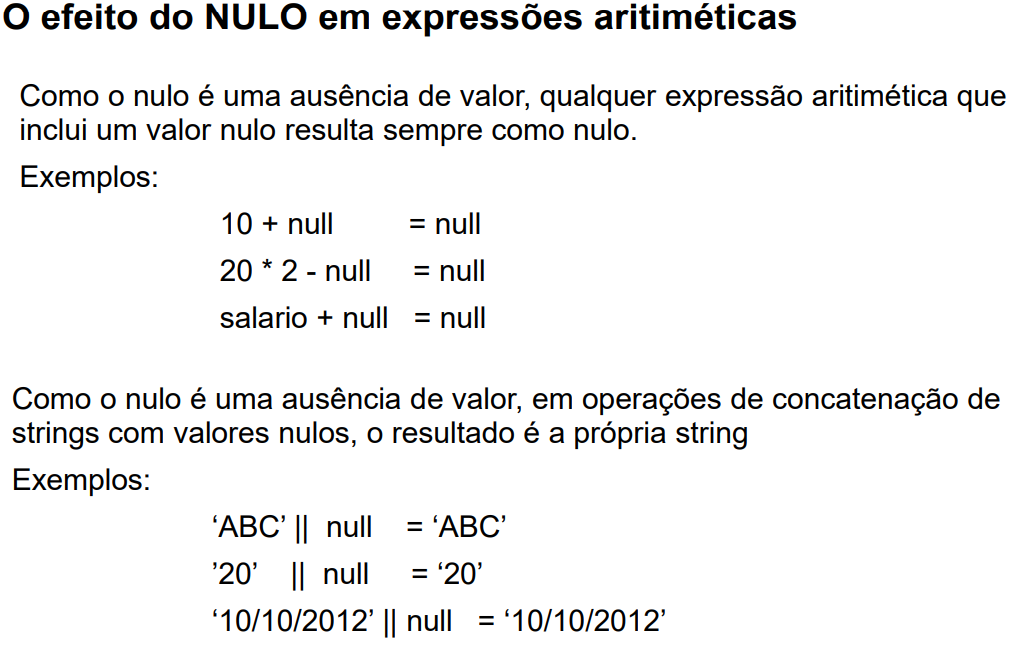






**Concatenação de colunas**



**Entendendo o NULO**

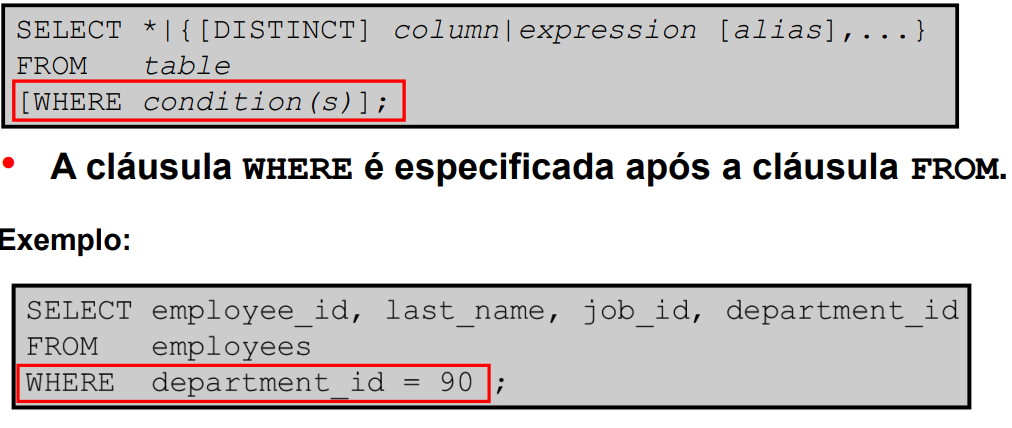
Nulo é ausência de valor. É diferente de zero ou espaço em branco.

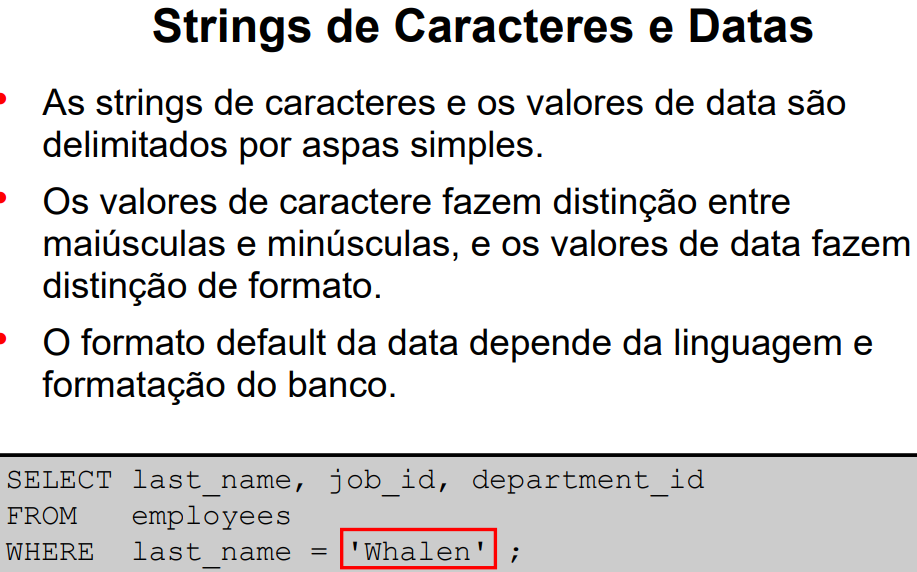
**Operador DISTINCT**

A exibição normal de consultas mostra todas as linhas, inclusive as duplicadas. Quando colocamos DISTINCT ele elimina as linhas duplicadas.

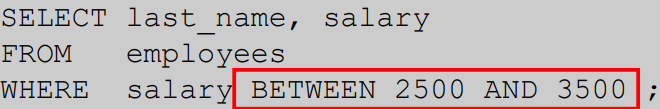
* SELECT DISTINCT *nomeTabela* FROM *coluna1*;

**Restringindo usando WHERE**



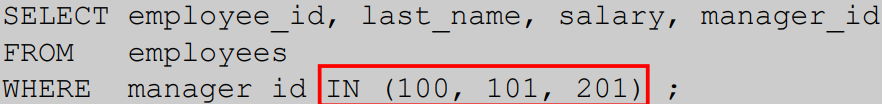


**Operador BETWEEN**

Use a condição **BETWEEN** para exibir linhas com base em uma faixa de valores.

**Operador IN**

Use a condição de associação **IN** para testar os valores de uma lista.

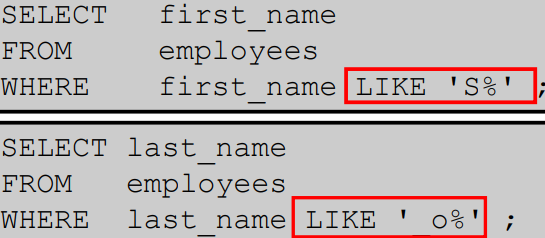


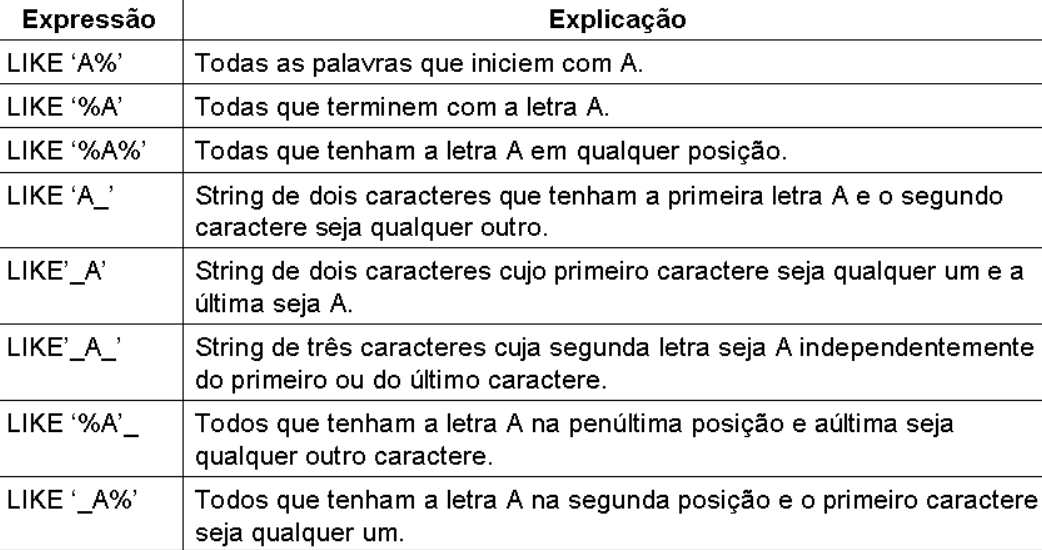
**Operador LIKE**

Use a condição **LIKE** para executar pesquisas com curinga de valores válidos de Strings de pesquisa.

As condições de pesquisa podem conter números ou caracteres literais:

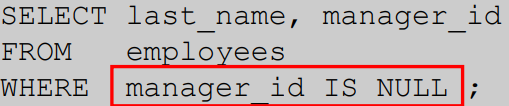
* % indica zero ou vários caracteres.
* \_ indica um caractere.



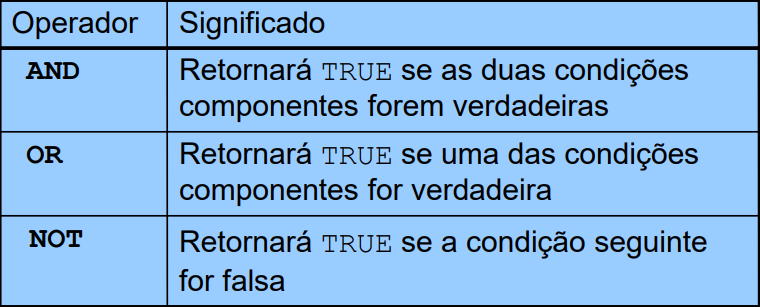


**Operador IS NULL**

Teste valores nulos com o operador **IS NULL**.

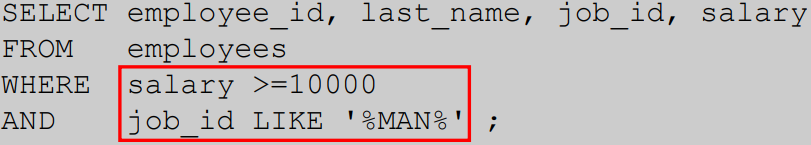


**Operadores Lógicos**

****

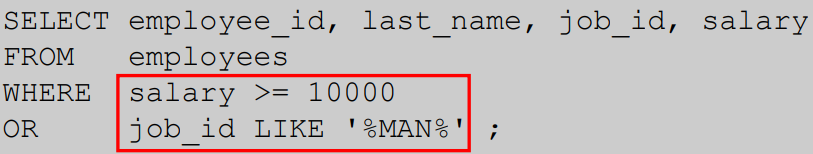
**Operador AND**

**AND** exige que as duas condições sejam verdadeiras:



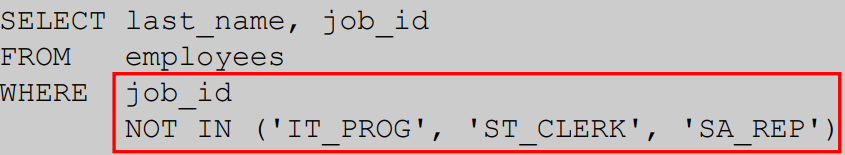
**Operador OR**

**OR** exige que uma das condições sejam verdadeiras:



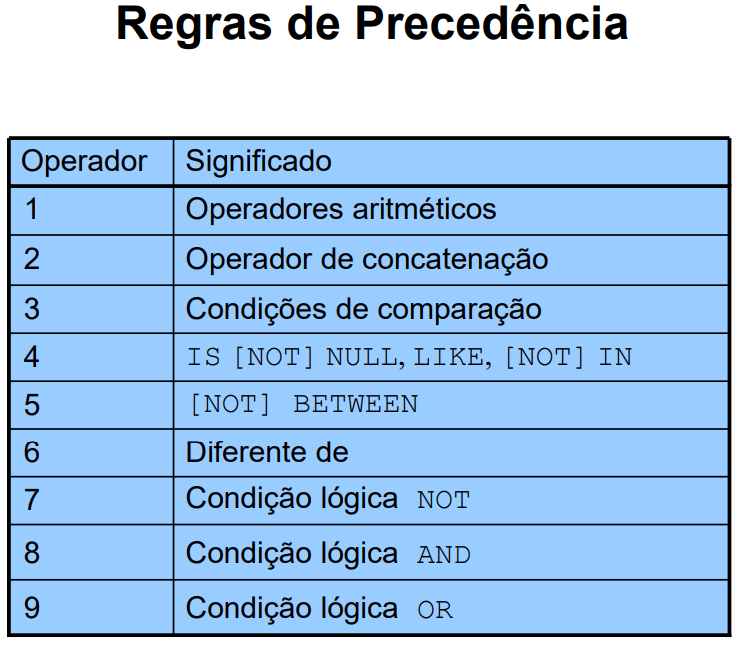
**Operador NOT**

**NOT** nega as condições, ou seja, retorna o ao contrário.



**Ordem do que é executado primeiro**

Podemos usar parênteses para sobrepor as regras de precedência.

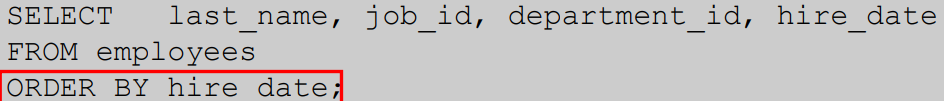
****

**Ordenando Resultados 🡪 ORDER BY**

Ordenar as linhas recuperadas com a cláusula **ORDER BY:**

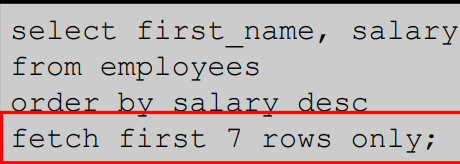
* ASC = ordem crescente, **default**
* DESC = ordem decrescente

A cláusula **ORDER BY** é inserida por último na instrução **SELECT:**

****

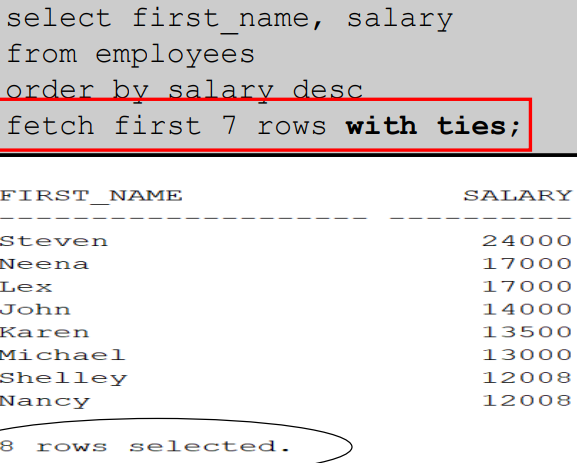
**Top N linhas**

“FETCH” 🡪 ENTREGAR

****Retorna as primeiras **N** linhas.

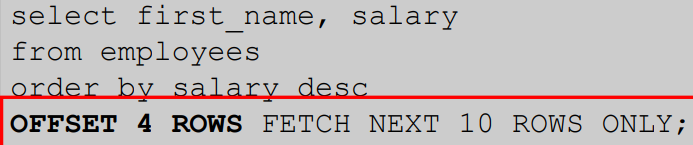
**Top N linhas repetindo o ultimo valor**

Retorna os top **N** campos, mostrando os valores referentes ao **último valor**

****

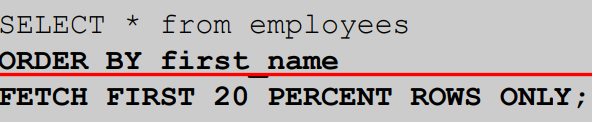
**Top N linhas ignorando X primeiras linhas**

Retorna os top **N** campos menores que os **X** primeiros.

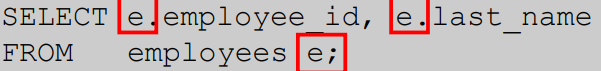


**Top X% de linhas**

Retorna **X%** das linhas da tabela.



**Apelidos de tabelas**

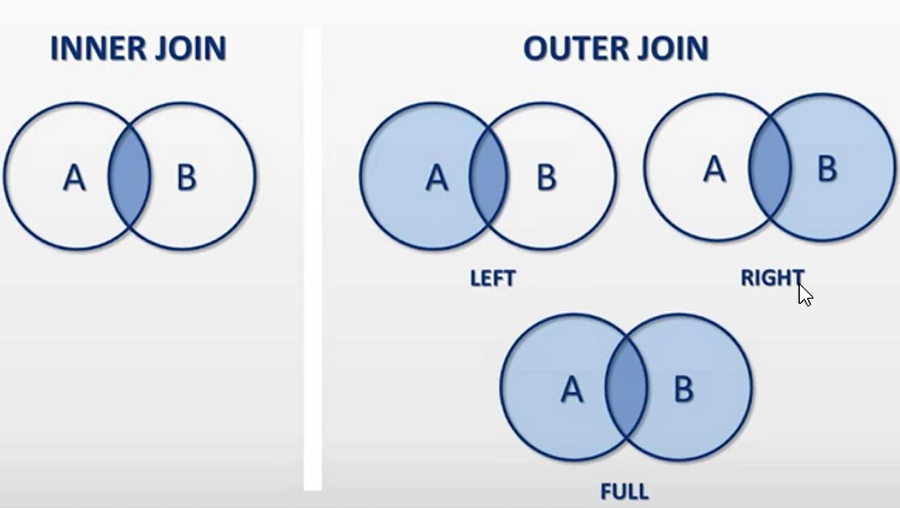
Podemos usar prefixos de tabela para qualificar nomes de colunas presentes em várias tabelas

**JOIN (JUNÇÃO)**

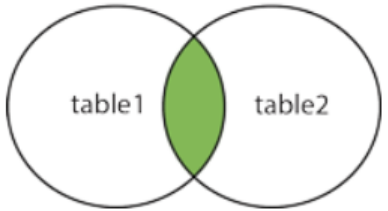
Quando se precisa projetar dados de mais de uma tabela, é necessária uma operação de junção entre as tabelas. Toda operação de **JOIN** envolve duas tabelas.

Tipos de **JOINS:**

* INNER JOIN (junções internas)
* OUTER JOIN (junções externas)
* CROSS JOIN (produtos cartesianos)

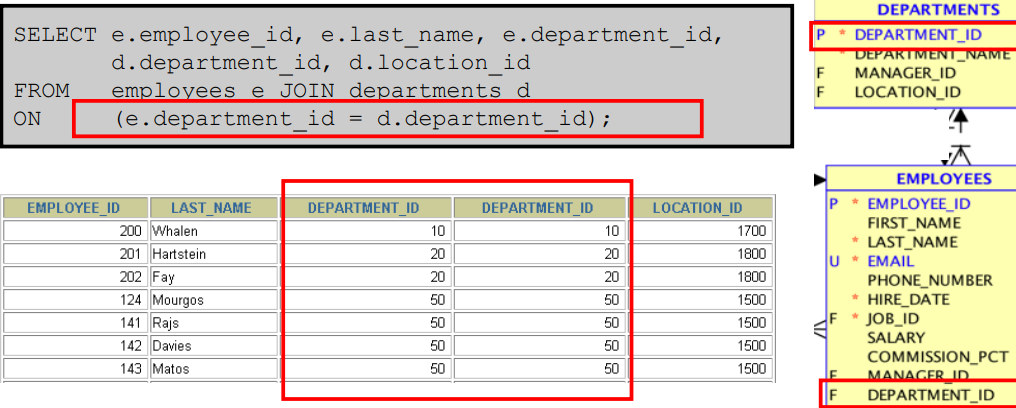


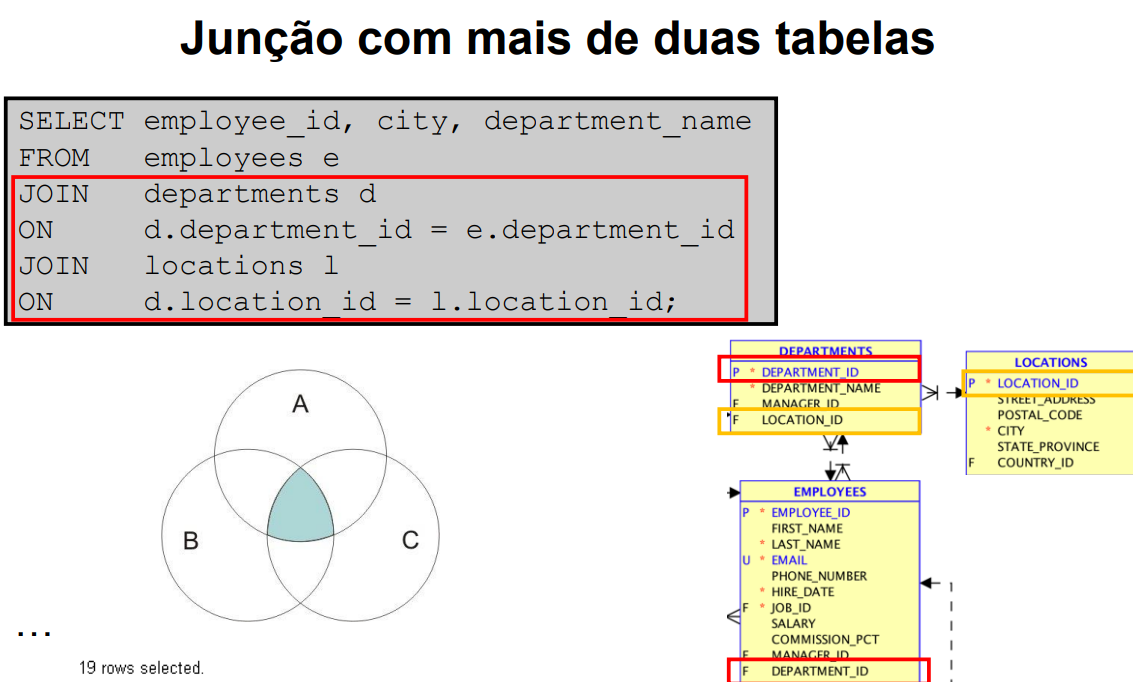
**INNER JOIN**

São usadas quando se deseja unir apenas as linhas que existem correspondência nas duas tabelas envolvidas na junção.

Um **INNER JOIN** precisa garantir que linhas das duas tabelas representem a mesma informação (normalmente **PKs** e **FKs**).

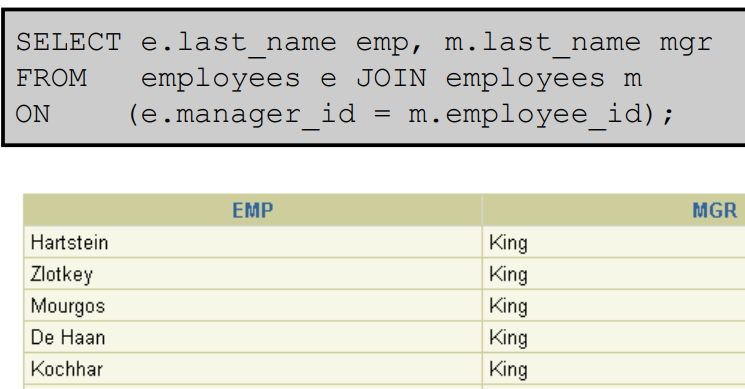
Usamos a clausula **ON** para especificar condições de **JUNÇÕES** a serem utilizadas em operações de **JOIN.**





**Self-JOIN**

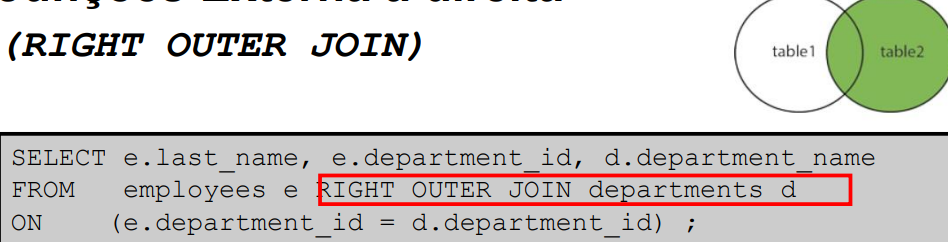
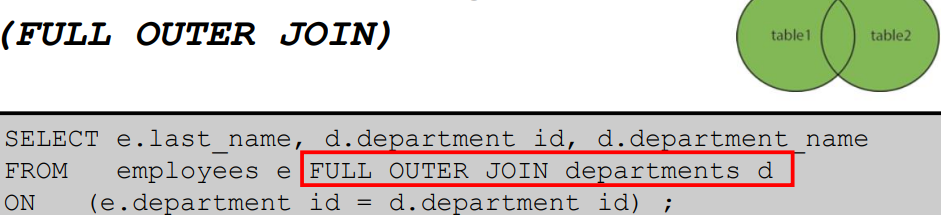
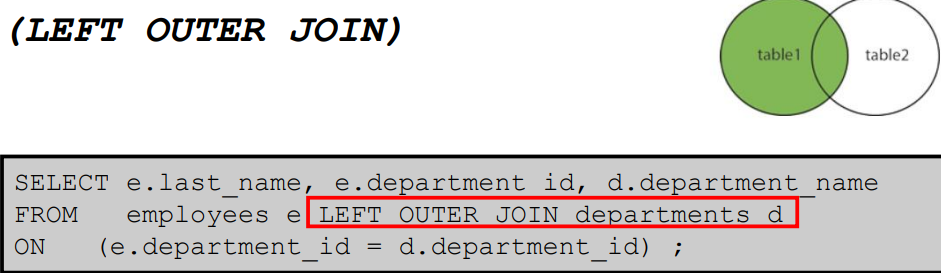
No caso de auto-relação.



**OUTER JOIN**

Temos um **JOIN** que retorna os resultados do INNER JOIN + as linhas correspondentes da tabela esquerda ou direita.

Temos também um **JOIN** que retorna os resultados do INNER JOIN + os resultados de uma JOIN esquerda e direita, chamada de **junção interna integral.**

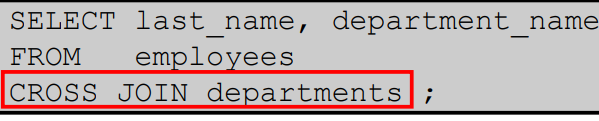


**Produtos cartesianos**

Um produto cartesiano será formado quando:

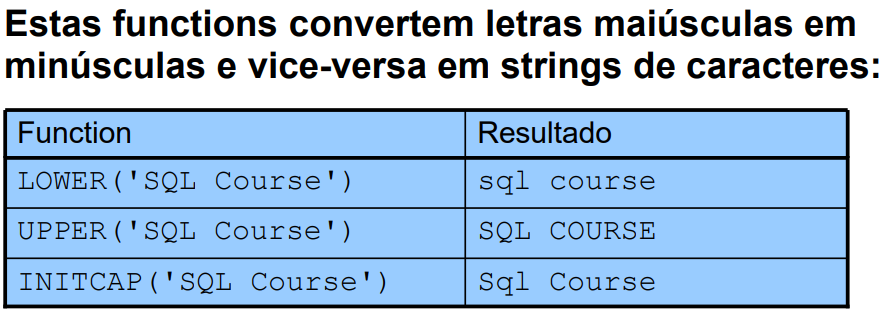
* Uma condição de **JOIN** for omitida
* Uma condição de **JOIN** for inválida
* Todas as linhas da primeira tabela se unirem a todas as linhas da segunda tabela

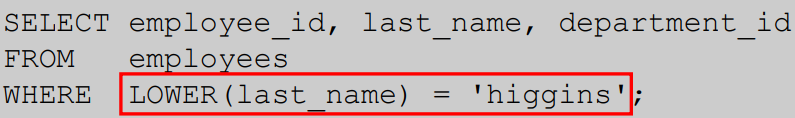
Para evitar um produto cartesiano, inclua sempre uma condição de **join** válida.



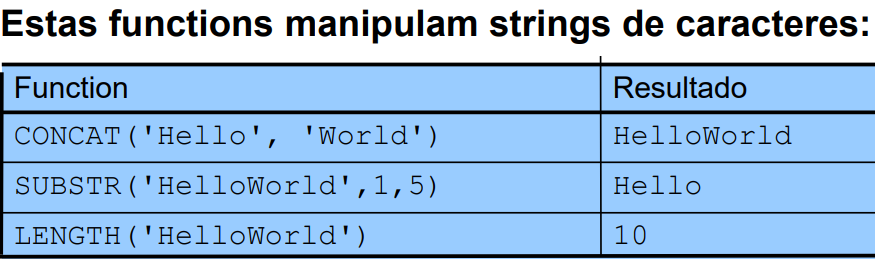
**FUNÇÕES SQL de uma única linha**

**Funções aplicadas a String**





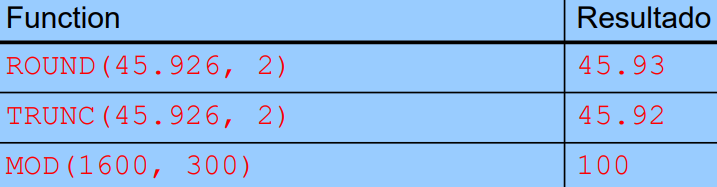
Exemplo2: SELECT UPPER(first\_name) from employees where id = 0;



SUBSTR legenda 🡪 primeiro é a string que vamos usar, segundo valor é a partir de qual posição de caractere, e terceiro valor diz quantos caracteres a partir daquela posição ele vai pegar. Se quiser pegar todos os caracteres a partir do caractere, apenas por a string e a posição inicial. Ex: (‘HelloWorld’, 1)

Temos a **INSTR** que retorna quantos caracteres específicos tem naquela string. Ex: (‘HelloWorld’, ‘o’) = 2

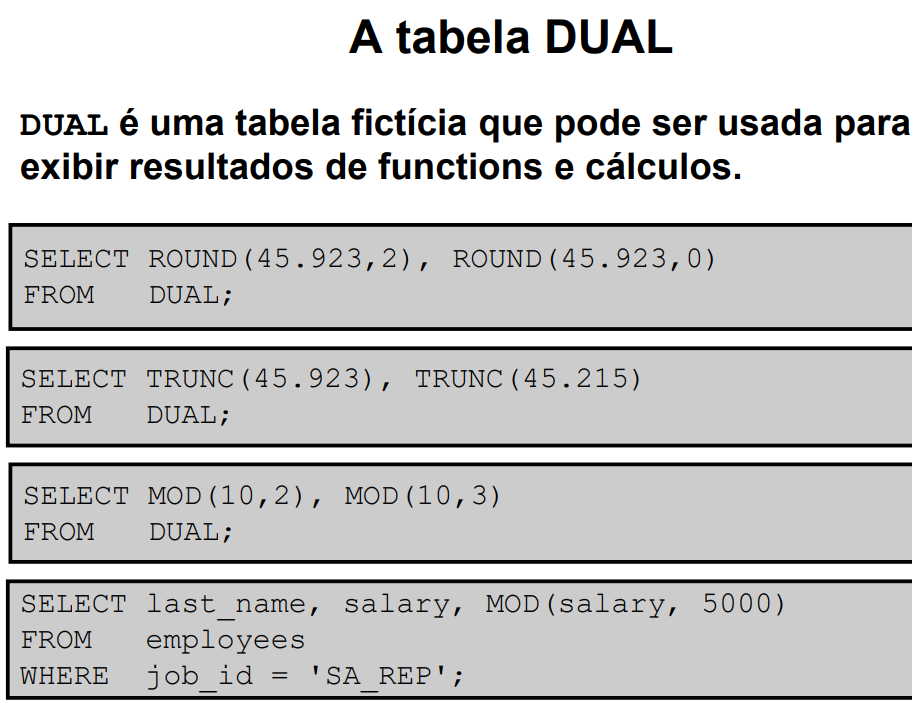
**Funções aplicada a números**



**Round :** Arredonda o valor até o decimal especificado.

**Trunc :** Trunca o valor até o decimal especificado.

**Mod :** Retorna o resto da divisão.

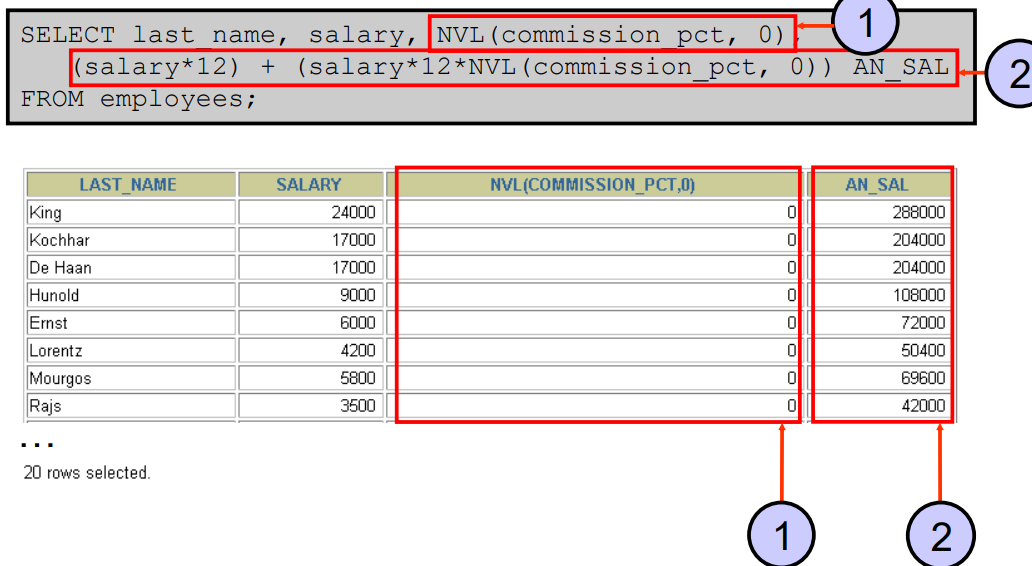


**Função NVL**

Essa função converte um valor nulo em um valor real:

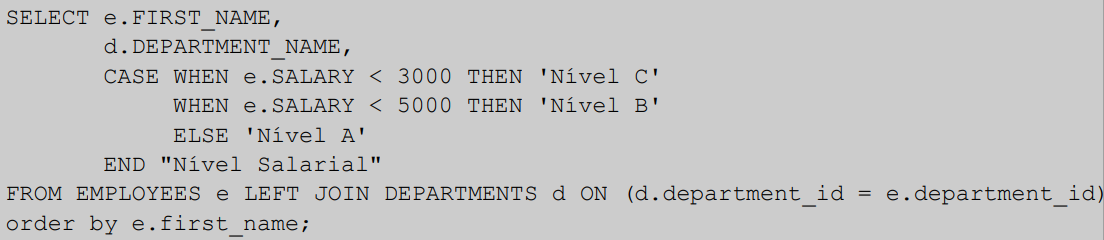
* É possível usar os tipos de dados de data, caractere e número.
* A correspondência entre os tipos de dados é necessária:
  + NVL (comission\_pct, 0)
  + NVL (hire\_date, ’01-JAN-97’)

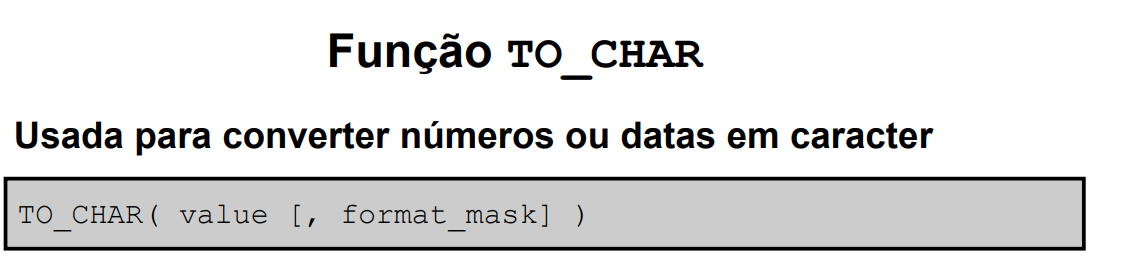
NVL (job\_id, ‘No Job Yet’)

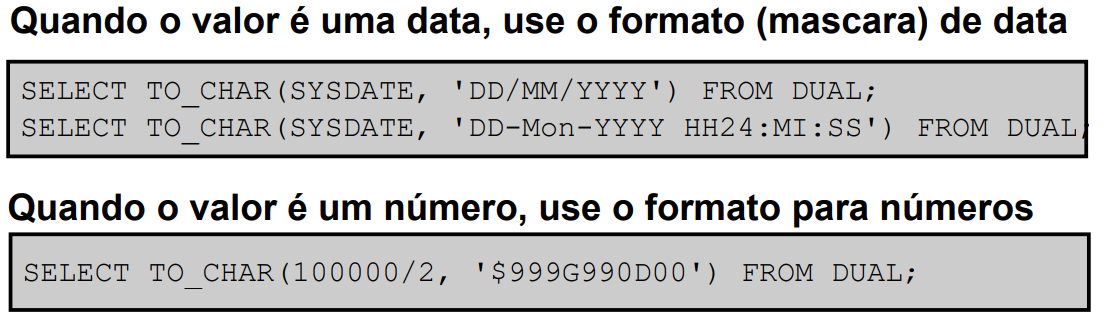


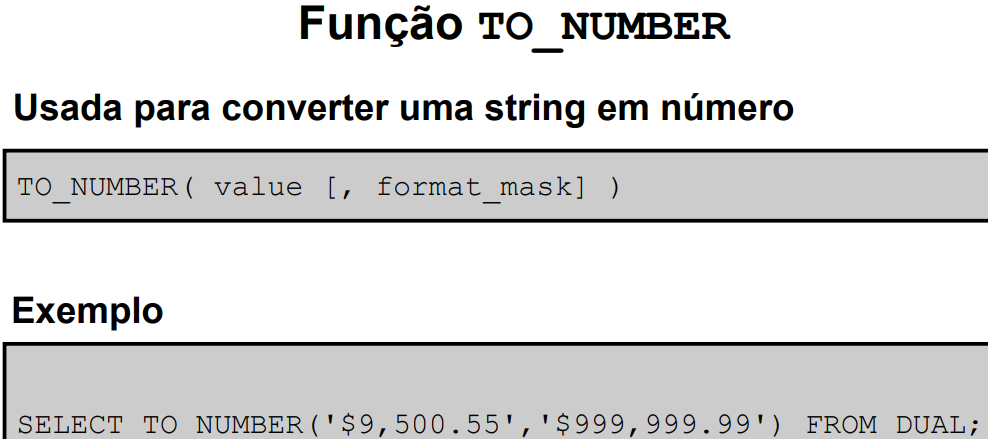
**Expressão CASE**

Tipo um IF ELSE









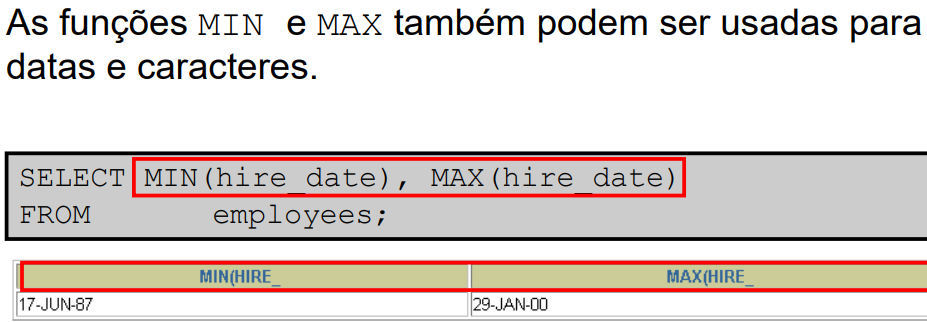
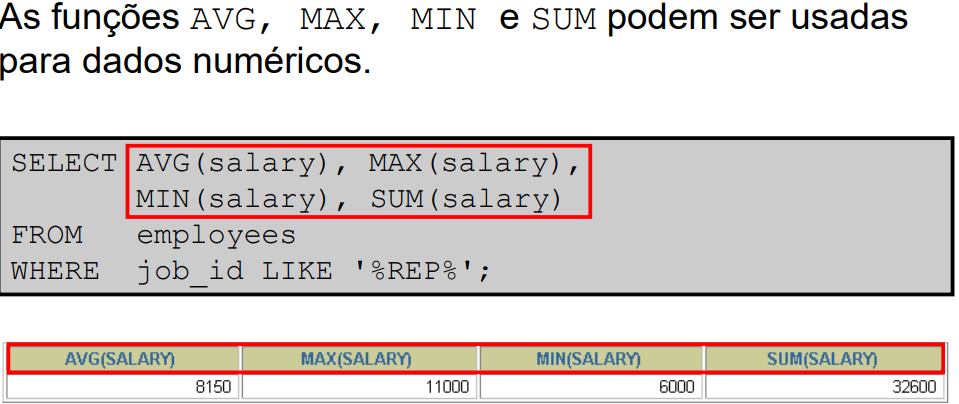


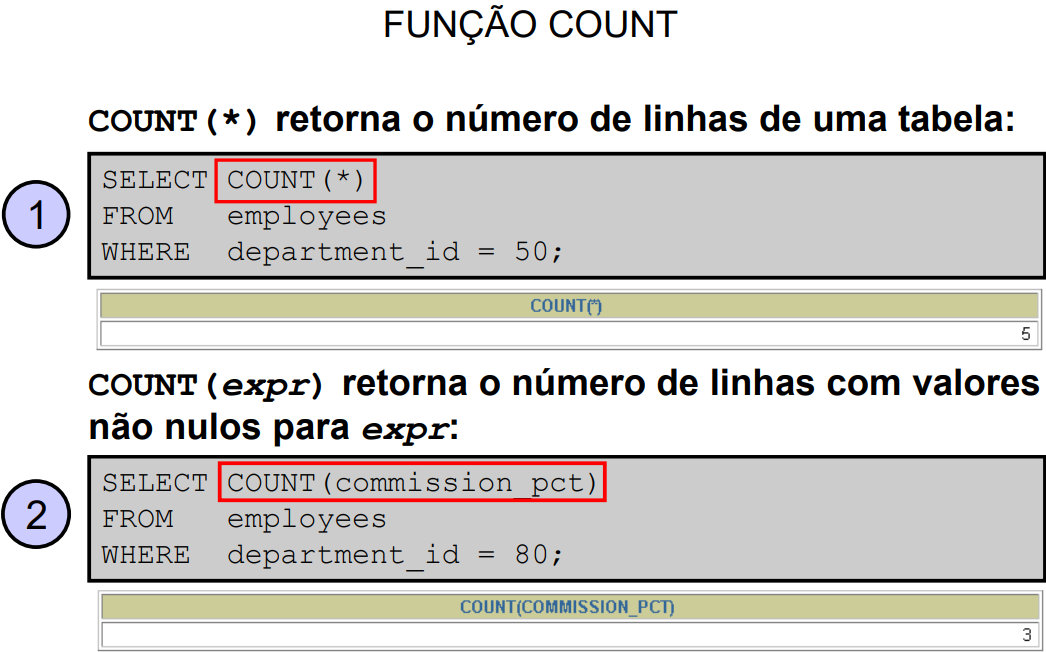
**FUNÇÕES SQL para grupo de linhas**

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

* AVG 🡪 Média
* COUNT 🡪 Totalizador (quantidade)]
* MAX 🡪 Maior Valor
* MIN 🡪 Menor Valor
* SUM 🡪 Somatório dos valores
* Entre outras...

Obs: Essas funções ignoram valores nulos. Se for necessário pode usar a função NVL





**Agrupar resultados**

