



# Consulta com uma tabela no PostgreSQL

Você vai aprender a criar comandos SQL usando expressões no comando SELECT, além de aprender a especificar condições na cláusula WHERE e a trabalhar com dados agrupados. Essas habilidades são importantes para projetar consultas em sistemas gerenciadores de banco de dados (SGBD) e são fundamentais para o dia a dia de programadores, analistas de sistemas e desenvolvedores.

Prof. Sidney Venturi    Profa. Nathielly de Souza Campos

### Propósito

Antes de iniciar este conteúdo, certifique-se de ter baixado e instalado o SGBD PostgreSQL em seu computador.

[Clique aqui](#) para baixar o arquivo com todos os códigos que serão utilizados nas consultas dos módulos a seguir.

### Objetivos

- Aplicar consultas com o comando SELECT.
- Aplicar consultas com as cláusulas where e order by.
- Aplicar consultas envolvendo agrupamento de dados.

### Introdução

Ao longo deste material, vamos explorar diversos exemplos de consultas envolvendo uma tabela. Aprenderemos a codificar consultas abrangendo tanto a recuperação de colunas da própria tabela quanto o uso de expressões no comando SELECT. Quando projetamos um banco de dados para determinado domínio de negócio, em geral, são criadas diversas tabelas que serão manipuladas pelas aplicações desenvolvidas para acessar os recursos do banco de dados.

Diversas operações que manipulam tabelas em um banco de dados necessariamente estão associadas a alguma operação de consulta. Por exemplo, se resolvermos aumentar em 10% o salário de todos os funcionários que ganham até R\$ 4.000, será necessário programarmos um comando de consulta para que o sistema gerenciador de banco de dados (SGBD) selecione os registros dos funcionários que são alvo da atualização. Assim, aprender de maneira efetiva a programar consultas trará benefícios, tanto para atividades de construção de relatórios quanto para o projeto de operações de remoção e atualização de dados.

No vídeo a seguir, vamos introduzir os conceitos iniciais da consulta em uma tabela no PostgreSQL.



#### Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

# Estrutura básica de uma comando SELECT

O comando SELECT básico foca habilidades fundamentais para manipular dados em bancos de dados. Aprenderemos a criar consultas SQL com a cláusula SELECT para selecionar informações específicas. Vamos lá!

Neste vídeo, vamos explorar o modelo relacional, que é a base estrutural de bancos de dados relacionais, e discutir os componentes essenciais de uma tabela, incluindo campos, chaves primárias e estrangeiras, além de exemplos práticos de como esses elementos são utilizados na modelagem de dados.



### Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

O comando SELECT é usado para exibir dados resultantes de uma consulta. Os dados podem ser colunas físicas de uma tabela, colunas calculadas ou mesmo resultado do uso de expressões e funções. Uma sintaxe básica para o comando SELECT está expressa a seguir. Veja com mais detalhes!

```
sql
SELECT COLUNA1 [[AS] APELIDOCOLUNA1],
       COLUNA2 [[AS] APELIDOCOLUNA2],
       ...
       COLUNAN [[AS] APELIDOCOLUNAN]
FROM TABELA;
```

Estamos diante de uma sintaxe simplificada o suficiente para entendimento dos exemplos que iremos explorar. A sintaxe completa abrange todos os recursos do PostgreSQL.

Uma sintaxe complexa envolve uma série de cláusulas e recursos bastante úteis para consultas de maior complexidade.

Na prática, o comando SELECT, dependendo da consulta desejada, pode ser usado de diferentes formas para obter o mesmo resultado. É importante frisar que a cláusula SELECT realiza a operação de projeção da álgebra relacional.

Caso haja interesse em exibir todas as colunas especificadas em uma consulta, basta adicionar um "\*", da seguinte forma: SELECT \* FROM TABELA;



### Saiba mais

Alguns SGBDs, como o PostgreSQL, implementam uma forma simplificada do comando `SELECT * FROM TABELA`, que é simplesmente `TABLE tabela` (você pode testar isso no PostgreSQL).

Vamos estudar alguns exemplos?

Construiremos as consultas com base na tabela ALUNO, conforme imagem a seguir.

ALUNO		
CODIGOALUNO	int	PK
NOME	varchar(90)	
SEXO	char(1)	
DTNASCIMENTO	date	
EMAIL	varchar(50)	N

Tabela ALUNO.

Recomendamos que você crie a tabela e insira algumas linhas, o que pode ser feito usando o script a seguir, a partir da ferramenta de sua preferência. Para isso, tenha em mente que é necessário estar conectado ao PostgreSQL e acessando algum database criado por você. Veja com mais detalhes!

sql

```
CREATE TABLE ALUNO (  
    CODIGOALUNO int NOT NULL,  
    NOME varchar(90) NOT NULL,  
    SEXO char(1) NOT NULL,  
    DTNASCIMENTO date NOT NULL,  
    EMAIL varchar(30) NULL,  
    CONSTRAINT ALUNO_pk PRIMARY KEY (CODIGOALUNO));  
INSERT INTO ALUNO (CODIGOALUNO, NOME, SEXO, DTNASCIMENTO, EMAIL) VALUES (1, 'JOSÉ  
FRANCISCO TERRA', 'M', '28/10/1989', 'JFT@GMAIL.COM');  
INSERT INTO ALUNO (CODIGOALUNO, NOME, SEXO, DTNASCIMENTO, EMAIL) VALUES (2, 'ANDREY COSTA  
FILHO', 'M', '20/10/1999', 'ANDREYCF@HOTMAIL.COM');  
INSERT INTO ALUNO (CODIGOALUNO, NOME, SEXO, DTNASCIMENTO, EMAIL) VALUES (3, 'PATRICIA  
TORRES LOUREIRO', 'F', '20/10/1980', 'PTORRES@GMAIL.COM');  
INSERT INTO ALUNO (CODIGOALUNO, NOME, SEXO, DTNASCIMENTO, EMAIL) VALUES (4, 'CARLA MARIA  
MACIEL', 'F', '20/11/1996', NULL);  
INSERT INTO ALUNO (CODIGOALUNO, NOME, SEXO, DTNASCIMENTO, EMAIL) VALUES (5, 'LEILA SANTANA  
COSTA', 'F', '20/11/2001', NULL);
```

Vamos ver alguns exemplos de consultas?

### Consulta 01

Exibir todas as informações dos alunos.

```
SELECT * FROM ALUNO;
```

A tabela a seguir apresenta os resultados da consulta.

Resultados da consulta 01.



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

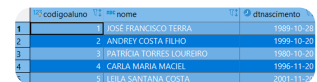
Ao executar a consulta, o SGBD percorre todos os registros da tabela ALUNO e exibe as colunas dessa tabela.

### Consulta 02

Retornar o código, o nome e a data de nascimento de todos os alunos.

```
SELECT CODIGOALUNO, NOME, DTNASCIMENTO FROM ALUNO;
```

Resultados da consulta 02.



1	2	3
1	JOSÉ FRANCISCO TERRA	1989-10-28
2	ANDREY COSTA FILHO	1999-10-20
3	PATRICIA TORRES LOUREIRO	1980-10-20
4	CARLA MARIA MACIEL	1996-11-20
5	LEILA SANTANA COSTA	2001-11-20

Na consulta 02, foram especificadas três colunas da tabela ALUNO para serem exibidas ao usuário.

Em especial, pode ser interessante renomear as colunas resultantes da consulta, visando tornar os resultados mais apresentáveis ao usuário da aplicação. Por exemplo, a consulta 02 pode ser reescrita da seguinte forma:

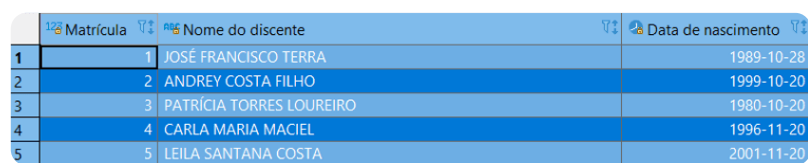
```
sql
```

```
SELECT CODIGOALUNO AS "Matrícula", NOME AS "Nome do discente",
```

```
DTNASCIMENTO AS "Data de nascimento"
```

```
FROM ALUNO;
```

O resultado dessa consulta seria este:



	Matrícula	Nome do discente	Data de nascimento
1	1	JOSÉ FRANCISCO TERRA	1989-10-28
2	2	ANDREY COSTA FILHO	1999-10-20
3	3	PATRICIA TORRES LOUREIRO	1980-10-20
4	4	CARLA MARIA MACIEL	1996-11-20
5	5	LEILA SANTANA COSTA	2001-11-20

Resultados da segunda versão da consulta 02.

Na tabela anterior, o nome apresentado para cada coluna não existe fisicamente no banco de dados.

Vamos aprender que nem toda coluna resultante de uma consulta representa necessariamente uma coluna de alguma tabela.

# Atividade 1

A imagem a seguir nos mostra o conteúdo da tabela empregado em um SGBD PostgreSQL.

Data Output Messages Notifications										
	id	ult_nome	prim_nome	cargo	salario	dt_admissao	cpf	id_depto	id_gerente	
	[PK] numeric (7)	character varying (20)	character varying (20)	character varying (30)	numeric (7,2)	date	character	numeric (7)	numeric (7)	
1	1	Velasques	Carmen	Presidente	36400.00	2009-05-05	34567890125	10		
2	2	Neves	Lauro	Diretor de Compras	24400.00	2009-03-03	23456789012	30	1	
3	3	Nogueira	Emane	Diretor de Vendas	22600.00	2010-04-07	34567890123	20	1	
4	4	Queiroz	Mark	Gerente de Compras	10600.00	2010-11-11	12345432123	30	2	
5	5	Rodrigues	Alberto	Vendedor	5800.00	2008-10-10	87965432123	20	3	
6	6	Ugarte	Marlene	Vendedor	7200.00	2009-03-03	87654345678	20	3	

Tabela empregado.

Qual comando retornaria apenas o primeiro nome e o último nome de todos os empregados, nessa ordem?

A

SELECT ID, PRIM\_NOME , ULT\_NOME FROM EMPREGADO

B

SELECT \* FROM EMPREGADO

C

SELECT PRIM\_NOME , ULT\_NOME FROM EMPREGADO

D

SELECT ULT\_NOME , PRIM\_NOME FROM EMPREGADO

E

SELECT PRIM\_NOME + ULT\_NOME FROM EMPREGADO



A alternativa C está correta.

Na cláusula SELECT, devemos escrever o nome das colunas que desejamos, em ordem correta e separada por vírgula.

## Funções de data e hora

Vamos agora estudar as funções de data e hora, explorando como trabalhar esses conceitos de forma eficiente em bancos de dados. Aprenderemos a usar funções SQL para realizar diversos cálculos entre datas, extrair partes específicas de uma data e realizar manipulações avançadas.

Essas habilidades são importantes para quem busca realizar análises temporais e gerenciar dados sensíveis ao tempo em sistemas de banco de dados.

Neste vídeo, vamos apresentar funções que envolvem dados representativos de datas.



### Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Quando desenvolvemos consultas, é comum manipularmos colunas e funções que envolvem dados representativos de datas.

Para entender melhor, observe a tabela!

Função	O que retorna?
current_date	data de hoje
current_time	hora do dia
current_timestamp	data e a hora
extract (campo from fonte)	subcampos de data e hora: século, ano, dia, mês

Tabela: Funções de data do PostgreSQL.  
Sidney Venturi.

Um quadro completo contendo informações sobre funções de data e hora pode ser encontrado na documentação oficial do PostgreSQL.

Vamos estudar alguns exemplos? Observe com atenção o código.

```
sql
```

```
SELECT CURRENT_DATE AS "Data Atual",  
       CURRENT_TIME AS "Hora Atual",  
       CURRENT_TIMESTAMP "Data e Hora atuais",  
       EXTRACT( DOY FROM CURRENT_DATE) AS "Dia do ano",  
-- DOW 0 - domingo, 1 - segunda, ..., 6 - sábado  
       EXTRACT( DOW FROM CURRENT_DATE) AS "Dia da semana",  
       EXTRACT( DAY FROM CURRENT_DATE) AS "Dia Atual",  
       EXTRACT( MONTH FROM CURRENT_DATE) AS "Mês Atual",  
       EXTRACT( YEAR FROM CURRENT_DATE) AS "Ano Atual",  
       EXTRACT( CENTURY FROM CURRENT_DATE) AS "Século Atual";
```

Agora veja na tabela os resultados da consulta!

Data Atual	Hora Atual	Data e Hora atuais	Dia do ano	Dia da semana	Dia Atual	Mês Atual	Ano Atual	Século Atual
2020-06-24	08:55:13	2020-06-24 08:55:13	176	3	24	6	2.020	21

Resultados da consulta envolvendo funções de data e hora.

Observe que utilizamos o qualificador AS “Apelido” para facilitar o entendimento do retorno de cada função. Note também que não há cláusula FROM na consulta, visto que todas as colunas retornadas representam o resultado de funções do PostgreSQL sem envolver qualquer tabela do domínio da aplicação.



### Atenção

No padrão SQL, a cláusula FROM é obrigatória. No entanto, o PostgreSQL permite executar um comando SELECT sem a cláusula FROM. Experimente executar `SELECT 5+5;`

## Exibindo o nome do dia da semana

Perceba que a linha 6 do código acima retorna um inteiro representativo do dia da semana. No entanto, se houver necessidade de exibir o dia da semana, você pode usar o código a seguir.

sql

```
SELECT CASE WHEN extract(dow from current_date) = 0 THEN 'domingo'
            WHEN extract(dow from current_date) = 1 THEN 'segunda-feira'
            WHEN extract(dow from current_date) = 2 THEN 'terça-feira'
            WHEN extract(dow from current_date) = 3 THEN 'quarta-feira'
            WHEN extract(dow from current_date) = 4 THEN 'quinta-feira'
            WHEN extract(dow from current_date) = 5 THEN 'sexta-feira'
            WHEN extract(dow from current_date)=6 THEN 'sábado'
END AS "Nome do dia da semana";
```

Observe que construímos uma lógica utilizando o comando CASE, que é equivalente ao comando IF; cada linha com a cláusula WHEN avalia expressão que retorna um inteiro representativo do dia da semana, caso a expressão tenha valor lógico verdadeiro.

## Calculando idade e faixa etária

Em geral, quando estamos diante de alguma coluna representativa da data de nascimento de uma pessoa, é comum extrair informações derivadas, tais como idade e faixa etária. Por exemplo, o código a seguir retorna o nome, a data de nascimento e a idade dos alunos. Veja!



sql

```
SELECT NOME,  
       DTNASCIMENTO,  
       AGE(DTNASCIMENTO) AS "Idade [ano/mês/dia]",  
       EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) AS "Idade do Aluno"  
FROM ALUNO;
```

	nome	dtnascimento	Idade [ano/mês/dia]	Idade do Aluno
1	JOSÉ FRANCISCO TERRA	1989-10-28	30 years 7 mons 30 days	30
2	ANDREY COSTA FILHO	1999-10-20	20 years 8 mons 7 days	20
3	PATRÍCIA TORRES LOUREIRO	1980-10-20	39 years 8 mons 7 days	39
4	CARLA MARIA MACIEL	1996-11-20	23 years 7 mons 7 days	23
5	LEILA SANTANA COSTA	2001-11-20	18 years 7 mons 7 days	18

Exibindo a idade dos alunos.

Muito bem, agora, vamos exibir o nome, a idade e a faixa etária dos alunos.

Observe o código SQL a seguir.

sql

```
SELECT NOME,  
       EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) AS "Idade do Aluno",  
       CASE WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) <=20 THEN '1. até 20 anos'  
            WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) BETWEEN 21 AND 30 THEN '2. 21 a 30 anos'  
            WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) BETWEEN 31 AND 40 THEN '3. 31 a 40 anos'  
            WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) BETWEEN 41 AND 50 THEN '4. 41 a 50 anos'  
            WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) BETWEEN 51 AND 60 THEN '5. 51 a 60 anos'  
            WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) > 60 THEN '6. mais de 60 anos'  
       END AS "Faixa Etária"  
FROM ALUNO;
```

Perceba que cada linha com a cláusula WHEN avalia a expressão que retorna uma faixa etária de acordo com a idade do aluno.

A seguir, o resultado da consulta. Confira!

	nome	Idade do Aluno	Faixa Etária
1	JOSÉ FRANCISCO TERRA	30	2. 21 a 30 anos
2	ANDREY COSTA FILHO	20	1. até 20 anos
3	PATRÍCIA TORRES LOUREIRO	39	3. 31 a 40 anos
4	CARLA MARIA MACIEL	23	2. 21 a 30 anos
5	LEILA SANTANA COSTA	18	1. até 20 anos

Resultados da consulta envolvendo idade e faixa etária dos alunos.

## Atividade 2

Considere a tabela e o código SQL a seguir.

ALUNO		
CODIGOALUNO	int	PK
NOME	varchar(90)	
SEXO	char(1)	
DTNASCIMENTO	date	

```

1 SELECT CURRENT_DATE AS "Data de Emissão do Relatório",
2     CODIGOALUNO AS "Matrícula",
3     NOME AS "Nome do discente",
4     DTNASCIMENTO AS "Data de nascimento",
5     CASE
6         WHEN SEXO='M' THEN 'Masculino'
7         WHEN SEXO='F' THEN 'Feminino'
8     END AS SEXO
9 FROM ALUNO;

```

Tabela e código SQL.

Analise as seguintes proposições:

1. A consulta retorna informações sobre cinco colunas existentes na tabela ALUNO.
2. A consulta retorna informações sobre todos os alunos cadastrados.
3. Pode existir registro com valor "Masculino" armazenado na coluna SEXO.
4. O resultado de CURRENT\_DATE (linha 1) está armazenado em uma coluna da tabela ALUNO.
5. A consulta retorna informações sobre quatro colunas existentes na tabela ALUNO.

São proposições verdadeiras:

A

I e II.

B

II e V.

C

II, III e IV.

D

III e V.

E

II, III e V



A alternativa B está correta.

A proposição II é verdadeira, pois não há condição de filtro na consulta. A proposição V é verdadeira, pois retorna informações a respeito de todas as colunas da tabela ALUNO. As demais proposições são falsas.

## Funções de resumo ou de agregação

Vamos nos aprofundar nas técnicas essenciais para resumir e analisar grandes conjuntos de dados em bancos de dados. Abordaremos funções como SUM, AVG, COUNT, entre outras, para calcular estatísticas importantes e extrair insights valiosos.

Esse conhecimento é fundamental para profissionais que buscam criar relatórios e análises robustas em sistemas de gerenciamento de banco de dados.

Neste vídeo, vamos compreender a utilidade da aplicação das funções de resumo ou de agregação na obtenção de resumos de dados de alguma tabela.



### Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

As funções a seguir são úteis para obtermos resumo dos dados de alguma tabela.

Função	O que retorna?
COUNT(*)	número de linhas da consulta
MIN(COLUNA/EXPRESSÃO)	menor valor de uma coluna ou expressão
AVG(COLUNA/EXPRESSÃO)	valor médio da coluna ou expressão
MAX(COLUNA/EXPRESSÃO)	maior valor de uma coluna ou expressão
SUM(COLUNA/EXPRESSÃO)	soma dos valores de uma coluna ou expressão
STDDEV(COLUNA/EXPRESSÃO)	desvio-padrão dos valores de uma coluna ou expressão
VARIANCE(COLUNA/EXPRESSÃO)	variância dos valores de uma coluna ou expressão

Tabela: Funções para resumo de dados.  
Sidney Venturi.

Vamos estudar um exemplo?

Observe o código a seguir.

sql

```
SELECT
    COUNT(*) AS "Número de alunos",
    MIN(EXTRACT(YEAR FROM AGE(DTNASCIMENTO))) AS "Menor Idade",
    AVG(EXTRACT(YEAR FROM AGE(DTNASCIMENTO))) AS "Idade Média",
    MAX(EXTRACT(YEAR FROM AGE(DTNASCIMENTO))) AS "Maior Idade",
    SUM(EXTRACT(YEAR FROM AGE(DTNASCIMENTO)))/COUNT(*) AS "Idade Média"
FROM ALUNO;
```

Perceba que, como estamos usando somente o comando SELECT/FROM, cada função é calculada levando em consideração todos os registros da tabela.

Veja na imagem a seguir o resultado da consulta.

	Número de alunos	Menor Idade	Idade Média	Maior Idade	Idade Média
1	5	18	26	39	26

Resultado da consulta envolvendo funções para extrair resumo a partir da tabela aluno.

Perceba, também, que o código da linha 6 é equivalente ao da linha 4: ambos calculam a idade média dos alunos.

## Listando resumos em uma linha

Suponha que haja interesse em conhecer os quantitativos de cursos, disciplinas e alunos do nosso banco de dados. Poderíamos submeter ao SGBD as consultas a seguir.

Curso

```
SELECT COUNT(*) NCURSOS FROM CURSO;
```

Disciplina

```
SELECT COUNT(*) NDISCIPLINAS FROM
DISCIPLINA;
```

Aluno

```
SELECT COUNT(*) NALUNOS FROM ALUNO;
```

Estamos diante de três consultas. No entanto, pode ser mais interessante mostrarmos os resultados em apenas uma linha. Podemos, então, submeter o seguinte código:

sql

```
SELECT
  (SELECT COUNT(*) NCURSOS FROM CURSO),
  (SELECT COUNT(*) NALUNOS FROM ALUNO),
  (SELECT COUNT(*) NDISCIPLINAS FROM DISCIPLINA);
```

O que fizemos?

Como cada consulta (linhas 2 a 4) retorna somente um valor, utilizamos um SELECT externo (linha 1) para exibir cada coluna resultante.

Observe o resultado!

	123 ncursos 🔒 🔍 ⬆ ⬆	123 nalunos 🔒 🔍 ⬆ ⬆	123 ndisciplinas 🔒 🔍 ⬆ ⬆
1	4	5	4

Resultado da consulta envolvendo quantitativos de cursos, alunos e disciplinas.

Convém ressaltar que o comando é válido, visto que, no PostgreSQL, a cláusula FROM não é obrigatória.

## Atividade 3

### Questão 1

Seja uma tabela assim definida: FUNCIONARIO (IDFUNC, NOME, DATANASCIMENTO, SALARIO). Qual consulta SQL retorna o maior salário?

A

SELECT > SALARIO FROM FUNCIONARIO.

B

SELECT MAX(SALARIO) FROM FUNCIONARIO.

C

SELECT AVG(SALARIO) FROM FUNCIONARIO.

D

SELECT FUNCIONARIO FROM SALÁRIO.

E

SELECT SUM(SALARIO) FROM FUNCIONARIO.



A alternativa B está correta.

Na alternativa B, foi usado o comando MAX para retornar o maior valor da coluna SALÁRIO da tabela FUNCIONARIO.

## Criando tabela e view a partir de consulta

No núcleo de criação de tabelas e visões a partir de comandos SELECT, exploraremos como projetar estruturas de dados eficientes e criar visualizações personalizadas em bancos de dados.

Aprenderemos a utilizar comandos SELECT para criar tabelas e visões que atendam às necessidades específicas do nosso projeto. Esse conhecimento é essencial para construir bases sólidas de dados e facilitar análises avançadas. Então, mãos à obra!

Neste vídeo, vamos explorar o processo de criação de uma tabela e de view a partir de uma consulta SQL. Veremos como definir estruturas de dados, organizar informações e utilizar consultas para criar visualizações úteis e acessíveis no contexto de bancos de dados.



### Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

## Criando tabela a partir de consulta

Em alguns momentos, você terá interesse em salvar os resultados de uma consulta em uma nova tabela.

Para isso, basta usar o comando CREATE TABLE < CONSULTA >.

sql

```
CREATE TABLE TTESTE AS
SELECT NOME,
       EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) AS "Idade do Aluno",
       CASE WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) <=20 THEN '1. até 20 anos'
            WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) BETWEEN 21 AND 30 THEN '2. 21 a 30 anos'
            WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) BETWEEN 31 AND 40 THEN '3. 31 a 40 anos'
            WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) BETWEEN 41 AND 50 THEN '4. 41 a 50 anos'
            WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) BETWEEN 51 AND 60 THEN '5. 51 a 60 anos'
            WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) > 60 THEN '6. mais de 60 anos'
       END AS "Faixa Etária"
FROM ALUNO;
```

No exemplo apresentado, o SGBD criará uma tabela denominada TTESTE e armazenará os dados resultantes da consulta (linhas 2 a 11) em questão.

## Criando view a partir de consulta

Outro recurso interessante, diretamente relacionado ao processo de construção de consultas, é o objeto view (visão). Uma view encapsula a complexidade da consulta SQL, que a forma. Para criar esse objeto, usa-se o comando CREATE VIEW < CONSULTA >. Veja com mais detalhes!

sql

```
CREATE VIEW VTESTE AS
SELECT NOME,
       EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) AS "Idade do Aluno",
       CASE WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) <=20 THEN '1. até 20 anos'
            WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) BETWEEN 21 AND 30 THEN '2. 21 a 30 anos'
            WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) BETWEEN 31 AND 40 THEN '3. 31 a 40 anos'
            WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) BETWEEN 41 AND 50 THEN '4. 41 a 50 anos'
            WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) BETWEEN 51 AND 60 THEN '5. 51 a 60 anos'
            WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) > 60 THEN '6. mais de 60 anos'
       END AS "Faixa Etária"
FROM ALUNO;
```

No exemplo, o SGBD criará uma view denominada VTESTE. Na prática, quando usuário submeter, por exemplo, a consulta `SELECT * FROM VTESTE`, o SGBD executará o código associado à view em questão.

## Atividade 4

Imagine que você é um administrador de banco de dados responsável por gerenciar uma loja on-line e precisa otimizar a consulta frequente que mostra o total de vendas de cada produto por mês para facilitar o relatório mensal de vendas. Qual comando SQL você usaria para criar uma visualização capaz de simplificar essa consulta?

A

CREATE TABLE

B

CREATE INDEX

C

CREATE VIEW

D

ALTER TABLE

E

SELECT INTO



A alternativa C está correta.

Ao criar uma view usando o comando CREATE VIEW, você pode definir uma consulta SQL que calcula o total de vendas de cada produto por mês. Essa visualização pode então ser consultada facilmente sempre quando precisar gerar o relatório mensal de vendas, simplificando o processo e evitando a necessidade de escrever a consulta complexa repetidamente. As outras opções mencionadas não são adequadas para criar uma visualização no PostgreSQL.

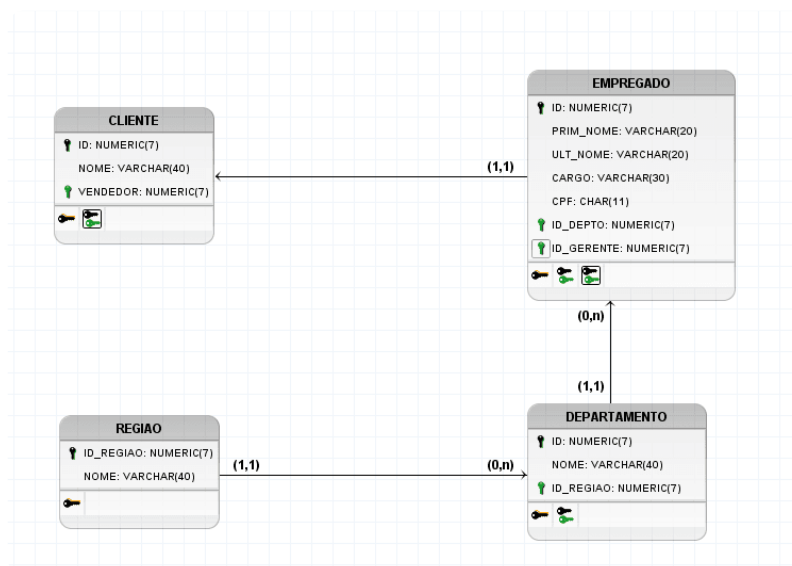
## Consultas básicas com SELECT na prática

Vamos agora criar o banco de dados da empresa que utilizaremos como exemplo nas nossas demonstrações e, em seguida, iremos executar alguns comandos básicos de SELECT.

Você pode baixar o **script da criação** das tabelas e inserção das linhas do banco de dados da empresa.

Link para download do arquivo [empresa.sql](#)

O modelo de dados da empresa é apresentado a seguir.



Modelo de dados de uma empresa.

Neste vídeo, vamos demonstrar por meio de um passo a passo como fazer consultas básicas em SQL com a cláusula SELECT.



Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

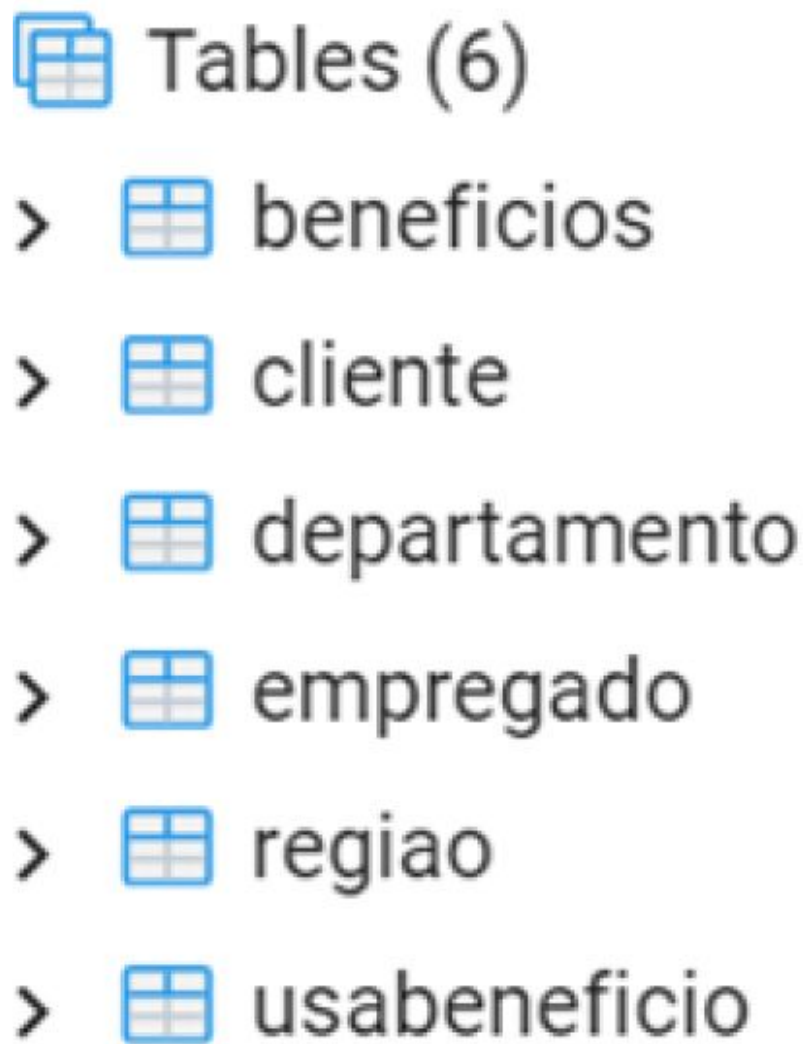
## Roteiro de prática



Para criar o banco de dados da empresa, siga estes passos:

1. Abra o PGAdmin4 e faça conexão no servidor.
2. Crie um banco de dados chamado EMPRESA.
3. Abra uma janela de consulta.
4. Carregue o script da empresa na janela de consulta.
5. Execute o script.
6. Valide a criação das tabelas na aba tabelas do banco da empresa.

Veja um exemplo!



Tables (6).

---

Execute agora os seguintes comandos:

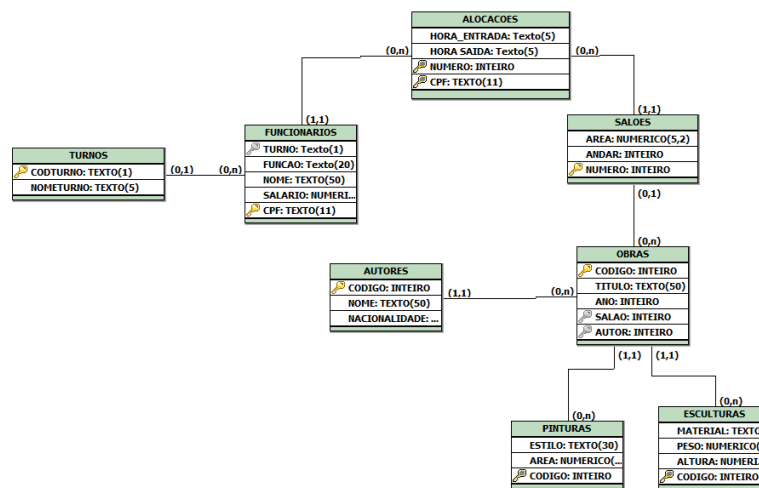
- Retorne todo o conteúdo da tabela EMPREGADO.
- Retorne apenas as colunas id, prim\_nome e ult\_nome da tabela EMPREGADO.
- Retorne o salário anual do empregado.
- Retorne o nome completo do empregado.

- Retorne o nome completo do empregado com o alias NOME COMPLETO.
- Retorne o resultado da soma de 9 com 5 a data e hora do servidor.
- Retorne a média salarial, a soma dos salários, o maior e o menor salário dos empregados.
- Retorne os maiores e menores ID, ult\_nome e data de admissão dos empregados.

Você pode acessar os comandos que resolvem as consultas no arquivo [sqlmod1.5.txt](#)

## Atividade 5

Vamos agora executar uma atividade prática utilizando o banco de dados do museu. Seu modelo lógico é o seguinte:



Modelo lógico de banco de dados do minimundo EMPRESA.

Você pode baixar o script da criação das tabelas e inserção das linhas do banco de dados do museu.

Conteúdo do link script da criação









Link para download do arquivo [museu.sql](#)

Crie o banco do museu e siga estes passos:

1. Abra o PGAdmin4 e faça conexão no servidor.
2. Crie um banco de dados chamado MUSEU.
3. Abra uma janela de consulta.
4. Carregue o script do museu na janela de consulta.
5. Execute o script.
6. Valide a criação das tabelas na aba tabelas do banco do museu.

Veja um exemplo!

## Tables (8)

- >  alocacoes
- >  autores
- >  esculturas
- >  funcionarios
- >  obras
- >  pinturas
- >  saloes
- >  turnos

Tables (8).

---

Para realizar o exercício, faça conexão e realize as seguintes consultas:

- Listar o conteúdo da tabela de autores.
- Listar o nome e a nacionalidade de todos os autores.
- Listar CPF, nome, salário e salário anual dos funcionários.
- Listar o nome do autor concatenado com a sua nacionalidade com o alias de coluna nome e nacionalidade.
- Listar o nome do autor concatenado com a sua nacionalidade.
- Listar a área média com o alias área média, e número de salões com o alias 'Total de salões'.
- Listar o total de pinturas com o alias 'Quantidade de pinturas'.
- Listar o maior peso de uma escultura.
- Listar o título de uma obra que aparece primeiro na ordem alfabética crescente.

Você pode acessar os comandos que resolvem as consultas no arquivo [exercmod1.5](#).

### Consultas usando a cláusula WHERE

Aprofunde seu conhecimento a partir das chamadas técnicas de filtragem. Veremos como usar o WHERE para selecionar registros com base em condições específicas.

Neste vídeo, vamos aprender a implementar consultas SQL utilizando a cláusula WHERE. Você entenderá como filtrar dados de maneira precisa e eficiente, utilizando condições específicas para obter resultados relevantes em bancos de dados.



#### Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Em nossas consultas, usaremos como base a tabela ALUNO, conforme imagem a seguir.

ALUNO		
CODIGOALUNO	int	PK
NOME	varchar(90)	
SEXO	char(1)	
DTNASCIMENTO	date	
EMAIL	varchar(50)	N

Tabela ALUNO.

Recomendamos que você crie a tabela e insira algumas linhas, o que pode ser feito usando o script a seguir, a partir da ferramenta de sua preferência. Para isso, tenha em mente que é necessário estar conectado ao PostgreSQL e acessando algum database criado por você.

sql

```
CREATE TABLE ALUNO (  
    CODIGOALUNO int NOT NULL,  
    NOME varchar(90) NOT NULL,  
    SEXO char(1) NOT NULL,  
    DTNASCIMENTO date NOT NULL,  
    EMAIL varchar(30) NULL,  
    CONSTRAINT ALUNO_pk PRIMARY KEY (CODIGOALUNO));  
INSERT INTO ALUNO (CODIGOALUNO, NOME, SEXO, DTNASCIMENTO, EMAIL) VALUES (1, 'JOSÉ  
FRANCISCO TERRA', 'M', '28/10/1989', 'JFT@GMAIL.COM');  
INSERT INTO ALUNO (CODIGOALUNO, NOME, SEXO, DTNASCIMENTO,EMAIL) VALUES (2, 'ANDREY COSTA  
FILHO', 'M', '20/10/1999', 'ANDREYCF@HOTMAIL.COM');  
INSERT INTO ALUNO (CODIGOALUNO, NOME, SEXO, DTNASCIMENTO,EMAIL) VALUES (3, 'PATRICIA  
TORRES LOUREIRO', 'F', '20/10/1980', 'PTORRES@GMAIL.COM');  
INSERT INTO ALUNO (CODIGOALUNO, NOME, SEXO, DTNASCIMENTO,EMAIL) VALUES (4, 'CARLA MARIA  
MACIEL', 'F', '20/11/1996', NULL);  
INSERT INTO ALUNO (CODIGOALUNO, NOME, SEXO, DTNASCIMENTO,EMAIL) VALUES (5, 'LEILA SANTANA  
COSTA', 'F', '20/11/2001', NULL);
```

## Recuperando dados com select/from/where/order by

Uma sintaxe básica para o comando SELECT, com o uso das cláusulas WHERE e ORDER BY, está expressa a seguir. Veja!

sql

```
SELECT COLUNA1 [[AS] APELIDOCOLUNA1],  
COLUNA2 [[AS] APELIDOCOLUNA2],  
...  
COLUNAN [[AS] APELIDOCOLUNAN]  
FROM TABELA  
WHERE  
ORDER BY EXPRESSÃO1[ASC|DESC] [NULLS \{FIRST|LAST\}], [EXPRESSÃO2[ASC|DESC] [NULLS \  
{FIRST|LAST\}...];
```

O propósito do SELECT é declararmos as colunas da consulta. No FROM, informamos a tabela que foi alvo da consulta. No WHERE, especificamos alguma condição, simples ou composta, para filtrar registros que serão recuperados pelo SGBD. No ORDER BY, declaramos uma ou mais colunas como critério de ordenação, com possibilidade de especificarmos se valores NULL aparecem no início ou no final do resultado.

A cláusula WHERE realiza a operação de restrição da álgebra relacional, também conhecida como seleção – não confundir com o comando SELECT.

Ainda, a construção de uma condição na cláusula WHERE envolve operadores relacionais, conforme tabela a seguir.

Operador	Significado
<	menor
<=	menor ou igual a

Operador	Significado
>	maior
>=	maior ou igual a
=	igual
<> ou !=	diferente

Tabela: Operadores relacionais.  
Sidney Venturi.

Além dos operadores relacionais, a construção de uma condição na cláusula WHERE pode fazer uso dos seguintes operadores lógicos:

Operador	Significado
AND	conjunção
OR	disjunção
NOT	negação

Tabela: Operadores lógicos.  
Sidney Venturi.

Vamos estudar alguns exemplos de consultas com o uso da cláusula WHERE!

## Consulta 01 + resultado

Para mostrar o nome e a data de nascimento das professoras, observe o exemplo da consulta no código:

```
sql
SELECT NOME, DTNASCIMENTO
FROM ALUNO
WHERE SEXO='F';
```

Agora, veja o resultado!

	asc nome	dtascimento
1	PATRICIA TORRES LOUREIRO	1980-10-20
2	CARLA MARIA MACIEL	1996-11-20
3	LEILA SANTANA COSTA	2001-11-20

Resultado consulta 01.

Perceba que foi criada uma condição simples de igualdade envolvendo a coluna SEXO da tabela ALUNO. O SGBD percorre cada registro da tabela ALUNO, avalia a condição (linha 3) e exibe as colunas NOME e DTNASCIMENTO para cada registro cuja avaliação da condição retorne verdadeiro.

## Consulta 02 + resultado

Para mostrar o nome e a data de nascimento das professoras que fazem aniversário em novembro, observe o exemplo:

sql

```
SELECT NOME, DTNASCIMENTO  
FROM ALUNO  
WHERE SEXO='F' AND EXTRACT (MONTH FROM DTNASCIMENTO)=11;
```

O resultado da consulta 02 está expresso na imagem a seguir.

	nome	dtnascimento
1	CARLA MARIA MACIEL	1996-11-20
2	LEILA SANTANA COSTA	2001-11-20

Resultado consulta 02.

Perceba que foi criada uma condição composta envolvendo uma conjunção. O SGBD retornará os registros que possuem o valor "F" para a coluna SEXO e o inteiro 11 como valor do mês referente à data de nascimento.

## Atividade 1

Observe a seguinte estrutura de tabela chamada produtos.

id	nome	preço	quantidade
1	Camiseta	25.99	100
2	Calça	39.99	75
3	Sapato	49.99	50

Qual comando SQL para retornar apenas o nome dos produtos com quantidade disponível de pelo menos 50 unidades?

A

SELECT Nome FROM produtos WHERE quantidade >= 50

B

SELECT \* FROM produtos WHERE quantidade >= 50

C

SELECT Nome, Quantidade FROM produtos WHERE quantidade > 50

D

SELECT \* FROM produtos WHERE quantidade > 50

E

```
SELECT * FROM produtos WHERE quantidade <> 50
```



A alternativa A está correta.

Como queremos apenas o nome dos produtos com quantidade maior ou igual a 50, no SELECT listamos apenas a coluna nome e na cláusula WHERE filtramos a quantidade que seja maior que 50.

## Recuperando dados com o uso dos operadores IN e BETWEEN

Vejamos agora a utilização do IN e do BETWEEN, explorando técnicas avançadas de filtragem de dados em bancos de dados. Aprenderemos a utilizar o operador IN para filtrar resultados com base em uma lista de valores e o operador BETWEEN para selecionar registros dentro de um intervalo específico.

Neste vídeo, vamos apresentar como fazer a recuperação de dados com o uso dos operadores IN e BETWEEN da linguagem SQL. Acompanhe!



Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

## Recuperando dados com o uso do operador in

O operador [NOT] IN pode ser utilizado em consultas que envolvam comparações usando uma lista de valores.

### Consulta 03 + resultado

Para listar o nome dos alunos que fazem aniversário no segundo semestre, observe o código:

```
sql
```

```
SELECT NOME, DTNASCIMENTO  
FROM ALUNO  
WHERE EXTRACT (MONTH FROM DTNASCIMENTO) IN (7,8,9,10,11,12);
```

O resultado da consulta 03 está expresso na imagem a seguir.



	ABC nome 
1	JOSÉ FRANCISCO TERRA
2	ANDREY COSTA FILHO
3	CARLA MARIA MACIEL
4	LEILA SANTANA COSTA

Resultado consulta 03.

Note que a expressão na cláusula WHERE compara o mês de nascimento de cada aluno com os valores da lista contendo os inteiros correspondentes aos meses do segundo semestre.

## Recuperando dados com o uso do operador BETWEEN

O operador [NOT] BETWEEN verifica se determinado valor encontra-se no intervalo entre dois valores.



### Exemplo

X BETWEEN Y AND Z é equivalente a  $X \geq Y$  AND X. De modo semelhante, X NOT BETWEEN Y AND Z é equivalente a XZ.

## Consulta 04 + resultado

Para listar o nome dos alunos nascidos entre 1985 e 2005, observe o exemplo:

sql

```
1 SELECT NOME
2 FROM ALUNO
3 WHERE EXTRACT (YEAR FROM DTNASCIMENTO) BETWEEN 1985 AND 2005;
```

O resultado da consulta 04 está expresso na imagem a seguir.

	ABC nome 
1	ANDREY COSTA FILHO
2	LEILA SANTANA COSTA

Resultado consulta 04.

Note que a expressão na cláusula WHERE compara o ano de nascimento de cada aluno junto com o intervalo especificado pelo operador BETWEEN. Caso quiséssemos extrair o mesmo resultado sem o uso do BETWEEN, poderíamos programar um comando equivalente, conforme a seguir.

```
sql

SELECT NOME
FROM ALUNO
WHERE EXTRACT (YEAR FROM DTNASCIMENTO) >= 1985 AND EXTRACT (YEAR FROM DTNASCIMENTO) <=
2005;
```

## Atividade 2

Um programador recuperou os dados dos bairros Penha, Ipanema, Flamengo e Centro gravados na coluna BAIRRO da tabela CLIENTE, a seguir especificada. CLIENTE (IDCLIENTE, NOME, ENDERECO, BAIRRO, CIDADE, UF, CEP) A sintaxe SQL correta usada por ele para realizar essa atividade foi SELECT \* FROM CLIENTE

A

WHERE BAIRRO IN ('Penha ', 'Ipanema ', 'Flamengo ', 'Centro').

B

WHERE BAIRRO = ('Penha ', 'Ipanema ', 'Flamengo ', 'Centro').

C

WHEN BAIRRO = ('Penha ', 'Ipanema ', 'Flamengo ', 'Centro').

D

WHERE BAIRRO BETWEEN ('Penha ', 'Ipanema ', 'Flamengo ', 'Centro').

E

WHERE BAIRRO NOT IN ('Penha', 'Ipanema','Flamengo', 'Centro').



A alternativa A está correta.

Para recuperar os registros de interesse, foi utilizado o operador IN com o uso de uma lista contendo os bairros em questão. O SGBD compara o bairro do cliente juntamente com os elementos especificados na lista de bairros em questão.

## Recuperando dados com o uso do operador LIKE

Exploraremos agora uma poderosa ferramenta para filtrar padrões de texto em bancos de dados. Aprenderemos a utilizar o operador LIKE para encontrar registros que correspondam a padrões específicos de caracteres, permitindo buscas flexíveis e eficazes.

Neste vídeo, vamos entender como funciona a recuperação de dados com o uso do operador LIKE da linguagem SQL.



### Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

O uso do [NOT] LIKE permite realizar buscas em uma cadeia de caracteres.

Trata-se de um recurso bastante utilizado em buscas textuais. Você pode utilizar os símbolos especiais a seguir:

- \_ para ignorar qualquer caractere específico.
- % para ignorar qualquer padrão.

Vamos estudar alguns exemplos?

## Consulta 05 + resultado

Para listar o nome dos alunos que possuem a string COSTA em qualquer parte do nome, veja o código:

```
sql  
  
SELECT NOME  
FROM ALUNO  
WHERE NOME LIKE '%COSTA%';
```

O resultado da consulta 05 está expresso na imagem a seguir.

	nome
1	ANDREY COSTA FILHO
2	LEILA SANTANA COSTA

Resultado consulta 05.

O uso do padrão '%COSTA%' significa que não importa o conteúdo localizado antes e depois da string "COSTA".

## Consulta 06 + resultado

Para listar o nome dos alunos que possuem a letra "A" na segunda posição do nome, veja o exemplo de consulta:

```
sql
```

```
SELECT NOME  
FROM ALUNO  
WHERE NOME LIKE '_A%';
```

O resultado da consulta 06 está expresso na imagem a seguir.

	nome
1	PATRÍCIA TORRES LOUREIRO
2	CARLA MARIA MACIEL

Resultado consulta 06.

Note que, para especificar o “A” na segunda posição, o SGBD desprezará qualquer valor na primeira posição da string, não importando o que estiver localizado à direita do “A”.

## Consulta 07 + resultado

Para listar o nome e a data de nascimento dos alunos que não possuem a string “MARIA” fazendo parte do nome, veja o exemplo de consulta a seguir:

```
sql
```

```
SELECT NOME, DTNASCIMENTO  
FROM ALUNO  
WHERE NOME NOT LIKE '%MARIA%';
```

O resultado da consulta 07 está expresso na imagem a seguir.

	nome	dtnascimento
1	JOSÉ FRANCISCO TERRA	1989-10-28
2	ANDREY COSTA FILHO	1999-10-20
3	PATRÍCIA TORRES LOUREIRO	1980-10-20
4	LEILA SANTANA COSTA	2001-11-20

Resultado consulta 07.

Estamos diante de um caso semelhante ao da consulta 05.

No entanto, utilizamos o operador de negação para retornar os registros de interesse.

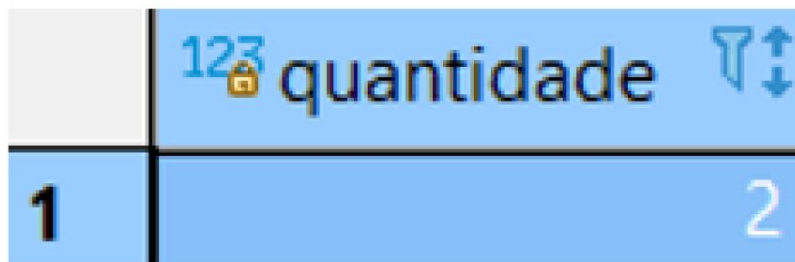
## Consulta 08 + resultado

Quanto alunos possuem conta de e-mail no Gmail? Veja o exemplo de consulta na imagem a seguir.

sql

```
SELECT COUNT(*) AS QUANTIDADE  
FROM ALUNO  
WHERE EMAIL LIKE '%@GMAIL.%';
```

O resultado da consulta 08 está expresso na imagem a seguir.



	123 quantidade
1	2

Resultado consulta 08.

Note que, mais uma vez, estamos diante de um caso semelhante ao da consulta 05. Buscamos pela string “@GMAIL.” em qualquer posição da coluna EMAIL.

### Atividade 3

Gabriel é analista de sistemas de uma empresa de tecnologia de informação e recebeu a tarefa de recuperar todos os registros da tabela CLIENTE em que o valor da coluna “NOMECLIENTE” contenha a cadeia “da Silva” em qualquer parte do nome. Assinale a alternativa que permita a Gabriel executar sua tarefa.

A

SELECT \* FROM CLIENTE WHERE NOMECLIENTE = 'da Silva '.

B

SELECT \* FROM CLIENTE WHERE NOMECLIENTE != 'da Silva '.

C

SELECT \* FROM CLIENTE WHERE NOME LIKE '%da Silva% '.

D

SELECT \* FROM CLIENTE WHERE NOMECLIENTE LIKE '%da Silva% '.

E

SELECT \* FROM CLIENTE WHERE NOMECLIENTE LIKE ' da Silva%'.



A alternativa D está correta.

Para recuperar os registros que contenham "da Silva" em qualquer parte do nome, utiliza-se o comando LIKE com auxílio do "%" como forma do SGBD desconsiderar qualquer padrão à esquerda e à direita da string de interesse.

## Recuperando dados com o uso do operador NULL e usando ordenação

Vamos explorar agora técnicas importantes para lidar com valores nulos em bancos de dados e para classificar resultados de consulta. Aprenderemos a usar o operador IS NULL para identificar registros sem valores e a ordenar dados usando a cláusula ORDER BY.

Neste vídeo, vamos compreender como ocorre a recuperação de dados com o uso do operador NULL, bem como a aplicação da ordenação na linguagem SQL.



Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

## Recuperando dados com o uso do operador NULL

Quando uma coluna é opcional, significa que existe possibilidade de que algum registro não possua valor cadastrado para a coluna em questão. Nessa hipótese, há entendimento de que o valor da coluna é "desconhecido" ou "não aplicável".

Para testar se uma coluna possui valor cadastrado, usa-se a expressão COLUNA IS NOT NULL.

Vamos estudar alguns exemplos!

### Consulta 09 + resultado

Para listar o nome, a data de nascimento e o e-mail dos alunos que têm endereço eletrônico cadastrado, veja o exemplo de consulta a seguir.

sql

```
1 SELECT NOME, DTNASCIMENTO, EMAIL
2 FROM ALUNO
3 WHERE EMAIL IS NOT NULL;
```

Observe na imagem a seguir os resultados da consulta.

	nome	dtascimento	email
1	JOSE FRANCISCO TERRA	1989-10-28	JFT@GMAIL.COM
2	ANDREY COSTA FILHO	1999-10-20	ANDREYCF@HOTMAIL.COM
3	PATRICIA TORRES LOUREIRO	1980-10-20	PTORRES@GMAIL.COM

Resultado consulta 09.

O SGBD retorna os registros onde há algum conteúdo cadastrado na coluna EMAIL.

## Consulta 10 + resultado

Para retornar o nome dos alunos sem e-mail cadastrado no banco de dados, veja o exemplo:

```
sql
SELECT NOME
FROM ALUNO
WHERE EMAIL IS NULL;
```

Observe na imagem a seguir os resultados da consulta.

	nome
1	CARLA MARIA MACIEL
2	LEILA SANTANA COSTA

Resultado consulta 10.

O SGBD retorna os registros sobre os quais não há valor cadastrado na coluna EMAIL.

## Recuperando dados usando ordenação dos resultados

Para melhor organizar os resultados de uma consulta, podemos especificar critérios de ordenação. Vejamos alguns exemplos

### Consulta 11 + resultado

Para retornar o nome e a data de nascimento dos alunos, ordenando os resultados por nome, de maneira ascendente, veja o exemplo:

```
sql
1 SELECT NOME, DTNASCIMENTO
2 FROM ALUNO
3 ORDER BY NOME;
```

Observe na imagem a seguir os resultados da consulta.

	ABC nome	🕒 dtnascimento
1	ANDREY COSTA FILHO	1999-10-20
2	CARLA MARIA MACIEL	1996-11-20
3	JOSÉ FRANCISCO TERRA	1989-10-28
4	LEILA SANTANA COSTA	2001-11-20
5	PATRÍCIA TORRES LOUREIRO	1980-10-20

Resultado consulta 11.

O SGBD retorna os registros da tabela ALUNO, obedecendo ao critério de ordenação especificado na linha 3 da consulta. O padrão ascendente (ASC) é opcional.

## Consulta 12 + resultado

Para retornar o nome e a data de nascimento dos alunos, ordenando os resultados de modo ascendente pelo mês de nascimento e, em seguida, pelo nome, também de modo ascendente, veja o exemplo:

sql

```
1 SELECT NOME, DTNASCIMENTO
2 FROM ALUNO
3 ORDER BY EXTRACT(MONTH FROM DTNASCIMENTO), NOME;
```

Observe na imagem a seguir os resultados da consulta.

	ABC nome	🕒 dtnascimento
1	ANDREY COSTA FILHO	1999-10-20
2	JOSÉ FRANCISCO TERRA	1989-10-28
3	PATRÍCIA TORRES LOUREIRO	1980-10-20
4	CARLA MARIA MACIEL	1996-11-20
5	LEILA SANTANA COSTA	2001-11-20

Resultado consulta 12.

O SGBD retorna os registros da tabela ALUNO, levando em conta o critério de ordenação especificado na linha 3 da consulta. Foi realizada ordenação pelo mês de nascimento; em seguida, pelo nome.

## Atividade 4

Considere a tabela alunos com a estrutura a seguir.

id	nome	idade	nota_final
1	João	18	8.5



id	nome	idade	nota_final
2	Maria	20	9.2
3	Pedro	19	7.8
4	Ana	21	8.9

Qual comando SQL você usaria para selecionar todos os dados dos alunos, ordenando-os por nome em ordem alfabética?

A

SELECT \* FROM alunos ORDER BY nome ASC

B

SELECT \* FROM alunos ORDER BY nome DESC

C

SELECT \* FROM alunos SORT BY nome ASC

D

SELECT \* FROM alunos SORT BY nome DESC

E

SELECT Nome FROM alunos ORDER nome ASC

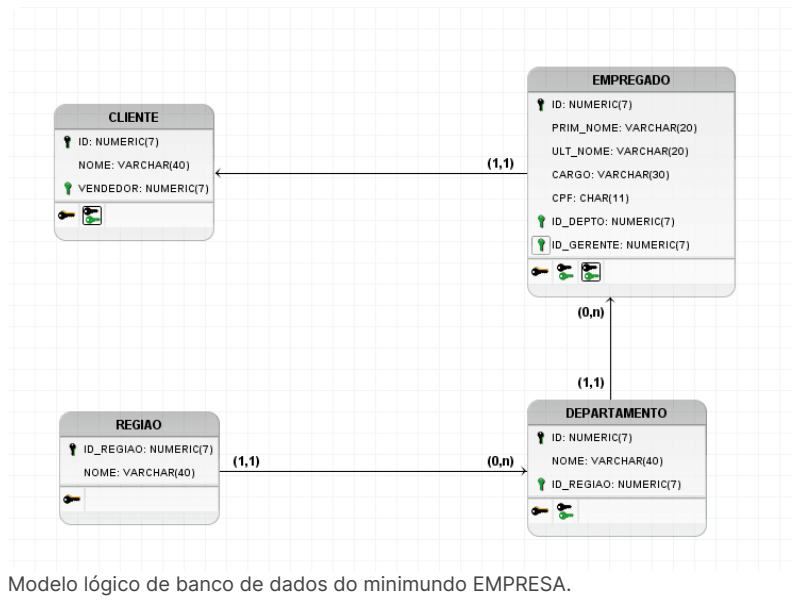


A alternativa A está correta.

O comando ORDER BY no SQL é usado para classificar os resultados de uma consulta de acordo com uma ou mais colunas especificadas. Nesse caso, queremos ordenar os alunos por nome em ordem alfabética, então usamos a cláusula ORDER BY nome ASC. As outras opções estão incorretas devido a erros de sintaxe ou uso incorreto do comando ORDER BY.

## Consultas com o comando SELECT e a cláusula WHERE utilizando PGADMIN

Vamos agora ver exemplos de uso dos comandos vistos até aqui. Para isso, iremos utilizar o banco de dados da empresa cujo modelo lógico pode ser vista na imagem a seguir.



Neste vídeo, vamos colocar em prática consultas com o comando SELECT e cláusula WHERE utilizando PGADMIN. Não perca!



Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

## Roteiro de prática

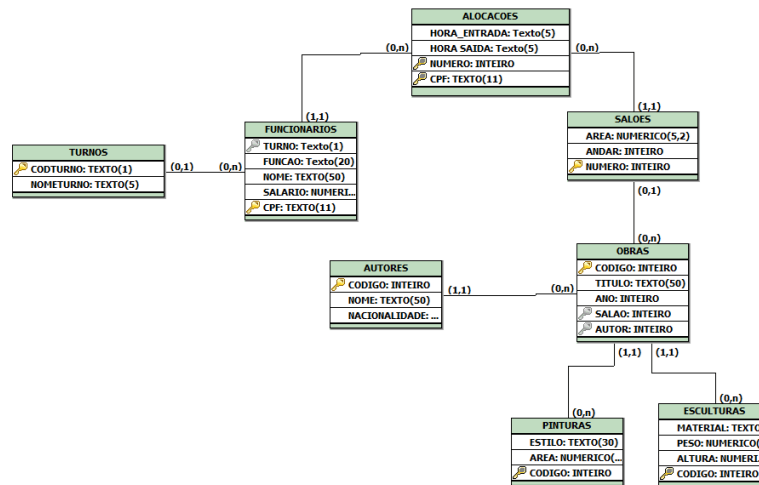
Execute comandos seguindo as estas diretrizes:

- Retornando empregado com ID > 3.
- Retornando ID do cliente Ponto Quente.
- Filtrando por data.
- Filtrando com várias condições.
- Uso do IN.
- Uso do BETWEEN.
- Uso de LIKE.
- Uso de IS NULL.
- Ordenando a consulta.

Você pode acessar os comandos que resolvem as consultas no arquivo [sqlmod2.5.txt](#)

## Atividade 5

Vamos executar uma atividade prática utilizando o banco de dados do museu cujo modelo lógico é:



Modelo lógico de banco de dados do minimundo MUSEU.

Escreva os seguintes comandos de consulta.

1. Listar todos os dados dos autores com código maior que 105.
2. Listar todos os dados dos autores com código maior que 105 e de nacionalidade francesa.
3. Listar todos os dados dos autores franceses ou brasileiros.
4. Listar todos os dados dos autores com código entre 104 e 107 inclusive.
5. Listar todos os dados dos autores com Dali no nome.
6. Listar todos os dados dos salões cuja área é nula.
7. Listar todos os dados dos autores em ordem descendente de nacionalidade.
8. Listar todos os dados dos autores em ordem ascendente de nacionalidade e descendente de código.
9. Repetir o comando anterior ordenando pela posição da coluna.

Você pode acessar os comandos que resolvem as consultas no arquivo [exercmod2.5](#).

## Agrupamento de dados

Vamos explorar como organizar e resumir grandes conjuntos de dados em bancos de dados. Aprenderemos a utilizar a cláusula GROUP BY para agrupar registros com base em valores comuns e a aplicar funções de agregação para calcular estatísticas sobre esses grupos.

Neste vídeo, vamos entender como a cláusula GROUP BY agrupa registros com base em valores comuns, aplicando as funções de agregação para calcular estatísticas sobre esses grupos.



#### Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

## Consultas com GROUP BY e HAVING

Em nossas consultas, usaremos como base a tabela FUNCIONARIO, conforme imagem a seguir.

FUNCIONARIO		
CODIGOFUNCIONARIO	int	PK
NOME	varchar(90)	
CPF	char(15)	
SEXO	char(1)	
DTNASCIMENTO	date	
SALARIO	real	N

Tabela FUNCIONÁRIO.

Recomendamos que você crie a tabela e insira algumas linhas, o que pode ser feito usando o script a seguir, a partir da ferramenta de sua preferência. Para isso, tenha em mente que é necessário estar conectado ao PostgreSQL e acessando algum database criado por você.

sql

```
CREATE TABLE FUNCIONARIO (  
    CODIGOFUNCIONARIO int NOT NULL,  
    NOME varchar(90) NOT NULL,  
    CPF char(15) NULL,  
    SEXO char(1) NOT NULL,  
    DTNASCIMENTO date NOT NULL,  
    SALARIO real NULL,  
    CONSTRAINT FUNCIONARIO_pk PRIMARY KEY (CODIGOFUNCIONARIO));  
INSERT INTO FUNCIONARIO (CODIGOFUNCIONARIO, NOME, CPF, SEXO, DTNASCIMENTO, SALARIO)  
VALUES (1, 'ROBERTA SILVA BRASIL', '82998', 'F', '20/02/1980',7000);  
INSERT INTO FUNCIONARIO (CODIGOFUNCIONARIO, NOME, CPF, SEXO, DTNASCIMENTO, SALARIO)  
VALUES (2, 'MARIA SILVA BRASIL', '9876', 'F', '20/09/1988',9500);  
INSERT INTO FUNCIONARIO (CODIGOFUNCIONARIO, NOME, CPF, SEXO, DTNASCIMENTO, SALARIO)  
VALUES (3, 'GABRIELLA PEREIRA LIMA', '32998', 'F', '20/02/1990',6000);  
INSERT INTO FUNCIONARIO (CODIGOFUNCIONARIO, NOME, CPF, SEXO, DTNASCIMENTO, SALARIO)  
VALUES (4, 'MARCOS PEREIRA BRASIL', '9999', 'M', '20/02/1999',6000);  
INSERT INTO FUNCIONARIO (CODIGOFUNCIONARIO, NOME, CPF, SEXO, DTNASCIMENTO, SALARIO)  
VALUES (5, 'HEMERSON SILVA BRASIL', '9111', 'M', '20/12/1992',4000);
```

Após a criação da tabela e a inserção dos registros, podemos utilizar o código a seguir para exibir todo o seu conteúdo. Veja com mais detalhes!

sql

```
1 SELECT *  
2 FROM FUNCIONARIO;
```

O resultado da consulta será semelhante à tabela a seguir.

	121 codigofuncionario	nome	cpf	sexo	dt Nascimento	122 salario
1	1	ROBERTA SILVA BRASIL	[NULL]	F	1980-02-20	7000.0
2	2	MARIA SILVA BRASIL	[NULL]	F	1988-09-20	9500.0
3	3	GABRIELLA PEREIRA LIMA	[NULL]	F	1990-02-20	6000.0
4	4	MARCOS PEREIRA BRASIL	[NULL]	M	1999-02-20	6000.0
5	5	HEMERSON SILVA BRASIL	[NULL]	M	1992-12-20	4000.0

Registros da tabela FUNCIONARIO.

## Grupo de dados

Vamos aprender a projetar consultas com o uso de agrupamento de dados, com auxílio dos comandos GROUP BY e HAVING.

Perceba que a maior parte dessas consultas está atrelada ao uso de alguma função de resumo, por exemplo, SUM, AVG, MIN e MAX, as quais representam, respectivamente, soma, média, mínimo e máximo. Logo, essas consultas são úteis para quem tem interesse em construir relatórios e aplicações de natureza mais gerencial e analítica. Os valores de determinada coluna podem formar grupos sobre os quais podemos ter interesse em recuperar dados.

Por exemplo, se avaliarmos o resultado da consulta anterior, podemos naturalmente dividir os registros de acordo com o valor da coluna sexo. Teríamos, então, a seguinte estrutura:

M {4,5} {MARCOS PEREIRA BRASIL, HEMERSON SILVA BRASIL} {...}
F {1,2,3} {ROBERTA SILVA BRASIL, MARIA SILVA BRASIL, GABRIELLA PEREIRA LIMA} {...}

Tabela: Representação da coluna sexo.  
Sidney Venturi.

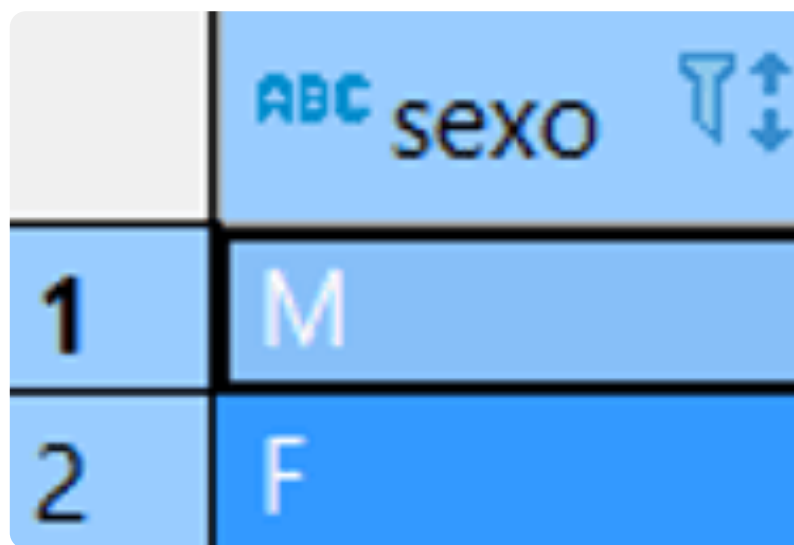
Todas as linhas com o mesmo valor para a coluna sexo formam um grupo. Representamos as linhas com mais de um valor com o uso de chaves para fins ilustrativos, dado que estamos diante de linhas dentro de colunas, uma representação que não existe no modelo relacional.

A estrutura anterior possui somente um grupo, o qual é formado pela coluna SEXO da tabela FUNCIONARIO. Como exibir esse grupo em SQL?

Uma solução é adicionar a cláusula DISTINCT ao comando SELECT.

```
sql
1 SELECT DISTINCT SEXO
2 FROM FUNCIONARIO;
```

O resultado da consulta anterior pode ser visualizado na imagem a seguir.



	ABC	sexo
1		M
2		F

Grupo de dados baseado na coluna SEXO da tabela FUNCIONARIO.

Vamos perceber que em SQL a cláusula mais adequada para trabalhar com agrupamento de dados é o GROUP BY.

## Grupo de dados com GROUP BY

A cláusula GROUP BY serve para exibir resultados de consulta de acordo com um grupo especificado. Ela é declarada após a cláusula FROM, ou após a cláusula WHERE, caso exista na consulta. Por exemplo, para obter o mesmo resultado do comando anterior, podemos usar o seguinte código:

```
sql
```

```
1 SELECT SEXO
2 FROM FUNCIONARIO
3 GROUP BY SEXO;
```

No entanto, vamos perceber que o uso mais conhecido da cláusula GROUP BY ocorre quando associada a funções de agregação, tais como COUNT, MIN, MAX e AVG. Vamos estudar alguns exemplos!

## Consulta 01 + resultado

Para retornar o número de funcionários por sexo, temos o seguinte código:

```
sql
```

```
1 SELECT SEXO, COUNT(*) AS QUANTIDADE
2 FROM FUNCIONARIO
3 GROUP BY SEXO;
```

O resultado da consulta 01 está expresso na imagem a seguir.

	ABC sexo ↕	123 quantidade ↕
1	M	2
2	F	3

Resultado consulta 01.

O SGBD realiza o agrupamento de dados de acordo com os valores da coluna SEXO. Em seguida, para cada grupo encontrado, a função COUNT(\*) é executada e o resultado exibido.

E se tivéssemos interesse em exibir os resultados da consulta anterior em uma única linha? Poderíamos usar o código a seguir.

```
sql
```

```
SELECT
  (SELECT COUNT(*) AS "M" FROM FUNCIONARIO WHERE SEXO='M' ),
  (SELECT COUNT (*) AS "F" FROM FUNCIONARIO WHERE SEXO='F' );
```

Veja o seu resultado!

## Consulta 02 + resultado

Para retornar a média salarial por sexo, temos o seguinte código:

```
sql

SELECT SEXO,
       AVG(SALARIO) AS MEDIASALARIAL
FROM FUNCIONARIO
GROUP BY SEXO;
```

O resultado da consulta 02 está expresso na imagem a seguir.

	ABC sexo T↑↓	123 mediasalarial T↑↓
1	M	5.000
2	F	7.500

Resultado consulta 02.

O SGBD realiza o agrupamento de dados de acordo com os valores da coluna SEXO. Em seguida, para cada grupo encontrado, a função AVG (SALARIO) é executada; e o resultado, exibido.

## Consulta 03 + resultado

Para retornar, por mês de aniversário, a quantidade de colaboradores, o menor salário, o maior salário e o salário médio, ordene os resultados por mês de aniversário, conforme o código.

```
sql

SELECT EXTRACT(MONTH FROM DTNASCIMENTO) AS MES,
       COUNT(*) AS QUANTIDADE,
       MIN(SALARIO) AS MENORSALARIO,
       ROUND(AVG(SALARIO)::NUMERIC,0) AS SALARIOMEDIO,
       MAX(SALARIO) AS MAIORSALARIO
FROM FUNCIONARIO
GROUP BY EXTRACT(MONTH FROM DTNASCIMENTO)
ORDER BY EXTRACT(MONTH FROM DTNASCIMENTO);
```

O resultado da consulta 03 está expresso na imagem a seguir.



	mes	quantidade	menorsalario	salariomedio	maiorsalario
1	2	3	6.000	6.333	7.000
2	9	1	9.500	9.500	9.500
3	12	1	4.000	4.000	4.000

Resultado consulta 03.

O SGBD realiza o agrupamento de dados de acordo com o mês de nascimento dos funcionários. Depois, para cada grupo encontrado, as funções de agregação são executadas e, em seguida, exibidos os resultados. Perceba também que, na linha 4, utilizamos a função ROUND com objetivo de mostrar ao usuário final somente a parte inteira dos valores resultantes da média salarial.

## Consulta 04 + resultado

Para retornar, por mês de aniversário, o mês, o sexo e a quantidade de colaboradores, veja o código a seguir. Os resultados ordenados são apresentados pelo mês.

```
sql

SELECT EXTRACT(MONTH FROM DTNASCIMENTO) AS MES,
       SEXO,
       COUNT(*) AS QUANTIDADE
FROM FUNCIONARIO
GROUP BY EXTRACT(MONTH FROM DTNASCIMENTO), SEXO
ORDER BY EXTRACT(MONTH FROM DTNASCIMENTO);
```

O resultado da consulta 04 está expresso na imagem a seguir.

	mes	sexo	quantidade
1	2	F	2
2	2	M	1
3	9	F	1
4	12	M	1

Resultado consulta 04.

O SGBD realiza o agrupamento de dados de acordo com os valores do mês de aniversário. Em seguida, no contexto de cada mês encontrado, mais um grupo é construído por sexo. Finalmente, para cada ocorrência mês/sexo, o número de colaboradores é calculado.

## Atividade 1

Suponha que exista em um banco de dados uma tabela denominada CLIENTE, assim estruturada: CLIENTE (CODIGOCLIENTE, NOME, SEXO, BAIRRO, RENDA). Você foi solicitado a escrever um comando SQL para obter a renda média dos clientes por bairro.

O comando correto é:

A

```
SELECT BAIRRO, MIN(RENDA)FROM CLIENTE GROUP BY BAIRRO
```

B

```
SELECT SEXO, SUM(RENDA)FROM CLIENTE GROUP BY BAIRRO
```

C

```
SELECT BAIRRO, AVG(RENDA)FROM CLIENTE GROUP BY BAIRRO
```

D

```
SELECT BAIRRO,MAX(RENDA)FROM CLIENTE GROUP BY SEXO
```

E

```
SELECT BAIRRO, AVG(RENDA)FROM CLIENTE GROUP BY (SEXO)
```



A alternativa C está correta.

Para recuperar corretamente os registros de interesse, é necessário agrupar os dados pela coluna BAIRRO e em seguida usar a função de média (AVG), tendo como base a coluna RENDA.

## Grupo de dados com GROUP BY e HAVING

Vamos explorar técnicas avançadas de filtragem e agrupamento de dados em bancos de dados. Aprenderemos a utilizar a cláusula HAVING para aplicar condições de filtro a grupos de dados definidos pelo GROUP BY.

Essa combinação permite realizar análises mais detalhadas e refinadas, especialmente úteis em relatórios complexos e análises específicas.

Neste vídeo, vamos analisar como a cláusula HAVING aplica condições de filtro a grupos de dados definidos pelo GROUP BY.



### Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Até o momento, utilizamos a cláusula WHERE para programar filtros em consultas, com condições simples ou compostas envolvendo colunas da tabela ou funções de data.

Contudo, você vai vivenciar situações em que será necessário estabelecer algum tipo de filtro, tendo como base um cálculo originado a partir de uma função de agregação, não sendo possível usar a cláusula WHERE. Nesses casos, utilizamos a cláusula HAVING, que serve justamente para esse propósito.

Vamos ver a seguir um exemplo de quando utilizar essa cláusula.

## Consulta 05 + resultado

Suponha que o departamento de recursos humanos esteja estudando a viabilidade de oferecer bônus de 5% aos funcionários por mês de nascimento, mas limitado somente aos casos em que há mais de um colaborador aniversariando. Assim, para cada mês em questão, deseja-se listar o mês, o número de colaboradores e o valor do bônus.

Confira a solução!

```
sql

SELECT EXTRACT(MONTH FROM DTNASCIMENTO) AS MES,
       COUNT(*) AS QUANTIDADE,
       SUM(SALARIO*0.05) AS TOTALBONUS
FROM FUNCIONARIO
GROUP BY EXTRACT(MONTH FROM DTNASCIMENTO)
HAVING COUNT(*)>1
ORDER BY EXTRACT(MONTH FROM DTNASCIMENTO);
```

O resultado da consulta 05 está expresso na imagem a seguir.

	mes	quantidade	totalbonus
1	2	3	950

Resultado consulta 05.

Note que estamos diante de uma estrutura de consulta muito similar ao código da consulta 03. No entanto, estamos interessados em retornar somente o(s) registro(s) cujo valor da coluna quantidade seja maior que a unidade. Acontece que quantidade é uma coluna calculada com auxílio de uma função de agregação, não sendo possível programar um filtro na cláusula WHERE (WHERE QUANTIDADE>1). Assim, declaramos o filtro de interesse fazendo uso da cláusula HAVING, conforme linha 6 da consulta.



### Comentário

Ao longo da nossa jornada, estudamos o projeto de consultas com o uso de agrupamento de dados. Percebemos que esse recurso é imprescindível quando temos interesse na extração de informações de caráter mais analítico a partir de alguma tabela, fazendo uso de funções de agregação associadas a uma ou diversas colunas.

Ainda, percebemos que, às vezes, a natureza do problema que estamos resolvendo requer o uso de filtro tendo como base o uso de alguma função de agregação. Para isso, fizemos uso da cláusula HAVING.

Agora é com você! Vamos realizar as atividades a seguir?

## Atividade 2

Suponha a existência de uma tabela no PostgreSQL com a seguinte estrutura: PRODUTO (CODIGOP, NOME, ANO QUANTIDADE). Suponha também que a tabela tenha os seguintes registros:

CODIGO P	NOME	ANO	QUANTIDADE
1	VIRTUS	2020	3
2	FIESTA	2014	1
3	CRUZE	2020	4
4	CAMARO	2018	1
5	KOMBI	1996	4
6	FOCUS	2016	3

Tabela: PostgreSQL - PRODUTO (CODIGOP, NOME, ANO QUANTIDADE).  
Sidney Venturi.

Qual consulta a seguir retorna mais de dois resultados?

A

```
SELECT ANO,SUM(QUANTIDADE) AS TOTAL  
FROM PRODUTO  
GROUP BY ANO  
HAVING SUM(QUANTIDADE)>1;
```

B

```
SELECT ANO,SUM(QUANTIDADE) AS TOTAL  
FROM PRODUTO  
GROUP BY ANO;
```

C

```
SELECT SUM(QUANTIDADE) AS TOTAL  
FROM PRODUTO;
```

D

```
SELECT ANO, COUNT(*) AS TOTAL  
FROM PRODUTO  
WHERE QUANTIDADE>5  
GROUP BY ANO;
```

E

```
SELECT ANO,SUM(QUANTIDADE) AS TOTAL  
FROM PRODUTO
```

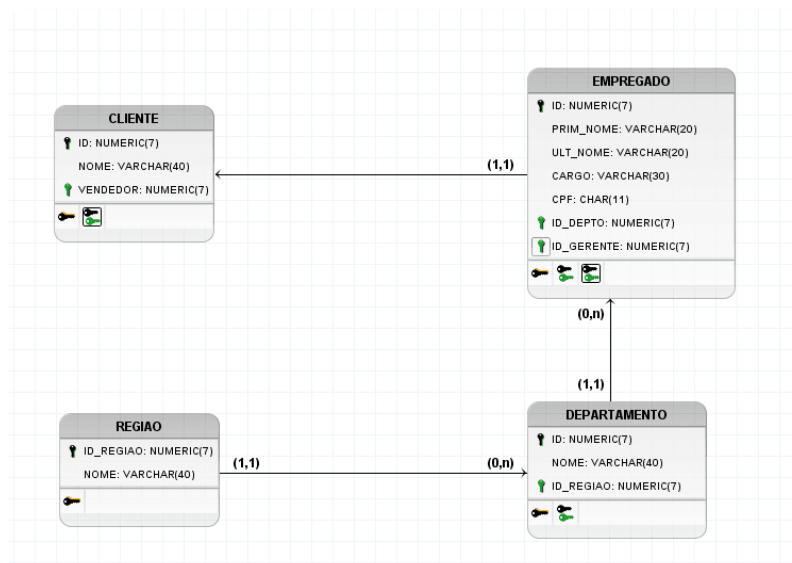


A alternativa A está correta.

Para recuperar os registros de interesse, a consulta em questão retorna o total de automóveis por ano, no entanto, levando em conta somente os grupos em que o total seja maior que 1. Na prática, os anos 2014 e 2018 não farão parte dos resultados da consulta e os demais o farão, totalizando três resultados.

## Consultas com GROUP BY e HAVING

Vamos ver exemplos de uso dos comandos vistos até aqui. Para isso, iremos utilizar o banco de dados da empresa cujo modelo lógico é:



Modelo de banco de dados de uma empresa.

Assista ao vídeo e coloque em prática como realizar as consultas em SQL com as cláusulas GROUP BY e HAVING. Não perca!



Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

## Roteiro de prática

Execute comandos seguindo as seguintes diretrizes:

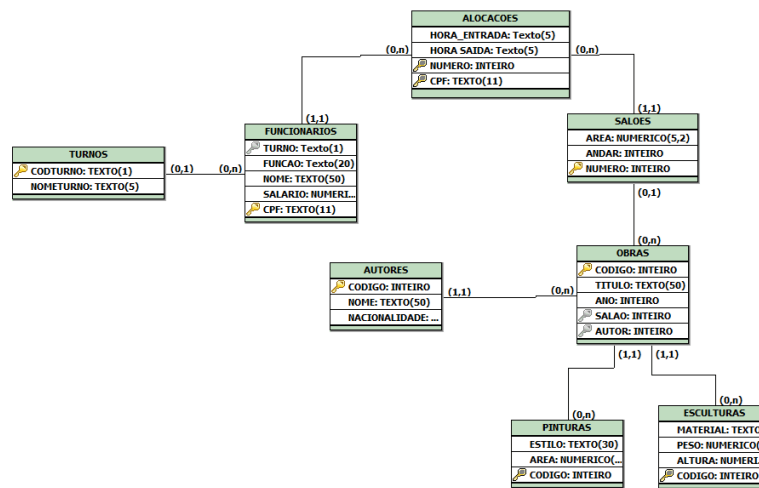
- Retornando cargos de emprego.
- Uso de DISTINCT.
- Uso de GROUP BY.

- Filtrar grupo com WHERE.
- Filtrar grupo com HAVING.
- Ordenando a consulta.

Você pode acessar os comandos que resolvem as consultas no arquivo [sqlmod3.3.txt](#).

## Atividade 3

Vamos executar uma atividade prática utilizando o banco de dados do museu cujo modelo lógico é:



Modelo lógico de banco de dados de um museu.

Escreva os seguintes comandos de consulta:

1. Listar a área média com o alias 'Área média', e número de salões com o alias 'Total de salões'.
2. Listar o estilo e o total de pinturas do estilo com o alias 'Quantidade de pinturas'.
3. Listar o estilo e o total de pinturas do estilo com o alias 'Quantidade de pinturas' para os estilos diferentes de cubista.
4. Listar o estilo e o total de pinturas do estilo com o alias 'Quantidade de pinturas' para os estilos diferentes de cubista e com pelo menos duas pinturas.
5. Listar os diferentes estilos de pintura.

Você pode acessar os comandos que resolvem as consultas no arquivo [exercmod3.3](#).

## Considerações finais

- Estrutura básica de um comando SELECT.
- Funções de agregação.
- Criação de tabela view a partir de consulta.
- Recuperação de dados com SELECT/FROM/WHERE/ORDER BY.
- Uso dos operadores IN, BETWEEN, LIKE e NULL.
- Agrupamento de dados com GROUP BY e HAVING.

## Explore +

Para aprofundar os seus conhecimentos sobre o assunto, leia:

**Tarefas comuns do administrador de banco de dados para PostgreSQL**, um interessante material sobre o dia a dia de um administrador de banco de dados. Você pode encontrá-lo no site da Amazon Web Services.

**EasyRA: uma ferramenta para tradução de consultas em álgebra relacional para SQL**, de Bilecki e Kalempa. Nesse artigo, você aprenderá sobre as operações básicas da álgebra relacional e conhecerá uma ferramenta para praticar comandos em álgebra e visualizá-los na linguagem SQL.

## Referências

AWS. **Tarefas comuns do administrador de banco de dados para PostgreSQL**. *In*: AWS. Consultado em meio eletrônico em: 30 mai. 2020.

BILECKI, L. F.; KALEMPA, V. C. **EasyRA**: Uma ferramenta para tradução de consultas em álgebra relacional para SQL. *In*: Computer On The Beach, 2015, Florianópolis. Computer on the Beach, 2015. p. 21-30.

CAMPOS, N. S. **Notas de Aula sobre Banco de Dados da professora Nathielly Campos**.

Disponível sob licença Creative Commons BR Atribuição – CC BY, 2020.

ELMASRI, R.; NAVATHE, S. **Sistemas de Banco de Dados**. 7. ed. São Paulo: Pearson, 2019.

POSTGRESQL. **Chapter 9. Functions and Operators**. *In*: PostgreSQL. Consultado em meio eletrônico em: 30 mai. 2020.

POSTGRESQL. **PostgreSQL Downloads**. *In*: PostgreSQL. Consultado em meio eletrônico em: 30 mai. 2020.

POSTGRESQL. **SELECT**. *In*: PostgreSQL. Consultado em meio eletrônico em: 30 mai. 2020.