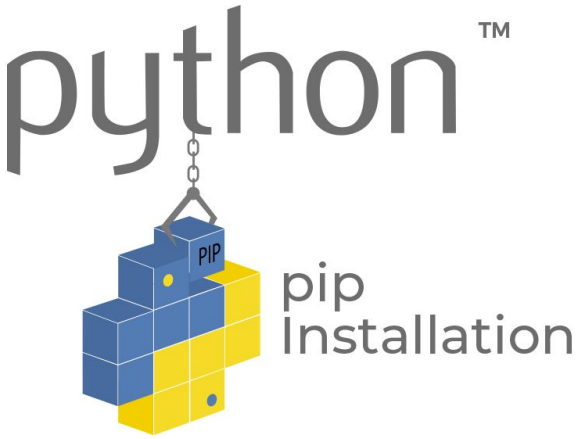
Banco de Dados com python

SQL e python

- Para utilizar SQL com Python, é essencial contar com bibliotecas especializadas que facilitam a integração entre a linguagem e o banco de dados.
- Essas bibliotecas permitem estabelecer conexões seguras, executar consultas SQL e recuperar os resultados de forma eficiente e estruturada.

PIP

- No python utilizamos o `pip` para gerenciar bibliotecas.
- `pip` é o gerenciador de pacotes oficial do Python. Ele permite instalar, atualizar e remover bibliotecas e ferramentas que ampliam as funcionalidades do Python.



PIP

- Verificando se o `pip` está instalado.

```
pip --version
```

- Se estiver instalado irá retornar a versão.
- Caso não esteja irá retornar uma mensagem de erro.

PIP

- Instalação
  - Instalando o `pip` no windows.
  - Baixe o script:

    https://bootstrap.pypa.io/get-pip.py

  - Execute como um arquivo python.
  - Via terminal ou via vscode.

```
python get-pip.py
```

PIP

- Instalação
  - Instalando o `pip` no windows.

```
 WARNING: The scripts pip.exe, pip3.13.exe and pip3.exe are installed in
'C:\Users\Fulano\AppData\Local\Programs\Python\Python313\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning,
use --no-warn-script-location.
```

- Se o aviso acima for exibido, indica que o pip foi instalado na pasta do usuário, mas não está incluído na variável de ambiente “Path” no sistema.
- Por esse motivo, para executar o pip corretamente, é necessário acessar diretamente o diretório onde ele foi instalado ou adicionar esse caminho manualmente às variáveis de ambiente do sistema sendo administrador.

PIP

- Instalação
  - Instalando o `pip` no windows.
  - Acessando a pasta de instalação e testando o `pip`.
  - Via CMD ou PowerShell.

```
PS C:\Windows\System32> cd C:\Users\Fulano\AppData\Local\Programs\Python\Python313\Scripts

PS C:\Users\Fulano\AppData\Local\Programs\Python\Python313\Scripts> .\pip.exe --version

pip 25.2 from C:\Users\Fulano\AppData\Local\Programs\Python\Python313\Lib\site-packages\pip (python 3.13)
```

PIP

- Instalação um pacote/biblioteca
  - Instalando o psycopg2. Uma biblioteca para manipulação SQL.

Comando

```
PS C:\Users\Fulano\AppData\Local\Programs\Python\Python313\Scripts> .\pip.exe install psycopg2-binary
```

Saída

```
Collecting psycopg2-binary
  Downloading psycopg2_binary-2.9.11-cp313-cp313-win_amd64.whl.metadata (5.1 kB)
Downloading psycopg2_binary-2.9.11-cp313-cp313-win_amd64.whl (2.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.7/2.7 MB 17.3 MB/s  0:00:00
Installing collected packages: psycopg2-binary
Successfully installed psycopg2-binary-2.9.11
```

Agora que a biblioteca está instalada, podemos importar nos códigos.

SQL e python

- Conectando no banco.
  - Variável **conn** (*pode ter qualquer nome*)
    - É necessário criar uma variável que represente a conexão, e passar os dados para conectar na base.

```
import psycopg2

conn = psycopg2.connect(
    dbname="meubanco",
    user="meuusuario",
    password="minhasenha",
    host="localhost",
    port="5432"
)
cursor = conn.cursor()
```

SQL e python

- Conectando no banco.
  - Variável **cursor** (*pode ter qualquer nome*)
    - O cursor é o objeto responsável por executar comandos SQL e manipular os resultados dentro da conexão com o banco.
    - Ele funciona como um "canal" entre o Python e o PostgreSQL.
  - Funções do cursor
    - Executar comandos SQL: `execute` e `executemany`.
    - Enviar parâmetros com segurança: evitar SQL Injection.
    - Buscar resultados: com métodos como, `fetchone, fetchall`.
    - Controlar transações: em conjunto com e `commit()` e `rollback()`.

```
cursor = conn.cursor()
```

SQL e python

- Executando comandos SQL
  - INSERT

```
cursor.execute("""
    INSERT INTO clientes (nome, email)
    VALUES (%s, %s) # << -------------------- placeholders (%s,%s), posições para inserir
valores
""", ("Fulano", "fulano@email.com")) # << -----tupla de valores para os placeholders (%s,%s)
```

  - SELECT

```
cursor.execute("SELECT id, nome, email FROM clientes")
resultados = cursor.fetchall()

for linha in resultados:
    print(linha)
```

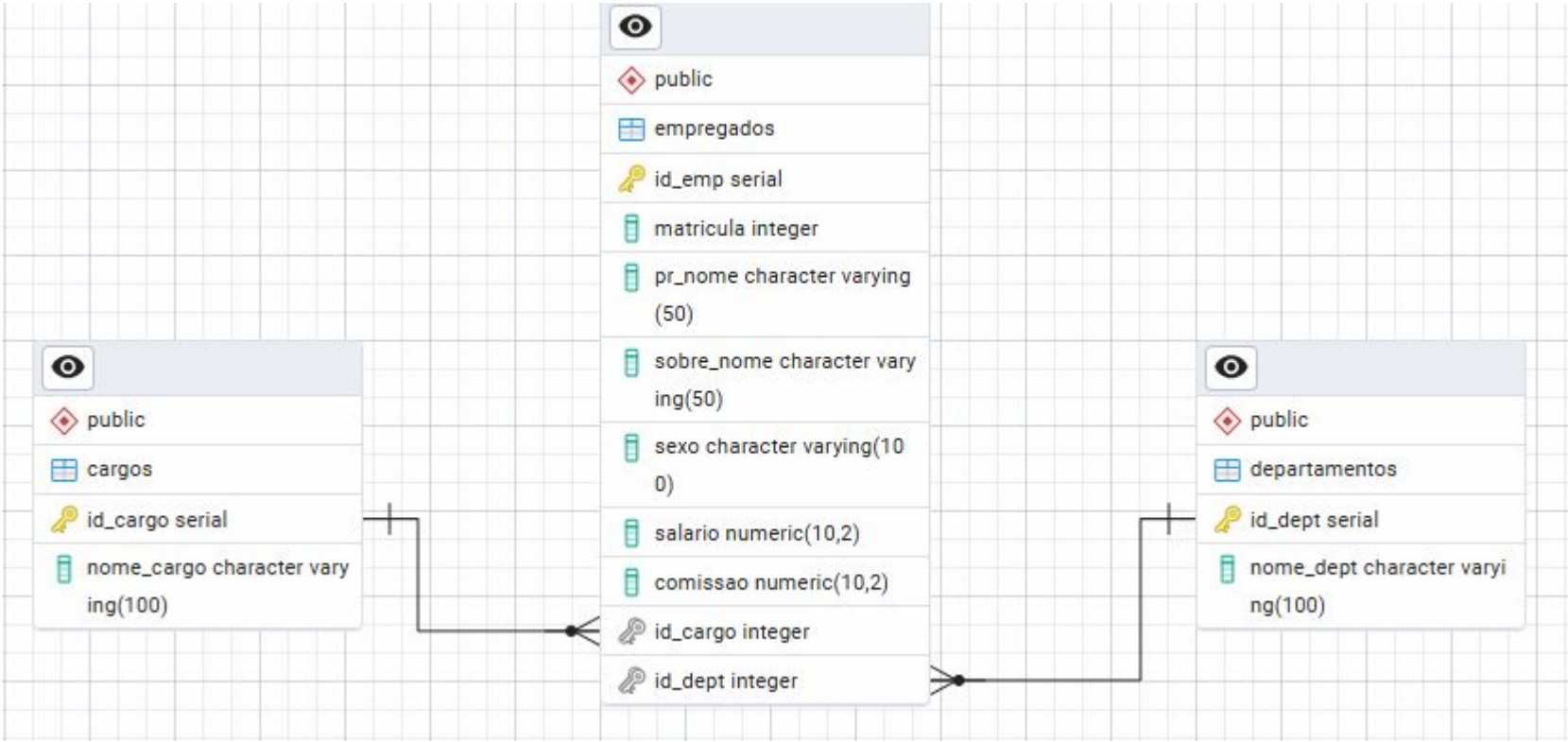
SQL e python

- Exemplos práticos:
  - Criando a estrutura para os testes (caso não exista) via **pgAdmin**.

```
--Criando tabelas
CREATE TABLE departamentos (
    id_dept SERIAL PRIMARY KEY,
    nome_dept VARCHAR(100) NOT NULL
);
CREATE TABLE cargos (
    id_cargo SERIAL PRIMARY KEY,
    nome_cargo VARCHAR(100) NOT NULL
);
CREATE TABLE empregados(
    id_emp SERIAL PRIMARY KEY,
    matricula INTEGER NOT NULL,
    pr_nome VARCHAR(50) NOT NULL,
    sobre_nome VARCHAR(50) NOT NULL,
    sexo VARCHAR(100),
    salario NUMERIC(10,2),
    comissao NUMERIC(10,2),
    id_cargo INTEGER REFERENCES cargos (id_cargo),
    id_dept INTEGER REFERENCES departamentos (id_dept)
);
```

SQL e python

- MER:



SQL e python

- Populando as tabelas via **pgAdmin**.

```
--populando tabelas

--departamentos

INSERT INTO departamentos (nome_dept)
VALUES
('Vendas'),('TI'),
('Gerencia'),('Atendimento');
SELECT * FROM departamentos;

--cargos
INSERT INTO cargos (nome_cargo)
VALUES
('Atendente'),('Tecnico TI'),('Vendedor'),
('Contador'),('Gerente de Vendas');
SELECT * FROM cargos;
```

SQL e python

- Populando as tabelas via **pgAdmin**.

```sql
--empregados

INSERT INTO empregados (matricula, pr_nome, sobre_nome, sexo, salario, comissao, id_cargo, id_dept)
VALUES (1001, 'João', 'Silva', 'M', 4500.00, NULL, 2, 2); -- 'Tecnico TI, TI

INSERT INTO empregados (matricula, pr_nome, sobre_nome, sexo, salario, comissao, id_cargo, id_dept)
VALUES (1002, 'Maria', 'Oliveira', 'F', 7000.00, NULL, 5, 3); -- Gerente, Gerencia

INSERT INTO empregados (matricula, pr_nome, sobre_nome, sexo, salario, comissao, id_cargo, id_dept)
VALUES (1003, 'Carlos', 'Souza', 'M', 3000.00, 300.00, 3, 1); -- Vendedor, Vendas

INSERT INTO empregados (matricula, pr_nome, sobre_nome, sexo, salario, comissao, id_cargo, id_dept)
VALUES (1004, 'Fernanda', 'Lima', 'F', 12000.00, NULL, 4, 3); -- Contador, Gerencia

INSERT INTO empregados (matricula, pr_nome, sobre_nome, sexo, salario, comissao, id_cargo, id_dept)
VALUES (1005, 'Bruna', 'Costa', 'F', 1500.00, NULL, 2, 2); -- 'Tecnico TI, TI

INSERT INTO empregados (matricula, pr_nome, sobre_nome, sexo, salario, comissao, id_cargo, id_dept)
VALUES (1006, 'Rafael', 'Mendes', 'M', 3200.00, 250.00, 1, 4); -- Atendente, Atendimento
```

SQL e python

- Visualizando via **pgAdmin**.

```
SELECT
  e.id_emp, e.matricula, pr_nome, e.sobre_nome, sexo, e.salario, e.comissao, c.nome_cargo, d.nome_dept
FROM
  empregados e
JOIN
  cargos c ON e.id_cargo = c.id_cargo
JOIN
  departamentos d ON e.id_dept = d.id_dept;
```

	id_emp integer	matricula integer	pr_nome character varying (50)	sobre_nome character varying (50)	sexo character varying (100)	salario numeric (10,2)	comissao numeric (10,2)	nome_cargo character varying (100)	nome_dept character varying (100)
1	6	1006	Rafael	Mendes	M	3200.00	250.00	Atendente	Atendimento
2	5	1005	Bruna	Costa	F	1500.00	[null]	Tecnico TI	TI
3	1	1001	João	Silva	M	4500.00	[null]	Tecnico TI	TI
4	3	1003	Carlos	Souza	M	3000.00	300.00	Vendedor	Vendas
5	4	1004	Fernanda	Lima	F	12000.00	[null]	Contador	Gerencia
6	2	1002	Maria	Oliveira	F	7000.00	[null]	Gerente de Vendas	Gerencia

SQL e python - SELECT

Exemplo - 1

- Select com python: Retornar dados da tabela empregados

```python
import psycopg2

try:
    # Conectando ao banco de dados
    conn = psycopg2.connect(
        dbname="postgres",      # substituir pelos dados do seu banco
        user="postgres",
        password="aluno",
        host="10.82.0.81",
        port="5432"
    )

    # Criando o cursor
    cursor = conn.cursor()

    # Executando o SELECT
    cursor.execute("""
        SELECT id_emp, matricula, pr_nome, sobre_nome, sexo, salario, comissao
        FROM empregados
    """)
```

SQL e python - SELECT

Exemplo - 1

- Select com python: Retornar dados da tabela empregados

```
    # Recuperando os resultados
    resultados = cursor.fetchall()

    # Exibindo os dados
    for res in resultados:
        print(f"ID: {res[0]}, Matrícula: {res[1]}, Nome: {res[2]} {res[3]}, Sexo: {res[4]}, Salário:
R${res[5]}, Comissão: {res[6]}")

except Exception as e:
    print("Erro ao acessar o banco:", e)

# Fechando conexões
if cursor:
    cursor.close()
if conn:
    conn.close()
```

SQL e python - SELECT

- Resultado via **pgAdmin**.

	id_emp [PK] integer	matricula integer	pr_nome character varying (50)	sobre_nome character varying (50)	sexo character varying (100)	salario numeric (10,2)	comissao numeric (10,2)
1	1	1001	João	Silva	M	4500.00	[null]
2	2	1002	Maria	Oliveira	F	7000.00	[null]
3	3	1003	Carlos	Souza	M	3000.00	300.00
4	4	1004	Fernanda	Lima	F	12000.00	[null]
5	5	1005	Bruna	Costa	F	1500.00	[null]
6	6	1006	Rafael	Mendes	M	3200.00	250.00

- Resultado via **Python**.

```
ID: 1, Matrícula: 1001, Nome: João Silva, Sexo: M, Salário: R$4500.00, Comissão: None
ID: 2, Matrícula: 1002, Nome: Maria Oliveira, Sexo: F, Salário: R$7000.00, Comissão: None
ID: 3, Matrícula: 1003, Nome: Carlos Souza, Sexo: M, Salário: R$3000.00, Comissão: 300.00
ID: 4, Matrícula: 1004, Nome: Fernanda Lima, Sexo: F, Salário: R$12000.00, Comissão: None
ID: 5, Matrícula: 1005, Nome: Bruna Costa, Sexo: F, Salário: R$1500.00, Comissão: None
ID: 6, Matrícula: 1006, Nome: Rafael Mendes, Sexo: M, Salário: R$3200.00, Comissão: 250.00
```

SQL e python - SELECT

- Select para retornar o nome do funcionário, o cargo e o departamento.
- Select com pgAdmin

```
SELECT
    e.pr_nome || ' ' || e.sobre_nome AS nome_funcionario,
    c.nome_cargo,
    d.nome_dept
FROM empregados e
JOIN cargos c ON e.id_cargo = c.id_cargo
JOIN departamentos d ON e.id_dept = d.id_dept;
```

SQL e python - SELECT

Exemplo - 2

- Select com python

```
import psycopg2
try:
    # Conectando ao banco de dados
    conn = psycopg2.connect(
        dbname="postgres",
        user="postgres",
        password="aluno",
        host="localhost",
        port="5432"
    )
    # Criando o cursor
    cursor = conn.cursor()
    # Executando o SELECT com JOIN
    cursor.execute("""
        SELECT
            e.pr_nome || ' ' || e.sobre_nome AS nome_funcionario,
            c.nome_cargo,
            d.nome_dept
        FROM empregados e
        JOIN cargos c ON e.id_cargo = c.id_cargo
        JOIN departamentos d ON e.id_dept = d.id_dept;
    """)
```

SQL e python - SELECT

Exemplo - 2

- Select com python

```
    # Recuperando os resultados
    resultados = cursor.fetchall()

    # Exibindo os dados
    for nome, cargo, departamento in resultados:
        print(f"Funcionário: {nome} | Cargo: {cargo} | Departamento: {departamento}")

except Exception as e:
    print("Erro ao consultar o banco:", e)


# Fechando conexões
if cursor:
    cursor.close()
if conn:
    conn.close()
```

SQL e python - SELECT

- Resultado via **pgAdmin**.

	nome_funcionario text	nome_cargo character varying (100)	nome_dept character varying (100)
1	Rafael Mendes	Atendente	Atendimento
2	Bruna Costa	Tecnico TI	TI
3	João Silva	Tecnico TI	TI
4	Carlos Souza	Vendedor	Vendas
5	Fernanda Lima	Contador	Gerencia
6	Maria Oliveira	Gerente de Vendas	Gerencia

- Resultado via **Python**.

```
Funcionário: Rafael Mendes | Cargo: Atendente | Departamento: Atendimento
Funcionário: Bruna Costa | Cargo: Tecnico TI | Departamento: TI
Funcionário: João Silva | Cargo: Tecnico TI | Departamento: TI
Funcionário: Carlos Souza | Cargo: Vendedor | Departamento: Vendas
Funcionário: Fernanda Lima | Cargo: Contador | Departamento: Gerencia
Funcionário: Maria Oliveira | Cargo: Gerente de Vendas | Departamento: Gerencia
```

Atividade de fixação 1 - SELECT

- Select para retornar dados das 3 tabelas.
- Select com pgAdmin

```
SELECT
  e.id_emp, e.matricula, pr_nome, e.sobre_nome, sexo, e.salario, e.comissao, c.nome_cargo,
d.nome_dept
FROM
  empregados e
JOIN
  cargos c ON e.id_cargo = c.id_cargo
JOIN
  departamentos d ON e.id_dept = d.id_dept;
```

- Implementar select com python.

SQL e python - INSERT

- Insert: inserir 3 entradas na tabela empregados:
- Insert com pgAdmin:

```
INSERT INTO empregados (
    matricula, pr_nome, sobre_nome, sexo, salario, comissao, id_cargo, id_dept
) VALUES (
    1007, 'Inocêncio', 'Coitadinho da Silva', 'M', 5000.00, NULL, 4, 3
);

INSERT INTO empregados (
    matricula, pr_nome, sobre_nome, sexo, salario, comissao, id_cargo, id_dept
) VALUES (
    1008, 'Luciana', 'Ferraz', 'F', 3800.00, 200.00, 3, 1
);

INSERT INTO empregados (
    matricula, pr_nome, sobre_nome, sexo, salario, comissao, id_cargo, id_dept
) VALUES (
    1009, 'Eduardo', 'Pereira', 'M', 2500.00, NULL, 1, 4
);
```

SQL e python - INSERT

Exemplo - 3

- Insert com python:

```
import psycopg2

try:
    # Conectando ao banco de dados
    conn = psycopg2.connect(
        dbname="postgres",
        user="postgres",
        password="aluno",
        host="localhost",
        port="5432"
    )

    cursor = conn.cursor()

    # Lista de empregados a serem inseridos
    empregados = [
        (1007, 'Inocêncio', 'Coitadinho da Silva', 'M', 5000.00, None, 4, 3),
        (1008, 'Luciana', 'Ferraz', 'F', 3800.00, 200.00, 3, 1),
        (1009, 'Eduardo', 'Pereira', 'M', 2500.00, None, 1, 4)
    ]
```

SQL e python - INSERT

Exemplo - 3

- Insert com python:

```
    # Executando os INSERTs com executemany para inserir várias de uma vez
    cursor.executemany("""
        INSERT INTO empregados (
            matricula, pr_nome, sobre_nome, sexo, salario, comissao, id_cargo, id_dept
        ) VALUES (%s, %s, %s, %s, %s, %s, %s, %s)
    """, empregados)

    conn.commit()
    print("Empregados inseridos com sucesso!")

except Exception as e:
    print("Erro ao inserir empregados:", e)
    conn.rollback()

if cursor:
    cursor.close()
if conn:
    conn.close()
```

```
Empregados inseridos com sucesso!
PS C:\Windows\System32>
```

SQL e python - INSERT

- Para visualizar o resultado execute o código python no slide 21.
- Resultado via **pgAdmin**.

	nome_funcionario text	nome_cargo character varying (100)	nome_dept character varying (100)
1	Eduardo Pereira	Atendente	Atendimento
2	Rafael Mendes	Atendente	Atendimento
3	Bruna Costa	Tecnico TI	TI
4	João Silva	Tecnico TI	TI
5	Luciana Ferraz	Vendedor	Vendas
6	Carlos Souza	Vendedor	Vendas
7	Inocêncio Coitadinho da Sil...	Contador	Gerencia
8	Fernanda Lima	Contador	Gerencia
9	Maria Oliveira	Gerente de Vendas	Gerencia

- Resultado via **python.**

```
Funcionário: Eduardo Pereira | Cargo: Atendente | Departamento: Atendimento
Funcionário: Rafael Mendes | Cargo: Atendente | Departamento: Atendimento
Funcionário: Bruna Costa | Cargo: Tecnico TI | Departamento: TI
Funcionário: João Silva | Cargo: Tecnico TI | Departamento: TI
Funcionário: Luciana Ferraz | Cargo: Vendedor | Departamento: Vendas
Funcionário: Carlos Souza | Cargo: Vendedor | Departamento: Vendas
Funcionário: Inocêncio Coitadinho da Silva | Cargo: Contador | Departamento: Gerencia
Funcionário: Fernanda Lima | Cargo: Contador | Departamento: Gerencia
Funcionário: Maria Oliveira | Cargo: Gerente de Vendas | Departamento: Gerencia
```

SQL e python - Delete

- Delete: apagar 2 entradas na tabela empregados:
- Delete com pgAdmin:

```
DELETE FROM empregados WHERE id_emp = 12;
DELETE FROM empregados WHERE id_emp = 13;
```

SQL e python - Delete

Exemplo - 4

- Delete com python:

```python
import psycopg2
try:
    # Conectando ao banco de dados
    conn = psycopg2.connect(
        dbname="postgres",
        user="postgres",
        password="aluno",
        host="localhost",
        port="5432"    )
    cursor = conn.cursor()
    # Executando os DELETEs
    cursor.execute("DELETE FROM empregados WHERE id_emp = %s", (14,)) # Verificar os IDs na tabela e substituir aqui
    cursor.execute("DELETE FROM empregados WHERE id_emp = %s", (15,))
    conn.commit()
    print("Registros excluídos com sucesso!")
except Exception as e:
    print("Erro ao excluir registros:", e)
    conn.rollback()
if cursor:
    cursor.close()
if conn:
    conn.close()
```

SQL e python - DELETE

- Para visualizar o resultado execute o código python no slide 21.
- Resultado via **pgAdmin**.

	nome_funcionario text	nome_cargo character varying (100)	nome_dept character varying (100)
1	Rafael Mendes	Atendente	Atendimento
2	Bruna Costa	Tecnico TI	TI
3	João Silva	Tecnico TI	TI
4	Carlos Souza	Vendedor	Vendas
5	Inocêncio Coitadinho da Sil...	Contador	Gerencia
6	Fernanda Lima	Contador	Gerencia
7	Maria Oliveira	Gerente de Vendas	Gerencia

- Resultado via **python.**

```
Funcionário: Rafael Mendes | Cargo: Atendente | Departamento: Atendimento
Funcionário: Bruna Costa | Cargo: Tecnico TI | Departamento: TI
Funcionário: João Silva | Cargo: Tecnico TI | Departamento: TI
Funcionário: Carlos Souza | Cargo: Vendedor | Departamento: Vendas
Funcionário: Inocêncio Coitadinho da Silva | Cargo: Contador | Departamento: Gerencia
Funcionário: Fernanda Lima | Cargo: Contador | Departamento: Gerencia
Funcionário: Maria Oliveira | Cargo: Gerente de Vendas | Departamento: Gerencia
```

SQL e python - Update

- Update: atualizar 1 entrada na tabela empregados:
- Atualizar salário para R$ 5500 e comissão para R$ 400 para o empregado Carlos.
- Update com pgAdmin:

```
UPDATE empregados
SET salario = 5500, comissao = 400
WHERE id_emp = 3; # Verificar ID do funcionário na tabela
```

SQL e python - Update

Exemplo - 5

- Delete com python:

```python
import psycopg2

try:
    # Conectando ao banco de dados
    conn = psycopg2.connect(
        dbname="postgres",
        user="postgres",
        password="aluno",
        host="localhost",
        port="5432"
    )
    cursor = conn.cursor()

    # Atualizando salário e comissão
    cursor.execute("""
        UPDATE empregados
        SET salario = %s, comissao = %s
        WHERE id_emp = %s
    """, (5500.00, 400.00, 3)) # Verificar ID do funcionário na tabela
```

SQL e python - Update

Exemplo - 5

- Delete com python:

```
    conn.commit()
    print("Empregado atualizado com sucesso!")

except Exception as e:
    print("Erro ao atualizar empregado:", e)
    conn.rollback()

if cursor:
    cursor.close()
if conn:
    conn.close()
```

SQL e python - Update

- Para visualizar o resultado execute o código python no slide 17.
- Resultado via **pgAdmin**.

	id_emp [PK] integer	matricula integer	pr_nome character varying (50)	sobre_nome character varying (50)	sexo character varying (100)	salario numeric (10,2)	comissao numeric (10,2)
1	1	1001	João	Silva	M	4500.00	[null]
2	2	1002	Maria	Oliveira	F	7000.00	[null]
3	4	1004	Fernanda	Lima	F	12000.00	[null]
4	5	1005	Bruna	Costa	F	1500.00	[null]
5	6	1006	Rafael	Mendes	M	3200.00	250.00
6	11	1007	Inocêncio	Coitadinho da Silva	M	5000.00	[null]
7	3	1003	Carlos	Souza	M	5500.00	400.00

- Resultado via **python.**

```
ID: 1, Matrícula: 1001, Nome: João Silva, Sexo: M, Salário: R$4500.00, Comissão: None
ID: 2, Matrícula: 1002, Nome: Maria Oliveira, Sexo: F, Salário: R$7000.00, Comissão: None
ID: 4, Matrícula: 1004, Nome: Fernanda Lima, Sexo: F, Salário: R$12000.00, Comissão: None
ID: 5, Matrícula: 1005, Nome: Bruna Costa, Sexo: F, Salário: R$1500.00, Comissão: None
ID: 6, Matrícula: 1006, Nome: Rafael Mendes, Sexo: M, Salário: R$3200.00, Comissão: 250.00
ID: 11, Matrícula: 1007, Nome: Inocêncio Coitadinho da Silva, Sexo: M, Salário: R$5000.00, Comissão: None
ID: 3, Matrícula: 1003, Nome: Carlos Souza, Sexo: M, Salário: R$5500.00, Comissão: 400.00
```

Atividade de fixação 2 - INSERT - DELETE - UPDATE

- Inserir um novo registro na tabela cargos.
- Inserir um novo registro na tabela departamentos.
- Apagar um registro da tabela empregados com base no `id_emp`.
- Atualizar um registro da tabela empregados.

```sql
INSERT INTO cargos (nome_cargo)
VALUES ('Analista de Dados');

INSERT INTO departamentos (nome_dept)
VALUES ('Pesquisa e Desenvolvimento');

DELETE FROM empregados
WHERE id_emp =14; --verificar ID na tabela

UPDATE empregados
SET pr_nome = 'João Paulo' Where id_emp = 1; --verificar ID na tabela
```

- Implementar o código em python.

Continue os estudos

Continue os estudos fora de sala!

Considere refazer alguns dos códigos, dessa vez utilizando funções e menus interativos.

FIM