Projeto de Bloco: Fundamentos de Dados

Professor: *Alcione Santos Dolavale*

Kaike Torres da silva

03.11.2024

Análise e Desenvolvimento de Sistemas (ADS)

INTRODUÇÃO

Este projeto, TP3 - Fundamento de Dados, tem como objetivo criar uma base com 5 tabelas: Funcionários, Cargos, Departamentos, Histórico Salarial e Dependentes. As tabelas estão relacionadas e contêm dados como nome, cargo, salário e departamento. Cada tabela possui um campo extra, e o banco foi criado no SQLite com Python e `sqlite3`. Registros suficientes foram inseridos via CSV para consultas que exibem dados como descrição do cargo e nome do departamento, destacando o uso de chaves estrangeiras.

link para o projeto:https://replit.com/@kaikesilva2/kaiketorresTP3#main.py

RESULTADOS

1. Listar individualmente as tabelas de: Funcionários, Cargos, Departamentos, Histórico de Salários e Dependentes em ordem crescente.

Screenshots:

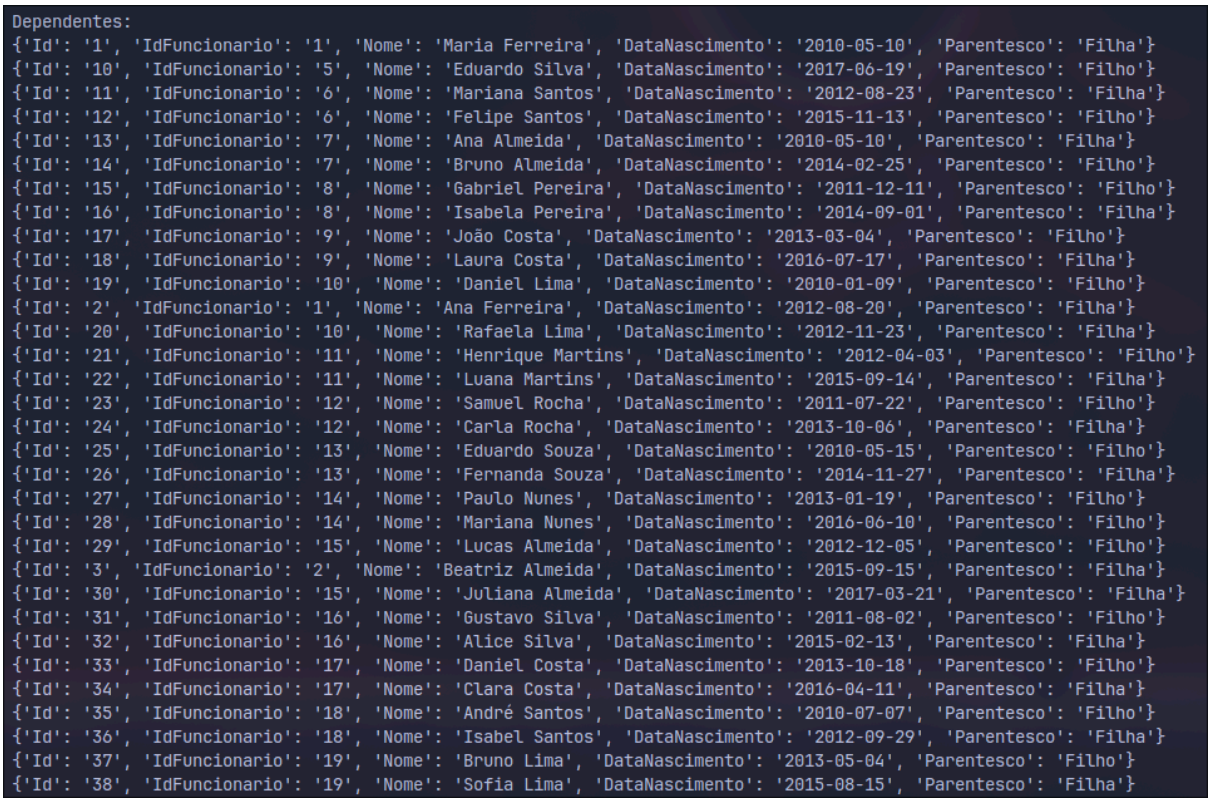
```
print('-'*50)
print('1 - S/SQL - Listar individualmente as tabelas de: Funcionários, Cargos, Departamentos, Histórico
de Salários e Dependentes em ordem crescente.')
funcionarios = ler_arquivo_csv('CSV/TB_FUNCIONARIO.csv')
cargos = ler_arquivo_csv('CSV/TB_CARGO.csv')
departamentos = ler_arquivo_csv('CSV/TB_DEPARTAMENTO.csv')
historico_salarios = ler_arquivo_csv('CSV/TB_HISTORICO_SALARIO.csv')
dependentes = ler_arquivo_csv('CSV/TB_DEPENDENTE.csv')

# Listar os dados em ordem crescente
print("Funcionários:")
for func in sorted(funcionarios, key=lambda x: x['Id']):
    print(func)

print("\nCargos:")
for cargo in sorted(cargos, key=lambda x: x['Id']):
    print(cargo)

print("\nDepartamentos:")
for depa in sorted(departamentos, key=lambda x: x['Id']):
    print(depa)

print("\nHistórico de Salários:")
for hist in sorted(historico_salarios, key=lambda x: x['IdFuncionario']):
    print(hist)
```

Explicação:

1. **Exibição do Cabeçalho:**

- O código imprime uma linha divisória e uma descrição do processo que será realizado.

2. **Leitura dos Arquivos CSV:**
   - A função `ler_arquivo_csv` é usada para carregar os dados de cada tabela, armazenando-os em listas onde cada item é um dicionário representando uma linha do arquivo CSV.
3. **Ordenação e Exibição dos Dados:**
   - Para cada tabela, os dados são organizados em ordem crescente usando a função `sorted()`, de acordo com uma chave específica (por exemplo, `Id` para Funcionários, `Id` para Cargos, etc.). Um loop `for` percorre a lista ordenada e exibe cada registro.
4. **Processo Repetido para Outras Tabelas:**
   - O procedimento de listagem é repetido para as tabelas de Cargos, Departamentos, Histórico de Salários e Dependentes, garantindo que todas as informações sejam exibidas de forma ordenada e clara.



```
Dependentes:
{'Id': '1', 'IdFuncionario': '1', 'Nome': 'Maria Ferreira', 'DataNascimento': '2010-05-10', 'Parentesco': 'Filha'}
{'Id': '10', 'IdFuncionario': '5', 'Nome': 'Eduardo Silva', 'DataNascimento': '2017-06-19', 'Parentesco': 'Filho'}
{'Id': '11', 'IdFuncionario': '6', 'Nome': 'Mariana Santos', 'DataNascimento': '2012-08-23', 'Parentesco': 'Filha'}
{'Id': '12', 'IdFuncionario': '6', 'Nome': 'Felipe Santos', 'DataNascimento': '2015-11-13', 'Parentesco': 'Filho'}
{'Id': '13', 'IdFuncionario': '7', 'Nome': 'Ana Almeida', 'DataNascimento': '2010-05-10', 'Parentesco': 'Filha'}
{'Id': '14', 'IdFuncionario': '7', 'Nome': 'Bruno Almeida', 'DataNascimento': '2014-02-25', 'Parentesco': 'Filho'}
{'Id': '15', 'IdFuncionario': '8', 'Nome': 'Gabriel Pereira', 'DataNascimento': '2011-12-11', 'Parentesco': 'Filho'}
{'Id': '16', 'IdFuncionario': '8', 'Nome': 'Isabela Pereira', 'DataNascimento': '2014-09-01', 'Parentesco': 'Filha'}
{'Id': '17', 'IdFuncionario': '9', 'Nome': 'João Costa', 'DataNascimento': '2013-03-04', 'Parentesco': 'Filho'}
{'Id': '18', 'IdFuncionario': '9', 'Nome': 'Laura Costa', 'DataNascimento': '2016-07-17', 'Parentesco': 'Filha'}
{'Id': '19', 'IdFuncionario': '10', 'Nome': 'Daniel Lima', 'DataNascimento': '2010-01-09', 'Parentesco': 'Filho'}
{'Id': '2', 'IdFuncionario': '1', 'Nome': 'Ana Ferreira', 'DataNascimento': '2012-08-20', 'Parentesco': 'Filha'}
{'Id': '20', 'IdFuncionario': '10', 'Nome': 'Rafaela Lima', 'DataNascimento': '2012-11-23', 'Parentesco': 'Filha'}
{'Id': '21', 'IdFuncionario': '11', 'Nome': 'Henrique Martins', 'DataNascimento': '2012-04-03', 'Parentesco': 'Filho'}
{'Id': '22', 'IdFuncionario': '11', 'Nome': 'Luana Martins', 'DataNascimento': '2015-09-14', 'Parentesco': 'Filha'}
{'Id': '23', 'IdFuncionario': '12', 'Nome': 'Samuel Rocha', 'DataNascimento': '2011-07-22', 'Parentesco': 'Filho'}
{'Id': '24', 'IdFuncionario': '12', 'Nome': 'Carla Rocha', 'DataNascimento': '2013-10-06', 'Parentesco': 'Filha'}
{'Id': '25', 'IdFuncionario': '13', 'Nome': 'Eduardo Souza', 'DataNascimento': '2010-05-15', 'Parentesco': 'Filho'}
{'Id': '26', 'IdFuncionario': '13', 'Nome': 'Fernanda Souza', 'DataNascimento': '2014-11-27', 'Parentesco': 'Filha'}
{'Id': '27', 'IdFuncionario': '14', 'Nome': 'Paulo Nunes', 'DataNascimento': '2013-01-19', 'Parentesco': 'Filho'}
{'Id': '28', 'IdFuncionario': '14', 'Nome': 'Mariana Nunes', 'DataNascimento': '2016-06-10', 'Parentesco': 'Filho'}
{'Id': '29', 'IdFuncionario': '15', 'Nome': 'Lucas Almeida', 'DataNascimento': '2012-12-05', 'Parentesco': 'Filho'}
{'Id': '3', 'IdFuncionario': '2', 'Nome': 'Beatriz Almeida', 'DataNascimento': '2015-09-15', 'Parentesco': 'Filha'}
{'Id': '30', 'IdFuncionario': '15', 'Nome': 'Juliana Almeida', 'DataNascimento': '2017-03-21', 'Parentesco': 'Filha'}
{'Id': '31', 'IdFuncionario': '16', 'Nome': 'Gustavo Silva', 'DataNascimento': '2011-08-02', 'Parentesco': 'Filho'}
{'Id': '32', 'IdFuncionario': '16', 'Nome': 'Alice Silva', 'DataNascimento': '2015-02-13', 'Parentesco': 'Filha'}
{'Id': '33', 'IdFuncionario': '17', 'Nome': 'Daniel Costa', 'DataNascimento': '2013-10-18', 'Parentesco': 'Filho'}
{'Id': '34', 'IdFuncionario': '17', 'Nome': 'Clara Costa', 'DataNascimento': '2016-04-11', 'Parentesco': 'Filha'}
{'Id': '35', 'IdFuncionario': '18', 'Nome': 'André Santos', 'DataNascimento': '2010-07-07', 'Parentesco': 'Filho'}
{'Id': '36', 'IdFuncionario': '18', 'Nome': 'Isabel Santos', 'DataNascimento': '2012-09-29', 'Parentesco': 'Filha'}
{'Id': '37', 'IdFuncionario': '19', 'Nome': 'Bruno Lima', 'DataNascimento': '2013-05-04', 'Parentesco': 'Filho'}
{'Id': '38', 'IdFuncionario': '19', 'Nome': 'Sofia Lima', 'DataNascimento': '2015-08-15', 'Parentesco': 'Filha'}
```

```
Histórico de Salários:
{'IdFuncionario': '1', 'MesAno': '2024-09-01', 'SalarioRecebido': '1000.00'}
{'IdFuncionario': '1', 'MesAno': '2024-08-01', 'SalarioRecebido': '1000.00'}
{'IdFuncionario': '1', 'MesAno': '2024-07-01', 'SalarioRecebido': '1000.00'}
{'IdFuncionario': '1', 'MesAno': '2024-06-01', 'SalarioRecebido': '1000.00'}
{'IdFuncionario': '1', 'MesAno': '2024-05-01', 'SalarioRecebido': '1000.00'}
{'IdFuncionario': '1', 'MesAno': '2024-04-01', 'SalarioRecebido': '1000.00'}
{'IdFuncionario': '10', 'MesAno': '2024-09-01', 'SalarioRecebido': '1000.00'}
{'IdFuncionario': '10', 'MesAno': '2024-08-01', 'SalarioRecebido': '1000.00'}
{'IdFuncionario': '10', 'MesAno': '2024-07-01', 'SalarioRecebido': '1000.00'}
{'IdFuncionario': '10', 'MesAno': '2024-06-01', 'SalarioRecebido': '1000.00'}
{'IdFuncionario': '10', 'MesAno': '2024-05-01', 'SalarioRecebido': '1000.00'}
{'IdFuncionario': '10', 'MesAno': '2024-04-01', 'SalarioRecebido': '1000.00'}
{'IdFuncionario': '11', 'MesAno': '2024-09-01', 'SalarioRecebido': '3000.00'}
{'IdFuncionario': '11', 'MesAno': '2024-08-01', 'SalarioRecebido': '3000.00'}
{'IdFuncionario': '11', 'MesAno': '2024-07-01', 'SalarioRecebido': '3000.00'}
{'IdFuncionario': '11', 'MesAno': '2024-06-01', 'SalarioRecebido': '3000.00'}
{'IdFuncionario': '11', 'MesAno': '2024-05-01', 'SalarioRecebido': '3000.00'}
{'IdFuncionario': '11', 'MesAno': '2024-04-01', 'SalarioRecebido': '3000.00'}
{'IdFuncionario': '12', 'MesAno': '2024-09-01', 'SalarioRecebido': '5000.00'}
{'IdFuncionario': '12', 'MesAno': '2024-08-01', 'SalarioRecebido': '5000.00'}
{'IdFuncionario': '12', 'MesAno': '2024-07-01', 'SalarioRecebido': '5000.00'}
{'IdFuncionario': '12', 'MesAno': '2024-06-01', 'SalarioRecebido': '5000.00'}
{'IdFuncionario': '12', 'MesAno': '2024-05-01', 'SalarioRecebido': '5000.00'}
{'IdFuncionario': '12', 'MesAno': '2024-04-01', 'SalarioRecebido': '5000.00'}
{'IdFuncionario': '13', 'MesAno': '2024-09-01', 'SalarioRecebido': '1000.00'}
{'IdFuncionario': '13', 'MesAno': '2024-08-01', 'SalarioRecebido': '1000.00'}
{'IdFuncionario': '13', 'MesAno': '2024-07-01', 'SalarioRecebido': '1000.00'}
{'IdFuncionario': '13', 'MesAno': '2024-06-01', 'SalarioRecebido': '1000.00'}
{'IdFuncionario': '13', 'MesAno': '2024-05-01', 'SalarioRecebido': '1000.00'}
{'IdFuncionario': '13', 'MesAno': '2024-04-01', 'SalarioRecebido': '1000.00'}
{'IdFuncionario': '14', 'MesAno': '2024-09-01', 'SalarioRecebido': '2000.00'}
```

```
Departamentos:
{'Id': '1', 'IdFuncionario': '22', 'Nome': 'TI', 'Andar': '1'}
{'Id': '2', 'IdFuncionario': '22', 'Nome': 'RH', 'Andar': '2'}
{'Id': '3', 'IdFuncionario': '24', 'Nome': 'Financeiro', 'Andar': '3'}
{'Id': '4', 'IdFuncionario': '25', 'Nome': 'Marketing', 'Andar': '4'}
{'Id': '5', 'IdFuncionario': '26', 'Nome': 'Vendas', 'Andar': '5'}
{'Id': '6', 'IdFuncionario': 'NULL', 'Nome': 'Limpeza', 'Andar': '6'}
```

```
Cargos:
{'Id': '1', 'IdNivel': '2', 'Descricao': 'Analista de Sistemas', 'SalarioBase': '2000.0'}
{'Id': '10', 'IdNivel': '3', 'Descricao': 'Contador', 'SalarioBase': '5000.0'}
{'Id': '11', 'IdNivel': '4', 'Descricao': 'Contador', 'SalarioBase': '7000.0'}
{'Id': '12', 'IdNivel': '1', 'Descricao': 'Analista de Marketing', 'SalarioBase': '1000.0'}
{'Id': '13', 'IdNivel': '2', 'Descricao': 'Analista de Marketing', 'SalarioBase': '2000.0'}
{'Id': '14', 'IdNivel': '3', 'Descricao': 'Analista de Marketing', 'SalarioBase': '3000.0'}
{'Id': '15', 'IdNivel': '4', 'Descricao': 'Analista de Marketing', 'SalarioBase': '4000.0'}
{'Id': '16', 'IdNivel': '1', 'Descricao': 'Vendedor', 'SalarioBase': '1000.0'}
{'Id': '17', 'IdNivel': '2', 'Descricao': 'Vendedor', 'SalarioBase': '2000.0'}
{'Id': '18', 'IdNivel': '3', 'Descricao': 'Vendedor', 'SalarioBase': '3000.0'}
{'Id': '19', 'IdNivel': '4', 'Descricao': 'Vendedor', 'SalarioBase': '5000.0'}
{'Id': '2', 'IdNivel': '3', 'Descricao': 'Analista de Sistemas', 'SalarioBase': '4000.0'}
{'Id': '20', 'IdNivel': '5', 'Descricao': 'Gerente', 'SalarioBase': '15000.0'}
{'Id': '21', 'IdNivel': '6', 'Descricao': 'Diretor', 'SalarioBase': '25000.0'}
{'Id': '3', 'IdNivel': '4', 'Descricao': 'Analista de Sistemas', 'SalarioBase': '8000.0'}
{'Id': '4', 'IdNivel': '1', 'Descricao': 'Analista de RH', 'SalarioBase': '1000.0'}
{'Id': '5', 'IdNivel': '2', 'Descricao': 'Analista de RH', 'SalarioBase': '3000.0'}
{'Id': '6', 'IdNivel': '3', 'Descricao': 'Analista de RH', 'SalarioBase': '4500.0'}
{'Id': '7', 'IdNivel': '4', 'Descricao': 'Analista de RH', 'SalarioBase': '6500.0'}
{'Id': '8', 'IdNivel': '1', 'Descricao': 'Contador', 'SalarioBase': '1000.0'}
{'Id': '9', 'IdNivel': '2', 'Descricao': 'Contador', 'SalarioBase': '3000.0'}
```

```
1 - S/SQL - Exibir separadamente as tabelas de Funcionários, Cargos, Departamentos, Histórico de Salários e Dependentes, organizadas em ordem crescente.
Funcionários:
{'Id': '1', 'Nome': 'Lucas Ferreira', 'IdCargo': '1', 'IdDepartamento': '1', 'Salario': '1000.0'}
{'Id': '10', 'Nome': 'Gabriel Lima', 'IdCargo': '9', 'IdDepartamento': '3', 'Salario': '1000.0'}
{'Id': '11', 'Nome': 'Roberta Martins', 'IdCargo': '10', 'IdDepartamento': '3', 'Salario': '3000.0'}
{'Id': '12', 'Nome': 'Tatiane Rocha', 'IdCargo': '11', 'IdDepartamento': '3', 'Salario': '5000.0'}
{'Id': '13', 'Nome': 'Felipe Souza', 'IdCargo': '12', 'IdDepartamento': '3', 'Salario': '7000.0'}
{'Id': '14', 'Nome': 'Laura Nunes', 'IdCargo': '13', 'IdDepartamento': '4', 'Salario': '1000.0'}
{'Id': '15', 'Nome': 'Luana Almeida', 'IdCargo': '14', 'IdDepartamento': '4', 'Salario': '2000.0'}
{'Id': '16', 'Nome': 'Renato Silva', 'IdCargo': '15', 'IdDepartamento': '4', 'Salario': '3000.0'}
{'Id': '17', 'Nome': 'Marcela Costa', 'IdCargo': '16', 'IdDepartamento': '4', 'Salario': '4000.0'}
{'Id': '18', 'Nome': 'Eduardo Santos', 'IdCargo': '17', 'IdDepartamento': '5', 'Salario': '1000.0'}
{'Id': '19', 'Nome': 'Priscila Lima', 'IdCargo': '18', 'IdDepartamento': '5', 'Salario': '2000.0'}
{'Id': '2', 'Nome': 'Lucas Almeida', 'IdCargo': '1', 'IdDepartamento': '1', 'Salario': '1000.0'}
{'Id': '20', 'Nome': 'Leonardo Rocha', 'IdCargo': '19', 'IdDepartamento': '5', 'Salario': '3000.0'}
{'Id': '21', 'Nome': 'Paula Pereira', 'IdCargo': '20', 'IdDepartamento': '5', 'Salario': '5000.0'}
{'Id': '22', 'Nome': 'João Fernandes', 'IdCargo': '21', 'IdDepartamento': '1', 'Salario': '15000.0'}
{'Id': '23', 'Nome': 'Cristina Andrade', 'IdCargo': '22', 'IdDepartamento': '2', 'Salario': '25000.0'}
{'Id': '24', 'Nome': 'Juliano Lima', 'IdCargo': '21', 'IdDepartamento': '3', 'Salario': '15000.0'}
{'Id': '25', 'Nome': 'Aline Costa', 'IdCargo': '21', 'IdDepartamento': '4', 'Salario': '15000.0'}
{'Id': '26', 'Nome': 'Sandra Oliveira', 'IdCargo': '21', 'IdDepartamento': '5', 'Salario': '15000.0'}
{'Id': '27', 'Nome': 'Janaina Pereira', 'IdCargo': '', 'IdDepartamento': '6', 'Salario': '1500.0'}
{'Id': '3', 'Nome': 'Fernanda Oliveira', 'IdCargo': '2', 'IdDepartamento': '1', 'Salario': '2000.0'}
{'Id': '4', 'Nome': 'Ricardo Costa', 'IdCargo': '3', 'IdDepartamento': '1', 'Salario': '4000.0'}
{'Id': '5', 'Nome': 'Júlia Silva', 'IdCargo': '4', 'IdDepartamento': '1', 'Salario': '8000.0'}
{'Id': '6', 'Nome': 'Pedro Santos', 'IdCargo': '5', 'IdDepartamento': '2', 'Salario': '1000.0'}
{'Id': '7', 'Nome': 'Mariana Almeida', 'IdCargo': '6', 'IdDepartamento': '2', 'Salario': '3000.0'}
{'Id': '8', 'Nome': 'Carlos Pereira', 'IdCargo': '7', 'IdDepartamento': '2', 'Salario': '4500.0'}
{'Id': '9', 'Nome': 'Ana Costa', 'IdCargo': '8', 'IdDepartamento': '2', 'Salario': '6500.0'}
```

2. Listar os funcionários, com seus cargos, departamentos e os

respectivos dependentes.

```python
print('-'*50)
print('2 - S/SQL - Exibir os funcionários juntamente com seus cargos, departamentos e respectivos dependentes.')
print('-'*50)
resultado = []
for func in funcionarios:
    cargo = next((c for c in cargos if c['Id'] == func['IdCargo']), None)
    departamento = next((d for d in departamentos if d['Id'] == func['IdDepartamento']), None)
    dependente = [d for d in dependentes if d['IdFuncionario'] == func['Id']]

    resultado.append({
        'Funcionario': func['Nome'],
        'Cargo': cargo['Descricao'] if cargo else 'Desconhecido',
        'Departamento': departamento['Nome'] if departamento else 'Desconhecido',
        'Dependentes': [d['Nome'] for d in dependente] if dependente else ['Nenhum']
    })
print("Funcionários com Cargos, Departamentos e Dependentes:")
for res in resultado:
    print(f"Funcionário: {res['Funcionario']} | Cargo: {res['Cargo']} | Departamento: {res['Departamento']} | Dependentes: {', '.
    join(res['Dependentes'])}")
```

Explicação:

1. **Exibição do Cabeçalho:**
   - O código inicia com a impressão de uma linha divisória e uma descrição que informa que os dados dos funcionários e suas informações associadas estão sendo listados.
2. **Inicialização da Lista de Resultados:**
   - Uma lista vazia chamada `resultado` é criada para armazenar as informações dos funcionários.
3. **Iteração sobre Funcionários:**
   - O código percorre cada funcionário na lista de funcionários.
   - Para cada funcionário, utiliza-se uma expressão geradora com a função `next()` para encontrar o cargo e o departamento correspondentes, verificando suas IDs.
   - Uma lista de dependentes associados ao funcionário é criada, filtrando a lista de dependentes pela chave `IdFuncionario`.
4. **Construção dos Resultados:**
   - As informações (nome do funcionário, descrição do cargo, nome do departamento e nomes dos dependentes) são organizadas em um dicionário e adicionadas à lista `resultado`.
   - Se algum dado não for encontrado, valores padrão como "Desconhecido" ou "Nenhum" são atribuídos.
5. **Impressão dos Resultados:**
   - Por fim, o código imprime as informações de cada funcionário de forma clara, exibindo nome, cargo, departamento e dependentes.

```
2 - S/SQL - Exibir os funcionários juntamente com seus cargos, departamentos e respectivos dependentes.
--------------------------------------------------
Funcionários com Cargos, Departamentos e Dependentes:
Funcionário: Lucas Ferreira | Cargo: Analista de Sistemas | Departamento: TI | Dependentes: Maria Ferreira, Ana Ferreira
Funcionário: Lucas Almeida | Cargo: Analista de Sistemas | Departamento: TI | Dependentes: Beatriz Almeida, Carlos Almeida
Funcionário: Fernanda Oliveira | Cargo: Analista de Sistemas | Departamento: TI | Dependentes: Sofia Oliveira, Ana Oliveira
Funcionário: Ricardo Costa | Cargo: Analista de Sistemas | Departamento: TI | Dependentes: Pedro Costa, Julia Costa
Funcionário: Júlia Silva | Cargo: Analista de RH | Departamento: TI | Dependentes: Alice Silva, Eduardo Silva
Funcionário: Pedro Santos | Cargo: Analista de RH | Departamento: RH | Dependentes: Mariana Santos, Felipe Santos
Funcionário: Mariana Almeida | Cargo: Analista de RH | Departamento: RH | Dependentes: Ana Almeida, Bruno Almeida
Funcionário: Carlos Pereira | Cargo: Analista de RH | Departamento: RH | Dependentes: Gabriel Pereira, Isabela Pereira
Funcionário: Ana Costa | Cargo: Contador | Departamento: RH | Dependentes: João Costa, Laura Costa
Funcionário: Gabriel Lima | Cargo: Contador | Departamento: Financeiro | Dependentes: Daniel Lima, Rafaela Lima
Funcionário: Roberta Martins | Cargo: Contador | Departamento: Financeiro | Dependentes: Henrique Martins, Luana Martins
Funcionário: Tatiane Rocha | Cargo: Contador | Departamento: Financeiro | Dependentes: Samuel Rocha, Carla Rocha
Funcionário: Felipe Souza | Cargo: Analista de Marketing | Departamento: Financeiro | Dependentes: Eduardo Souza, Fernanda Souza
Funcionário: Laura Nunes | Cargo: Analista de Marketing | Departamento: Marketing | Dependentes: Paulo Nunes, Mariana Nunes
Funcionário: Luana Almeida | Cargo: Analista de Marketing | Departamento: Marketing | Dependentes: Lucas Almeida, Juliana Almeida
Funcionário: Renato Silva | Cargo: Analista de Marketing | Departamento: Marketing | Dependentes: Gustavo Silva, Alice Silva
Funcionário: Marcela Costa | Cargo: Vendedor | Departamento: Marketing | Dependentes: Daniel Costa, Clara Costa
Funcionário: Eduardo Santos | Cargo: Vendedor | Departamento: Vendas | Dependentes: André Santos, Isabel Santos
Funcionário: Priscila Lima | Cargo: Vendedor | Departamento: Vendas | Dependentes: Bruno Lima, Sofia Lima
Funcionário: Leonardo Rocha | Cargo: Vendedor | Departamento: Vendas | Dependentes: Ricardo Rocha, Laura Rocha
Funcionário: Paula Pereira | Cargo: Gerente | Departamento: Vendas | Dependentes: Carlos Pereira, Isabela Pereira
Funcionário: João Fernandes | Cargo: Diretor | Departamento: TI | Dependentes: Miguel Fernandes, Amanda Fernandes
Funcionário: Cristina Andrade | Cargo: Desconhecido | Departamento: RH | Dependentes: Lucas Andrade, Fernanda Andrade
Funcionário: Juliano Lima | Cargo: Diretor | Departamento: Financeiro | Dependentes: Pedro Lima, Letícia Lima
Funcionário: Aline Costa | Cargo: Diretor | Departamento: Marketing | Dependentes: Enzo Costa, Sabrina Costa
Funcionário: Sandra Oliveira | Cargo: Diretor | Departamento: Vendas | Dependentes: Tiago Oliveira, Mariana Oliveira
Funcionário: Janaina Pereira | Cargo: Desconhecido | Departamento: Limpeza | Dependentes: João Pereira, Maria Pereira
```

3. Listar os funcionários que tiveram aumento salarial nos últimos 3 meses

```python
print('-'*50)
print('3 - S/SQL - Exibir os funcionários que receberam aumento salarial nos últimos 3 meses.')
print('-'*50)
tres_meses_antes = datetime.now() - timedelta(days=90)
aumentos_recent = [
    hist for hist in historico_salarios
    if datetime.strptime(hist['MesAno'], '%Y-%m-%d') >= tres_meses_antes
]
funcionarios_com_aumento = {}
for hist in aumentos_recent:
    func_id = hist['IdFuncionario']
    salario = float(hist['SalarioRecebido'])
    if func_id not in funcionarios_com_aumento:
        funcionarios_com_aumento[func_id] = salario
    else:
        funcionarios_com_aumento[func_id] = max(funcionarios_com_aumento[func_id], salario)
print("Funcionários que tiveram aumento salarial nos últimos 3 meses:")
for func_id, salario in funcionarios_com_aumento.items():
    funcionario = next((f for f in funcionarios if f['Id'] == func_id), None)
    if funcionario:
        print(f"Funcionário ID: {func_id} | Nome: {funcionario['Nome']} | Salário: {salario:.2f}")
```

Explicação:

1. **Exibição do Cabeçalho:**
    - O código inicia com a impressão de uma linha de separação e uma descrição indicando que está listando os funcionários que receberam aumento salarial recentemente.
2. **Cálculo da Data Limite:**
    - A variável `tres_meses_antes` é calculada subtraindo 90 dias da data atual, utilizando `datetime.now()` e `timedelta`.
3. **Filtragem de Aumentos Recentes:**
    - Uma lista chamada `aumentos_recent` é criada, contendo apenas os registros do histórico de salários (`historico_salarios`) que ocorreram nos últimos três meses. Isso é feito através de uma compreensão de lista que verifica se a data do salário recebido é maior ou igual à data limite.
4. **Compilação dos Funcionários com Aumento:**
    - Um dicionário denominado `funcionarios_com_aumento` é inicializado para armazenar os funcionários que tiveram aumentos. O código itera sobre a lista de aumentos recentes e, para cada registro, verifica se o ID do funcionário já está presente no dicionário.
    - Se não estiver, o salário é adicionado. Caso contrário, o código atualiza o salário no dicionário para o maior valor entre o salário já registrado e o novo.
5. **Impressão dos Resultados:**
    - O código imprime a lista dos funcionários que tiveram aumento salarial, formatando a saída para mostrar o ID do funcionário, seu nome (extraído da lista de funcionários) e o salário mais recente.

```
3 - S/SQL - Exibir os funcionários que receberam aumento salarial nos últimos 3 meses.
--------------------------------------------------
Funcionários que tiveram aumento salarial nos últimos 3 meses:
Funcionário ID: 1 | Nome: Lucas Ferreira | Salário: 1000.00
Funcionário ID: 2 | Nome: Lucas Almeida | Salário: 1000.00
Funcionário ID: 3 | Nome: Fernanda Oliveira | Salário: 2000.00
Funcionário ID: 4 | Nome: Ricardo Costa | Salário: 4000.00
Funcionário ID: 5 | Nome: Júlia Silva | Salário: 8000.00
Funcionário ID: 6 | Nome: Pedro Santos | Salário: 1000.00
Funcionário ID: 7 | Nome: Mariana Almeida | Salário: 3000.00
Funcionário ID: 8 | Nome: Carlos Pereira | Salário: 4500.00
Funcionário ID: 9 | Nome: Ana Costa | Salário: 6500.00
Funcionário ID: 10 | Nome: Gabriel Lima | Salário: 1000.00
Funcionário ID: 11 | Nome: Roberta Martins | Salário: 3000.00
Funcionário ID: 12 | Nome: Tatiane Rocha | Salário: 5000.00
Funcionário ID: 13 | Nome: Felipe Souza | Salário: 1000.00
Funcionário ID: 14 | Nome: Laura Nunes | Salário: 2000.00
Funcionário ID: 15 | Nome: Luana Almeida | Salário: 3000.00
Funcionário ID: 16 | Nome: Renato Silva | Salário: 4000.00
Funcionário ID: 17 | Nome: Marcela Costa | Salário: 1000.00
Funcionário ID: 18 | Nome: Eduardo Santos | Salário: 2000.00
Funcionário ID: 19 | Nome: Priscila Lima | Salário: 3000.00
Funcionário ID: 20 | Nome: Leonardo Rocha | Salário: 5000.00
Funcionário ID: 21 | Nome: Paula Pereira | Salário: 15000.00
Funcionário ID: 22 | Nome: João Fernandes | Salário: 25000.00
```

4. Listar a média de idade dos filhos dos funcionários por departamento.

```python
print('-'*50)
print('4 - S/SQL - Exibir a média de idade dos filhos dos funcionários, organizada por departamento.')
print('-'*50)
for dep in dependentes:
    dep['DataNascimento'] = datetime.strptime(dep['DataNascimento'], '%Y-%m-%d')
for dep in dependentes:
    dep['Idade'] = (datetime.now() - dep['DataNascimento']).days // 365
media_idade_por_depa = {}
for dep in dependentes:
    funcionario = next((f for f in funcionarios if f['Id'] == dep['IdFuncionario']), None)
    if funcionario:
        departamento = next((d for d in departamentos if d['Id'] == funcionario['IdDepartamento']), None)
        if departamento:
            dep_nome = departamento['Nome']
            if dep_nome not in media_idade_por_depa:
                media_idade_por_depa[dep_nome] = []
            media_idade_por_depa[dep_nome].append(dep['Idade'])
print("Média de idade dos filhos dos funcionários por departamento:")
for dep_nome, idades in media_idade_por_depa.items():
    media_idade = sum(idades) / len(idades) if idades else 0
    print(f"Departamento: {dep_nome} | Média de Idade: {media_idade:.2f}")
```

Explicação:

1. **Exibição do Cabeçalho:**
   - O código começa com a impressão de uma linha de separação e uma descrição que informa que está listando a média de idade dos filhos dos funcionários por departamento.
2. **Conversão das Datas de Nascimento:**
   - O primeiro loop percorre a lista de dependentes, convertendo a string da data de nascimento de cada um em um objeto datetime, facilitando o cálculo da idade.
3. **Cálculo das Idades:**
   - Em um segundo loop, a idade de cada dependente é calculada subtraindo a data de nascimento da data atual e convertendo o resultado em anos, utilizando `days // 365`.
4. **Compilação das Idades por Departamento:**
   - Um dicionário chamado `media_idade_por_depa` é criado para armazenar as idades dos filhos agrupadas por departamento.
   - O código itera novamente sobre a lista de dependentes e, para cada um, identifica o funcionário correspondente e seu departamento.
   - O nome do departamento é utilizado como chave no dicionário, e as idades dos filhos são adicionadas a uma lista correspondente.
5. **Cálculo e Impressão das Médias de Idade:**
   - Por fim, o código calcula a média de idade dos filhos para cada departamento, utilizando a soma das idades dividida pelo número de dependentes.
   - Os resultados são impressos, mostrando o nome do departamento e a média de idade formatada com duas casas decimais.

```
4 - S/SQL - Exibir a média de idade dos filhos dos funcionários, organizada por departamento.
------------------------------------------------
Média de idade dos filhos dos funcionários por departamento:
Departamento: TI | Média de Idade: 11.00
Departamento: RH | Média de Idade: 10.90
Departamento: Financeiro | Média de Idade: 11.30
Departamento: Marketing | Média de Idade: 9.60
Departamento: Vendas | Média de Idade: 11.10
```

5. Listar qual estagiário possui filho.

```python
print('-'*50)
print('5 - S/SQL - Exibir quais estagiários têm filhos.')
print('-'*50)
estagiario_ids = [cargo['Id'] for cargo in cargos if cargo['IdNivel'] == '1']
estagiarios = []
for func in funcionarios:
    if func['IdCargo'] in estagiario_ids:
        estagiarios.append(func)
estagiarios_com_filho = []
for estagiario in estagiarios:
    filhos = [dep for dep in dependentes if dep['IdFuncionario'] == estagiario['Id']]
    if filhos:
        estagiarios_com_filho.append((estagiario['Nome'], [f['Nome'] for f in filhos]))
print("Estagiários que possuem filhos:")
for est_nome, filhos in estagiarios_com_filho:
    print(f"Estagiário: {est_nome} | Filhos: {', '.join(filhos)}")
```

Explicação:

1. **Obtenção dos IDs de Estagiários:**
   - O código começa buscando todos os IDs de cargos que correspondem ao nível 1 (estagiário) na tabela de cargos.
2. **Filtragem de Funcionários:**
   - Em seguida, percorre a lista de funcionários, verificando se o `IdCargo` de cada um corresponde a um dos IDs de estagiários. Os funcionários que atendem a essa condição são armazenados em uma lista.
3. **Verificação de Filhos:**
   - O código, então, itera sobre a lista de estagiários e, para cada um, verifica na tabela de dependentes se existem filhos associados a esse estagiário, utilizando o `IdFuncionario`.
4. **Exibição dos Resultados:**
   - Por fim, imprime os nomes dos estagiários que possuem filhos, juntamente com os nomes dos filhos, em um formato legível.

```
5 - S/SQL - Exibir quais estagiários têm filhos.
--------------------------------------------------
Estagiários que possuem filhos:
Estagiário: Júlia Silva | Filhos: Alice Silva, Eduardo Silva
Estagiário: Ana Costa | Filhos: João Costa, Laura Costa
Estagiário: Felipe Souza | Filhos: Eduardo Souza, Fernanda Souza
Estagiário: Marcela Costa | Filhos: Daniel Costa, Clara Costa
```

6. Listar o funcionário que teve o salário médio mais alto.

```
print('-'*50)
print('6 - C/SQL - Exibir o funcionário com o salário médio mais alto.')
print('-'*50)
cursor.execute('''
    SELECT
        FUNC.Id AS FuncionarioId,
        FUNC.Nome AS FuncionarioNome,
        AVG(HISA.SalarioRecebido) AS SalarioMedio
    FROM TB_FUNCIONARIO FUNC
    JOIN TB_HISTORICO_SALARIO HISA ON FUNC.Id = HISA.IdFuncionario
    GROUP BY FUNC.Id, FUNC.Nome
    ORDER BY SalarioMedio DESC
    LIMIT 1;
''')
resultado = cursor.fetchall()
print("Funcionário com o salário médio mais alto:")
for linha in resultado:
    print(f"Funcionário ID: {linha[0]} | Nome: {linha[1]} | Salário Médio: {linha[2]:.2f}")
```

Explicação:

1. **Execução da Consulta SQL:**
    - O código utiliza uma consulta SQL para selecionar o ID e o nome dos funcionários, além da média dos salários recebidos. A consulta realiza um JOIN entre as tabelas `TB_FUNCIONARIO` e `TB_HISTORICO_SALARIO`, correlacionando os dados dos funcionários com seus históricos salariais.
2. **Cálculo da Média Salarial:**
    - A média salarial é calculada usando a função `AVG`, agrupando os resultados pelo ID e nome do funcionário para garantir que a média seja calculada corretamente para cada um.
3. **Ordenação e Limitação:**
    - Os resultados são ordenados em ordem decrescente com base no salário médio, e o comando `LIMIT 1` assegura que apenas o funcionário com o maior salário médio seja retornado.
4. **Exibição dos Resultados:**
    - Os resultados da consulta são armazenados em uma variável, que é então percorrida para imprimir o ID do funcionário, seu nome e o salário médio de forma legível.

```
6 - C/SQL - Exibir o funcionário com o salário médio mais alto.
------------------------------------------------
Funcionário com o salário médio mais alto:
Funcionário ID: 22 | Nome: João Fernandes | Salário Médio: 23333.33
```

7. Listar o analista que é pai de 2 (duas) meninas.

```python
print('-'*50)
print('7 - C/SQL - Exibir o analista que é pai de duas meninas.')
print('-'*50)
cursor.execute('''
    SELECT
        FUNC.Id AS FuncionarioId,
        FUNC.Nome AS FuncionarioNome
    FROM TB_FUNCIONARIO FUNC
    JOIN TB_CARGO CARG ON FUNC.IdCargo = CARG.Id
    JOIN TB_DEPENDENTE DEPE ON FUNC.Id = DEPE.IdFuncionario
    WHERE CARG.Descricao LIKE 'Analista%'
    AND DEPE.Parentesco = 'Filha'
    GROUP BY FUNC.Id, FUNC.Nome
    HAVING COUNT(DEPE.Id) = 2;
''')
resultado = cursor.fetchall()
print("Analista que é pai de 2 filhas:")
for linha in resultado:
    print(f"Funcionário ID: {linha[0]} | Nome: {linha[1]}")
```

Explicação:

1. **Execução da Consulta SQL:**
    - O código executa uma consulta SQL que combina dados das tabelas

`TB_FUNCIONARIO`, `TB_CARGO` e `TB_DEPENDENTE`. A junção (JOIN) é utilizada para conectar as informações dos funcionários com seus respectivos cargos e dependentes.

2. **Filtragem por Cargo e Parentesco:**
    - A cláusula `WHERE` filtra os funcionários que ocupam cargos que começam com "Analista" e cujos dependentes possuem o parentesco de "Filha".
3. **Agrupamento e Contagem:**
    - Os resultados são agrupados pelo ID e nome do funcionário. A cláusula `HAVING` é utilizada para garantir que apenas aqueles que têm exatamente duas filhas sejam incluídos nos resultados.
4. **Exibição dos Resultados:**
    - Os dados retornados da consulta são armazenados em uma variável, e um loop percorre os resultados, imprimindo o ID do funcionário e seu nome.

```
7 - C/SQL - Exibir o analista que é pai de duas meninas.
--------------------------------------------------
Analista que é pai de 2 filhas:
Funcionário ID: 1 | Nome: Lucas Ferreira
Funcionário ID: 3 | Nome: Fernanda Oliveira
Funcionário ID: 4 | Nome: Ricardo Costa
```

8. Listar o analista que tem o salário mais alto, e que ganhe entre 5000 e 9000.

```python
print('-'*50)
print('8 - C/SQL - Exibir o analista com o salário mais alto que recebe entre 5.000 e 9.000.')
print('-'*50)
cursor.execute('''
    SELECT
        FUNC.Id AS FuncionarioId,
        FUNC.Nome AS FuncionarioNome,
        CARG.SalarioBase AS Salario
    FROM TB_FUNCIONARIO FUNC
    JOIN TB_CARGO CARG ON FUNC.IdCargo = CARG.Id
    WHERE CARG.Descricao LIKE 'Analista%'
    AND CARG.SalarioBase BETWEEN 5000 AND 9000
    ORDER BY CARG.SalarioBase DESC
    LIMIT 1;
''')
resultado = cursor.fetchall()
print("Analista com o salário mais alto (entre 5000 e 9000):")
for linha in resultado:
    print(f"Funcionário ID: {linha[0]} | Nome: {linha[1]} | Salário: {linha[2]:.2f}")
```

Explicação:

1. **Execução da Consulta SQL:**
   - O código realiza uma consulta SQL que une dados das tabelas `TB_FUNCIONARIO` e `TB_CARGO` através de uma junção (JOIN), associando os funcionários aos seus respectivos cargos.
2. **Filtragem por Cargo e Salário:**
   - A cláusula `WHERE` filtra os funcionários que ocupam cargos que começam com "Analista" e cujos salários estão na faixa de 5.000 a 9.000. Isso assegura que somente os analistas dentro da faixa salarial especificada sejam considerados.
3. **Ordenação e Limitação:**
   - A cláusula `ORDER BY` é empregada para classificar os resultados pelo salário em ordem decrescente, e o comando `LIMIT 1` garante que apenas o analista com o maior salário dentro da faixa especificada seja retornado.
4. **Exibição dos Resultados:**
   - Os dados resultantes da consulta são armazenados em uma variável, e um loop itera sobre os resultados, imprimindo o ID do funcionário, seu nome e o salário formatado.

```
8 - C/SQL - Exibir o analista com o salário mais alto que recebe entre 5.000 e 9.000.
------------------------------------------------
Analista com o salário mais alto (entre 5000 e 9000):
Funcionário ID: 4 | Nome: Ricardo Costa | Salário: 8000.00
```

9. Listar qual departamento possui o maior número de dependentes.

```python
print('-'*50)
print('9 - C/SQL - Exibir qual departamento tem o maior número de dependentes.')
print('-'*50)
cursor.execute('''
    SELECT
        DEPA.Nome AS DepartamentoNome,
        COUNT(DEPE.Id) AS NumeroDependentes
    FROM TB_DEPENDENTE DEPE
    JOIN TB_FUNCIONARIO FUNC ON DEPE.IdFuncionario = FUNC.Id
    JOIN TB_DEPARTAMENTO DEPA ON FUNC.IdDepartamento = DEPA.Id
    GROUP BY DEPA.Id, DEPA.Nome
    ORDER BY NumeroDependentes DESC
    LIMIT 1;
''')
resultado = cursor.fetchall()
print("Departamento com o maior número de dependentes:")
for linha in resultado:
    print(f"Departamento: {linha[0]} | Número de Dependentes: {linha[1]}")
```

Explicação:

1. **Execução da Consulta SQL:**
   - O código realiza uma consulta SQL que combina dados das tabelas `TB_DEPENDENTE`, `TB_FUNCIONARIO` e `TB_DEPARTAMENTO por meio de junções (JOIN)`.
2. **Contagem de Dependentes:**
   - A consulta faz uso da função de agregação `COUNT` para contabilizar quantos dependentes (DEPE.Id) estão relacionados a cada departamento. Essa contagem é feita agrupando os resultados pelo ID e pelo nome do departamento (`GROUP BY`).
3. **Ordenação e Limitação:**

- Os resultados são organizados em ordem decrescente de dependentes (`ORDER BY NumeroDependentes DESC`), e o comando `LIMIT 1` assegura que apenas o departamento com a maior quantidade de dependentes seja retornado.

4. **Exibição dos Resultados:**
   - Os dados resultantes da consulta são armazenados em uma variável, e um loop percorre os resultados, imprimindo o nome do departamento e o total de dependentes.

```
9 - C/SQL - Exibir qual departamento tem o maior número de dependentes.
--------------------------------------------------
Departamento com o maior número de dependentes:
Departamento: TI | Número de Dependentes: 12
```

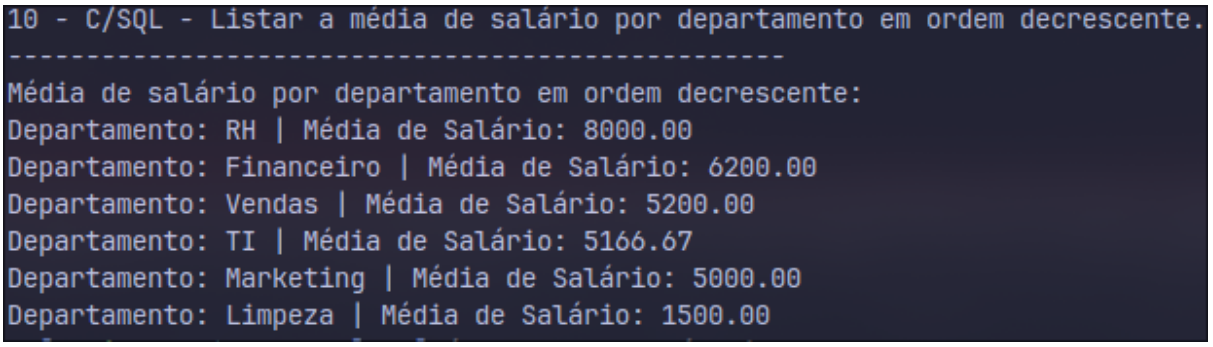
10. Listar a média de salário por departamento em ordem decrescente.

```python
print('-'*50)
print('10 - C/SQL - Listar a média de salário por departamento em ordem decrescente.')
print('-'*50)
cursor.execute('''
    SELECT
        DEPA.Nome AS DepartamentoNome,
        AVG(FUNC.Salario) AS MediaSalario
    FROM TB_FUNCIONARIO FUNC
    JOIN TB_DEPARTAMENTO DEPA ON FUNC.IdDepartamento = DEPA.Id
    GROUP BY DEPA.Id, DEPA.Nome
    ORDER BY MediaSalario DESC;
''')
resultado = cursor.fetchall()
print("Média de salário por departamento em ordem decrescente:")
for linha in resultado:
    print(f"Departamento: {linha[0]} | Média de Salário: {linha[1]:.2f}")
```

Explicação:

1. **Execução da Consulta SQL:** O código realiza uma consulta SQL que estabelece uma junção (JOIN) entre as tabelas TB_FUNCIONARIO e TB_DEPARTAMENTO, permitindo correlacionar os funcionários com seus respectivos departamentos.

2. **Cálculo da Média Salarial:** A função de agregação AVG é empregada para calcular a média dos salários dos funcionários (FUNC.Salario) em cada departamento. Os resultados são agrupados pelo ID e pelo nome do departamento (GROUP BY).
3. **Ordenação dos Resultados:** A cláusula ORDER BY MediaSalario DESC organiza os departamentos com base na média salarial em ordem decrescente, assegurando que aqueles com os salários mais altos sejam exibidos primeiro.
4. **Exibição dos Resultados:** Os dados obtidos da consulta são armazenados em uma variável, e um loop subsequente percorre os resultados, imprimindo o nome do departamento e a média salarial formatada com duas casas decimais.



Considerações Finais

Neste trabalho, diversas consultas SQL foram executadas para manipular e analisar dados em um banco de dados que abrange tabelas relacionadas a funcionários, cargos, departamentos, entre outros. As consultas foram elaboradas para atender a requisitos específicos e oferecer insights valiosos sobre a estrutura e o desempenho da organização, além de facilitar a integração com o Python.

- **Consultas de Listagem:**
    - As consultas permitiram listar informações sobre funcionários, cargos, departamentos, dependentes e histórico de salários em ordem alfabética, tornando a visualização e análise de dados básicos mais acessíveis.
- **Consultas de Filtro e Análise:**
    - Foi possível identificar departamentos específicos, como aqueles situados no quinto andar, e analisar os salários dos funcionários para detectar padrões e outliers.
    - As consultas também ajudaram a identificar o analista com o maior salário e a determinar qual departamento possui o maior número de estagiários, fornecendo informações valiosas sobre a distribuição e hierarquia dentro da empresa.

Em resumo, as consultas SQL realizadas mostraram-se eficazes na extração e análise de informações significativas do banco de dados. Elas ofereceram uma visão clara da estrutura organizacional e ajudaram a identificar áreas de interesse e oportunidades de melhoria. A documentação e as explicações fornecidas servem como uma base sólida para entender e utilizar os dados de forma eficiente.