TP3 - Projeto de Bloco: Fundamentos de Dados

Larissa Conti de Barros

Curso: Engenharia de Software

Período: 2024.1

Sumário

Arquivos CSV

cargos.csv

```
CargoID;Descricao;SalarioBase;Nivel;Beneficios
1;Diretor;9000;Diretor;Odontológico, Transporte, Médico, Alimentação
2;Gerente;8000;Gerente;Odontológico, Transporte, Alimentação
3;Analista;5000;Analista;Odontológico, Transporte, Alimentação
4;Técnico de Suporte;3000;Técnico;Transporte, Alimentação
5;Estagiário;2000;Estagiário;Transporte, Alimentação
```

Cargos.CSV

departamento.csv

```
DepartamentoID;NomeDepartamento;GerenteID;Andar;Funcoes
1;TI;2;7;Setor responsável pela arquitetura digital da empresa
2;Marketing;3;7;Setor responsável pela divulgação da empresa
3;Vendas;3;5;Setor responsável pelas transações da empresa
4;Suporte Técnico;2;4;Setor responsável pela manutenção e suporte à empresa
5;Financeiro;3;5;Setor de gestão financeira da empresa
```

Departamento.CSV

funcionarios.csv

```
FuncionarioID;Nome;CargoID;DepartamentoID;Salario;Genero
1;Leighton Marrow;1;1;9740;Feminino
2;Jamie Tolliver;2;1;8500;Feminino
3;Leonardo Gulliver;2;5;8200;Masculino
4;Marvin Guye;3;2;5200;Masculino
5;Jonas Fairwright;1;2;9750;Masculino
6;Mollie Goodrye;4;3;3540;Feminino
7;Stella Bordwell;3;5;5400;Feminino
8;Keith Dallas;4;4;3600;Masculino
9;Jessica Foxley;5;4;2400;Feminino
10;Garreth Wright;5;3;2300;Masculino
11;Gillian Sowsbury;null;4;1100;Feminino
12;Richard Folley;5;4;2000;Masculino
```

Funcionarios.CSV

dependentes.csv

dependentes.csv > data

```
DependenteID;FuncionarioID;Nome;Genero;Idade
1;1;Maddison DeLuca;Feminino;4
2;1;Jenna Ashley;Feminino;6
3;2;Kasey Valenzuela;Não-Binário;20
4;2;Everly Little;Feminino;14
5;3;Laurie Dillon;Feminino;8
6;3;Frances Chapman;Masculino;3
7;4;Kelly Noble;Feminino;16
8;4;Ashley Durham;Feminino;10
9;5;Louie Cooke;Masculino;1
10;5;Jade Cole;Feminino;5
11;5;Ronnie Palmer;Agênero;23
12;6;Georgie Irwin;Masculino;18
13;6;Marissa Norman;Feminino;9
14;6;Aston Dominguez;Agênero;17
15;6;Kaya Hawkins;Feminino;21
16;7;Dillan Abbott;Masculino;14
17;7;Nana Wallace;Feminino;9
18;8;Dana Vasquez;Feminino;12
19;8;Regan Farmer;Masculino;19
20;9;Malik Boyer;Não-Binário;23
21;9;Ashton Underwood;Masculino;15
22;10;Taylor Glass;Agênero;18
23;10;Ty Combs;Masculino;13
24;10;Kaden Bowman;Não-Binário;22
25;11;Charley Clayton;Masculino;2
26;11;Gabriel Green;Masculino;25
27;11;Maya Castillo;Feminino;8
28;12;Ollie Proctor;Não-Binário;16
29;12;Corey Riddle;Feminino;7
30;12;Callan Whitaker;Masculino;12
```

Dependentes.CSV

historico_salarios.csv

historico_salarios.csv > data

```
HistoricoSalarioID;FuncionarioID;Data;Salario
1;1;2023-01-05;9740
2;1;2023-02-05;9740
3;1;2023-03-05;9800
4;1;2023-04-05;9800
5;1;2023-05-05;9800
6;1;2023-06-05;9900
7;2;2023-01-05;8500
8;2;2023-02-05;8550
9;2;2023-03-05;8550
10;2;2023-04-05;8550
11;2;2023-05-05;8600
12;2;2023-06-05;8650
13;3;2023-01-05;8200
14;3;2023-02-05;8200
15;3;2023-03-05;8240
16;3;2023-04-05;8240
17;3;2023-05-05;8280
18;3;2023-06-05;8300
19;4;2023-01-05;5200
20;4;2023-02-05;5200
21;4;2023-03-05;5200
22;4;2023-04-05;5200
23;4;2023-05-05;5200
24;4;2023-06-05;5200
25;5;2023-01-05;9750
26;5;2023-02-05;9750
27;5;2023-03-05;9850
28;5;2023-04-05;9850
29;5;2023-05-05;9850
30;5;2023-06-05;9850
31;6;2023-01-05;3540
32;6;2023-02-05;3540
33;6;2023-03-05;3540
34;6;2023-04-05;3540
35;6;2023-05-05;3540
36;6;2023-06-05;3540
37;7;2023-01-05;5400
38;7;2023-02-05;5400
39;7;2023-03-05;5400
40;7;2023-04-05;5400
41;7;2023-05-05;5450
42;7;2023-05-05;5450
43;7;2023-06-05;5450
44;8;2023-01-05;3600
45;8;2023-02-05;3600
46;8;2023-03-05;3600
47;8;2023-04-05;3600
```

Historico_salarios.CSV (apenas uma parte)

Tabelas SQL criadas e populadas

```python
#Criando as tabelas ================================================================================
def criar_tabela(nome_tabela, query):
    try:
        # Conectar ao banco de dados
        conn = sqlite3.connect('DataDB.db')
        cursor = conn.cursor()

        # Verificar se a tabela já existe
        cursor.execute(f"SELECT name FROM sqlite_master WHERE type='table' AND name='{nome_tabela}';")
        if cursor.fetchone():
            print(f"\nTabela {nome_tabela} já existe.")
            print("\n====================================================================")
        else:
            # Criar a tabela
            cursor.execute(query)
            print(f"\nTabela {nome_tabela} criada com sucesso!")
            print("\n====================================================================")

        # Fechar a conexão com o banco de dados
        conn.close()
    except Exception as e:
        print(f"\nErro ao criar a tabela {nome_tabela}: {e}")

#Tabela Funcionarios
query = ('''
CREATE TABLE IF NOT EXISTS Funcionarios (
    FuncionarioID INT PRIMARY KEY,
    Nome VARCHAR(100),
    CargoID INT,
    DepartamentoID INT,
    Salario REAL,
    Genero VARCHAR(20),
    FOREIGN KEY (CargoID) REFERENCES Cargos(CargoID),
    FOREIGN KEY (DepartamentoID) REFERENCES Departamentos(DepartamentoID)
);
            ''')
criar_tabela('Funcionarios', query)

#Tabela Cargos
query = ('''
CREATE TABLE IF NOT EXISTS Cargos (
    CargoID INT PRIMARY KEY,
    Descricao VARCHAR(100),
    SalarioBase REAL,
    Nivel VARCHAR(20),
    Beneficios VARCHAR(100)
);
```

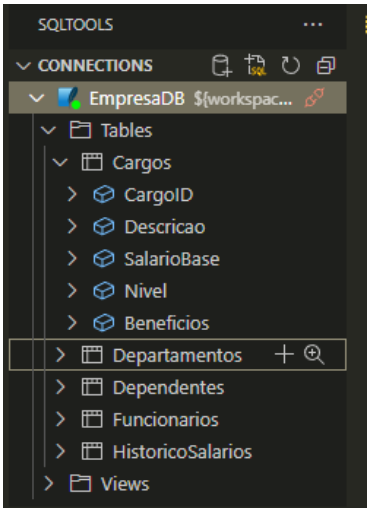
Criação das tabelas usando Python

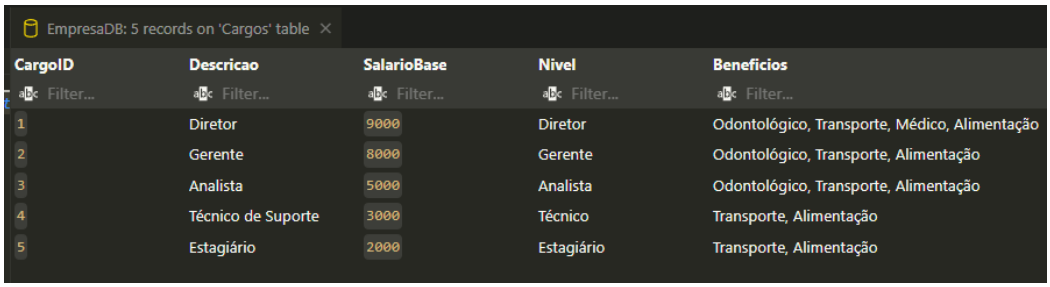
```python
#Populando as tabelas criadas com arquivos CSV ==================================
def inserir_dados(nome_tabela, arquivo_csv):
    try:
        # Conectar ao banco de dados
        conn = sqlite3.connect('DataDB.db')

        # Carregar dados do arquivo CSV para um DataFrame
        df_csv = pd.read_csv(arquivo_csv, delimiter=';', na_values='NULL')

        # Carregar dados existentes da tabela para um DataFrame
        df_db = pd.read_sql_query(f"SELECT * FROM {nome_tabela}", conn)

        # Salvar o número de linhas antes da inserção
        linhas_antes = len(df_db)

        # Concatenar os DataFrames e remover duplicatas
        df = pd.concat([df_db, df_csv]).drop_duplicates()

        # Salvar o número de linhas após a inserção
        linhas_depois = len(df)

        # Inserir dados na tabela do banco de dados usando to_sql
        df.to_sql(nome_tabela, conn, if_exists='replace', index=False)

        # Se o número de linhas aumentou, imprimir a mensagem de sucesso
        if linhas_depois > linhas_antes:
            print(f"\nDados inseridos na tabela {nome_tabela} com sucesso!")
        conn.close()

    except Exception as e:
        print(f"\nErro ao inserir dados na tabela {nome_tabela}: {e}")

# Inserir dados na tabela Funcionarios
inserir_dados('Funcionarios', 'funcionarios.csv')

# Inserir dados na tabela Cargos
inserir_dados('Cargos', 'cargos.csv')

# Inserir dados na tabela Departamentos
inserir_dados('Departamentos', 'departamento.csv')

# Inserir dados na tabela HistoricoSalarios
inserir_dados('HistoricoSalarios', 'historico_salarios.csv')

# Inserir dados na tabela Dependentes
inserir_dados('Dependentes', 'dependentes.csv')
```

Populando as tabelas usando Pandas e uma função no Python

```
Tabela Funcionarios já existe.

==========================================================================

Tabela Cargos já existe.

==========================================================================

Tabela Departamentos já existe.

==========================================================================

Tabela HistoricoSalarios já existe.

==========================================================================

Tabela Dependentes já existe.

==========================================================================
```

Console evidenciando que as tabelas não foram criadas pois já existem no database





Database com as 5 tabelas criadas e com todas elas populadas pelos respectivos arquivos .CSV

Queries SQL

```python
# Função para conectar ao banco de dados e executar uma consulta SQL
def executar_consulta_sql(query):
    conn = sqlite3.connect('DataDB.db')
    cursor = conn.cursor()
    cursor.execute(query)
    resultados = cursor.fetchall()
    conn.close()
    return resultados
```

Função criada para fazer os queries utilizando comandos SQL no Python

1. Listar individualmente as tabelas de: Funcionários, Cargos, Departamentos, Histórico de Salários e Dependentes em ordem crescente. (SQL no Python)

```python
# Consulta 1: Listar individualmente as tabelas em ordem crescente
def listar_tabela(tabela):
    query = f"SELECT * FROM {tabela} ORDER BY {tabela[:-1]}ID;"  #Removendo 's' para obter o nome da chave primária
    resultados = executar_consulta_sql(query)
    print(f"Conteúdo da tabela '{tabela}':")
    for linha in resultados:
        print(linha)
    print("\n======================================================================")
```

```
======================================================================
Conteúdo da tabela 'Funcionarios':
(1, 'Leighton Marrow', 1.0, 1, 9740, 'Feminino')
(2, 'Jamie Tolliver', 2.0, 1, 8500, 'Feminino')
(3, 'Leonardo Gulliver', 2.0, 5, 8200, 'Masculino')
(4, 'Marvin Guye', 3.0, 2, 5200, 'Masculino')
(5, 'Jonas Fairwright', 1.0, 2, 9750, 'Masculino')
(6, 'Mollie Goodrye', 4.0, 3, 3540, 'Feminino')
(7, 'Stella Bordwell', 3.0, 5, 5400, 'Feminino')
(8, 'Keith Dallas', 4.0, 4, 3600, 'Masculino')
(9, 'Jessica Foxley', 5.0, 4, 2400, 'Feminino')
(10, 'Garreth Wright', 5.0, 3, 2300, 'Masculino')
(11, 'Gillian Sowsbury', None, 4, 1100, 'Feminino')
(12, 'Richard Folley', 5.0, 4, 2000, 'Masculino')

======================================================================
Conteúdo da tabela 'Cargos':
(1, 'Diretor', 9000, 'Diretor', 'Odontológico, Transporte, Médico, Alimentação')
(2, 'Gerente', 8000, 'Gerente', 'Odontológico, Transporte, Alimentação')
(3, 'Analista', 5000, 'Analista', 'Odontológico, Transporte, Alimentação')
(4, 'Técnico de Suporte', 3000, 'Técnico', 'Transporte, Alimentação')
(5, 'Estagiário', 2000, 'Estagiário', 'Transporte, Alimentação')

======================================================================
Conteúdo da tabela 'Departamentos':
(1, 'TI', 2, 7, 'Setor responsável pela arquitetura digital da empresa')
(2, 'Marketing', 3, 7, 'Setor responsável pela divulgação da empresa')
(3, 'Vendas', 3, 5, 'Setor responsável pelas transações da empresa')
(4, 'Suporte Técnico', 2, 4, 'Setor responsável pela manutenção e suporte à empresa')
(5, 'Financeiro', 3, 5, 'Setor de gestão financeira da empresa')

======================================================================
```

```
======================================================================
Conteúdo da tabela 'HistoricoSalarios':
(1, 1, '2023-01-05', 9740)
(2, 1, '2023-02-05', 9740)
(3, 1, '2023-03-05', 9800)
(4, 1, '2023-04-05', 9800)
(5, 1, '2023-05-05', 9800)
(6, 1, '2023-06-05', 9900)
(7, 2, '2023-01-05', 8500)
(8, 2, '2023-02-05', 8550)
(9, 2, '2023-03-05', 8550)
(10, 2, '2023-04-05', 8550)
(11, 2, '2023-05-05', 8600)
(12, 2, '2023-06-05', 8650)
(13, 3, '2023-01-05', 8200)
(14, 3, '2023-02-05', 8200)
(15, 3, '2023-03-05', 8240)
(16, 3, '2023-04-05', 8240)
(17, 3, '2023-05-05', 8280)
(18, 3, '2023-06-05', 8300)
(19, 4, '2023-01-05', 5200)
(20, 4, '2023-02-05', 5200)
```

```
======================================================================
Conteúdo da tabela 'Dependentes':
(1, 1, 'Maddison DeLuca', 'Feminino', 4)
(2, 1, 'Jenna Ashley', 'Feminino', 6)
(3, 2, 'Kasey Valenzuela', 'Não-Binário', 20)
(4, 2, 'Everly Little', 'Feminino', 14)
(5, 3, 'Laurie Dillon', 'Feminino', 8)
(6, 3, 'Frances Chapman', 'Masculino', 3)
(7, 4, 'Kelly Noble', 'Feminino', 16)
(8, 4, 'Ashley Durham', 'Feminino', 10)
(9, 5, 'Louie Cooke', 'Masculino', 1)
(10, 5, 'Jade Cole', 'Feminino', 5)
(11, 5, 'Ronnie Palmer', 'Agênero', 23)
(12, 6, 'Georgie Irwin', 'Masculino', 18)
(13, 6, 'Marissa Norman', 'Feminino', 9)
(14, 6, 'Aston Dominguez', 'Agênero', 17)
(15, 6, 'Kaya Hawkins', 'Feminino', 21)
(16, 7, 'Dillan Abbott', 'Masculino', 14)
(17, 7, 'Nana Wallace', 'Feminino', 9)
(18, 8, 'Dana Vasquez', 'Feminino', 12)
(19, 8, 'Regan Farmer', 'Masculino', 19)
(20, 9, 'Malik Boyer', 'Não-Binário', 23)
(21, 9, 'Ashton Underwood', 'Masculino', 15)
(22, 10, 'Taylor Glass', 'Agênero', 18)
(23, 10, 'Ty Combs', 'Masculino', 13)
(24, 10, 'Kaden Bowman', 'Não-Binário', 22)
(25, 11, 'Charley Clayton', 'Masculino', 2)
(26, 11, 'Gabriel Green', 'Masculino', 25)
(27, 11, 'Maya Castillo', 'Feminino', 8)
(28, 12, 'Ollie Proctor', 'Não-Binário', 16)
(29, 12, 'Corey Riddle', 'Feminino', 7)
(30, 12, 'Callan Whitaker', 'Masculino', 12)
```

2. Listar os funcionários, com seus cargos, departamentos e os respectivos dependentes. (SQL no Python)

```python
# Consulta 2: Listar os funcionários com cargos, departamentos e os respectivos dependentes
def listar_funcionarios_com_info():
    query = """
        SELECT f.Nome AS Funcionario, c.Descricao AS Cargo, d.NomeDepartamento, dep.Nome AS Dependente
        FROM Funcionarios f
        JOIN Cargos c ON f.CargoID = c.CargoID
        JOIN Departamentos d ON f.DepartamentoID = d.DepartamentoID
        LEFT JOIN Dependentes dep ON f.FuncionarioID = dep.FuncionarioID;
        """
    resultados = executar_consulta_sql(query)
    print("Funcionários com informações completas (Nome, Cargo, Departamento, Dependente):")
    for linha in resultados:
        print(linha)
    print("\n========================================================================")
```

```
========================================================================
Funcionários com informações completas (Nome, Cargo, Departamento, Dependente):
('Leighton Marrow', 'Diretor', 'TI', 'Jenna Ashley')
('Leighton Marrow', 'Diretor', 'TI', 'Maddison DeLuca')
('Jamie Tolliver', 'Gerente', 'TI', 'Everly Little')
('Jamie Tolliver', 'Gerente', 'TI', 'Kasey Valenzuela')
('Leonardo Gulliver', 'Gerente', 'Financeiro', 'Frances Chapman')
('Leonardo Gulliver', 'Gerente', 'Financeiro', 'Laurie Dillon')
('Marvin Guye', 'Analista', 'Marketing', 'Ashley Durham')
('Marvin Guye', 'Analista', 'Marketing', 'Kelly Noble')
('Jonas Fairwright', 'Diretor', 'Marketing', 'Jade Cole')
('Jonas Fairwright', 'Diretor', 'Marketing', 'Louie Cooke')
('Jonas Fairwright', 'Diretor', 'Marketing', 'Ronnie Palmer')
('Mollie Goodrye', 'Técnico de Suporte', 'Vendas', 'Aston Dominguez')
('Mollie Goodrye', 'Técnico de Suporte', 'Vendas', 'Georgie Irwin')
('Mollie Goodrye', 'Técnico de Suporte', 'Vendas', 'Kaya Hawkins')
('Mollie Goodrye', 'Técnico de Suporte', 'Vendas', 'Marissa Norman')
('Stella Bordwell', 'Analista', 'Financeiro', 'Dillan Abbott')
('Stella Bordwell', 'Analista', 'Financeiro', 'Nana Wallace')
('Keith Dallas', 'Técnico de Suporte', 'Suporte Técnico', 'Dana Vasquez')
('Keith Dallas', 'Técnico de Suporte', 'Suporte Técnico', 'Regan Farmer')
('Jessica Foxley', 'Estagiário', 'Suporte Técnico', 'Ashton Underwood')
('Jessica Foxley', 'Estagiário', 'Suporte Técnico', 'Malik Boyer')
('Garreth Wright', 'Estagiário', 'Vendas', 'Kaden Bowman')
('Garreth Wright', 'Estagiário', 'Vendas', 'Taylor Glass')
('Garreth Wright', 'Estagiário', 'Vendas', 'Ty Combs')
('Richard Folley', 'Estagiário', 'Suporte Técnico', 'Callan Whitaker')
('Richard Folley', 'Estagiário', 'Suporte Técnico', 'Corey Riddle')
('Richard Folley', 'Estagiário', 'Suporte Técnico', 'Ollie Proctor')
```

3. Listar os funcionários que tiveram aumento salarial nos últimos 3 meses. (SQL no Python)

```python
# Consulta 3: Listar os funcionários que tiveram aumento salarial nos últimos 3 meses
def listar_funcionarios_com_aumento():
    query = """
      SELECT f.Nome
      FROM Funcionarios f
      JOIN HistoricoSalarios hs ON f.FuncionarioID = hs.FuncionarioID
      WHERE hs.Data BETWEEN '2023-01' AND '2023-06'
      GROUP BY f.Nome
      HAVING MIN(hs.Salario) != MAX(hs.Salario);
      """
    resultados = executar_consulta_sql(query)
    print("Funcionários com aumento salarial nos últimos 3 meses:")
    for linha in resultados:
        print(linha[0])
    print("\n========================================================================")
```

```
========================================================================
Funcionários com aumento salarial nos últimos 3 meses:
Jamie Tolliver
Jonas Fairwright
Leighton Marrow
Leonardo Gulliver
Richard Folley
Stella Bordwell
```

4. Listar a média de idade dos filhos dos funcionários por departamento. (SQL no Python)

```python
# Consulta 4: Listar a média de idade dos filhos dos funcionários por departamento
def listar_media_idade_filhos_por_departamento():
    query = """
        SELECT d.NomeDepartamento, ROUND(AVG(dep.Idade)) AS MediaIdade
        FROM Departamentos d
        JOIN Funcionarios f ON d.DepartamentoID = f.DepartamentoID
        JOIN Dependentes dep ON f.FuncionarioID = dep.FuncionarioID
        GROUP BY d.NomeDepartamento;
        """
    resultados = executar_consulta_sql(query)
    print("Média de idade dos filhos por departamento:")
    for linha in resultados:
        print(linha)
    print("\n========================================================================")
```

```
========================================================================
Média de idade dos filhos por departamento:
('Financeiro', 9.0)
('Marketing', 11.0)
('Suporte Técnico', 14.0)
('TI', 11.0)
('Vendas', 17.0)
```

5. Listar qual estagiário possui filho. (SQL no Python)

```python
# Consulta 5: Listar qual estagiário possui filho
def listar_estagiarios_com_filho():
    query = """
        SELECT DISTINCT f.Nome
        FROM Funcionarios f
        JOIN Cargos c ON f.CargoID = c.CargoID
        JOIN Dependentes dep ON f.FuncionarioID = dep.FuncionarioID
        WHERE c.Descricao = 'Estagiário';
        """
    resultados = executar_consulta_sql(query)
    print("Estagiários que possuem filho:")
    for linha in resultados:
        print(linha[0])
    print("\n========================================================================")
```

```
========================================================================
Estagiários que possuem filho:
Jessica Foxley
Garreth Wright
Richard Folley
```

```python
#Preparando os Dataframes
df_historico_salarios = pd.read_csv('historico_salarios.csv', sep=';', decimal=',')
df_cargos = pd.read_csv('cargos.csv', sep=';', decimal=',')
df_departamentos = pd.read_csv('departamento.csv', sep=';', decimal=',')
df_funcionarios = pd.read_csv('funcionarios.csv', sep=';', decimal=',')
df_dependentes = pd.read_csv('dependentes.csv', sep=';', decimal=',')
```

Criando DataFrames através dos arquivos .CSV para manipulação com Pandas

6. Listar o funcionário que teve o salário médio mais alto. (Python + Pandas)

```python
#Consulta 6 - Listar o funcionário com o salário médio mais alto
def listar_funcionario_com_maior_salario_medio():
    media_salarios = df_historico_salarios.groupby('FuncionarioID')['Salario'].mean()
    funcionario_com_maior_salario_medio = media_salarios.idxmax() - 1 #Acertando o índice para corresponder ao FuncionarioID
    nome_funcionario = df_funcionarios.loc[funcionario_com_maior_salario_medio, 'Nome']
    print(f'O funcionário com o maior salário médio é {nome_funcionario}, com um salário médio de R${media_salarios.max():.2f}.')
    print("\n======================================================================")
```

```
======================================================================
O funcionário com o maior salário médio é Jonas Fairwright, com um salário médio de R$9816.67.

======================================================================
```

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

```python
#Consulta 7 - Listar o analista que é pai de duas meninas
def listar_analistas_com_duas_filhas():

    # Filtra funcionários que são analistas homens
    analistas_homens = df_funcionarios[
        (df_funcionarios["CargoID"] == 3) & (df_funcionarios["Genero"] == "Masculino")
    ]

    # Filtra dependentes que são filhas do gênero feminino
    filhas = df_dependentes[(df_dependentes["Genero"] == "Feminino")]

    # Junta os DataFrames de analistas homens e filhas com base no FuncionarioID
    analistas_com_filhas = analistas_homens.merge(filhas, on="FuncionarioID")

    # Agrupa por FuncionarioID e conta o número de filhas
    contagem_filhas = analistas_com_filhas.groupby("FuncionarioID")["DependenteID"].count()

    # Filtra analistas com exatamente duas filhas
    analistas_com_duas_filhas = contagem_filhas[contagem_filhas == 2].index.tolist()

    # Filtra o DataFrame original de funcionários para incluir apenas os analistas com duas filhas
    resultado = df_funcionarios[df_funcionarios["FuncionarioID"].isin(analistas_com_duas_filhas)]

    print(f"Analistas que são pais de duas filhas: {resultado['Nome'].tolist()}")
    print("\n======================================================================")
```

```
======================================================================
Analistas que são pais de duas filhas: ['Marvin Guye']

======================================================================
```

8. Listar o analista que tem o salário mais alto, e que ganhe entre 5000 e 9000. (Python + Pandas)

```python
#Consulta 8 - Listar o analista com o salário mais alto e que ganhe entre R$ 5000 e R$ 9000
def listar_analista_salario_mais_alto():

    # Filtra funcionários que são analistas e estão na faixa salarial desejada
    analistas_faixa_salarial = df_funcionarios[
        (df_funcionarios["CargoID"] == 3) & (df_funcionarios["Salario"] >= 5000) & (df_funcionarios["Salario"] <= 9000)
    ]

    # Encontra o analista com o salário mais alto
    analista_salario_mais_alto = analistas_faixa_salarial.loc[analistas_faixa_salarial["Salario"].idxmax()]

    # Retorna o nome e salário do analista
    resultado = analista_salario_mais_alto[["Nome", "Salario"]]

    print(f"Analista com o salário mais alto entre R$ 5000 e R$ 9000: {resultado['Nome']} com salário de R${resultado['Salario']:.2f}")
    print("\n======================================================================")
```

```
======================================================================
Analista com o salário mais alto entre R$ 5000 e R$ 9000: Stella Bordwell com salário de R$5400.00

======================================================================
```

9. Listar qual departamento possui o maior número de dependentes. (Python + Pandas)

```python
#Consulta 9 - Listar qual departamento possui o maior número de dependentes
def listar_departamento_mais_dependentes():
    # Junta os DataFrames de funcionários e dependentes com base no FuncionarioID
    funcionarios_com_dependentes = df_funcionarios.merge(df_dependentes, on="FuncionarioID", how="left")

    # Junta o DataFrame resultante com o DataFrame de departamentos com base no DepartamentoID
    funcionarios_dependentes_departamentos = funcionarios_com_dependentes.merge(df_departamentos, on="DepartamentoID", how="left")

    # Agrupa por departamento e conta o número de dependentes
    contagem_dependentes = funcionarios_dependentes_departamentos.groupby("NomeDepartamento")["DependenteID"].count()

    # Encontra o departamento com o maior número de dependentes
    departamento_mais_dependentes = contagem_dependentes.idxmax()
    numero_dependentes = contagem_dependentes.max()

    # Retorna o nome do departamento e o número de dependentes
    print(f"O departamento com o maior número de dependentes é o {departamento_mais_dependentes}, com {numero_dependentes} dependentes.")
    print("\n======================================================================")
```

```
======================================================================
O departamento com o maior número de dependentes é o Suporte Técnico, com 10 dependentes.

======================================================================
```

10. Listar a média de salário por departamento em ordem decrescente. (Python + Pandas)

```python
#Consulta 10 - Listar a média de salário por departamento, em ordem decrescente
def listar_media_salarial_por_departamento():

    # Junta os DataFrames de funcionários e departamentos com base no DepartamentoID
    funcionarios_departamentos = df_funcionarios.merge(df_departamentos, on="DepartamentoID")

    # Calcula a média salarial por departamento e ordena em ordem decrescente
    media_salarial = funcionarios_departamentos.groupby("NomeDepartamento")["Salario"].mean().sort_values(ascending=False)

    #Oraganizando o resultado em um DataFrame e nomeando corretamente as colunas
    df_resultado = pd.DataFrame({'Departamento': media_salarial.index, 'Média Salarial': media_salarial.values})

    print("Média salarial por departamento (em ordem decrescente):")
    print(df_resultado)
    print("\n======================================================================")
```

```
======================================================================
Média salarial por departamento (em ordem decrescente):
      Departamento  Média Salarial
0               TI          9120.0
1        Marketing          7475.0
2       Financeiro          6800.0
3           Vendas          2920.0
4  Suporte Técnico          2275.0
```