

Fundamentos de BD (Márcio Victorino)

Partes do Projeto:

1. Introdução.
2. Modelo de dados Relacional.
3. O script SQL que gerou o banco de dados.
4. Apresentar o processo de ETL (Extract, Transform, Load) para importação dos dados para o banco de dados.
5. Utilização de pelo menos uma View.
6. Utilização de pelo menos uma Procedure (com comandos condicionais).
7. Utilização de pelo menos um trigger (com comandos condicionais).
8. No mínimo 5 Consultas SQL (a complexidade da consulta será avaliada).

Data Limite para a Entrega da Parte Escrita: 02/06/2023

Alunos:

Marcelo Anselmo de Souza Filho

- Matrícula: **231109719**
- Email: **marcelofilho@mpf.mp.br**

Arivaldo Gonçalves de Freitas Junior

- Matrícula: **231109620**
- Email: **arivaldofreitas@correios.com.br**

Luciana Maria de Araujo Freitas

- Matrícula: **231109700**
 - Email: **luciana@mpdft.mp.br**
-

1. Introdução

Sobre: Este estudo aborda a diferença salarial entre homens e mulheres na área de TI durante a pandemia. Ele também explora possíveis cenários para analisar a disparidade salarial e de desligamento entre gêneros na área de tecnologia, antes e após a pandemia. Utilizou-se dados a nível do indivíduo, de 2018 e 2019 (antes da pandemia) e de 2020 e 2021 (durante a pandemia), obtidos da Relação Anual de Informações Sociais (Rais), que proporciona dados oficiais sobre o mercado de trabalho no Brasil.

Resultados: No geral, ficou evidente que, no período analisado, a quantidade de homens na TI é muito maior do que a de mulheres. Constatou-se que a remuneração média das mulheres é maior que a dos homens apenas na região nordeste. Além disso, a quantidade de desligamento de homens e mulheres é maior em 2021 (691.982) e a menor é de 2019 (162.073).

Tecnologias utilizadas:

- BD: Mysql (docker)
- Linguagem: Python
- Dados: RAIS
- Ambiente de DEV: VsCode + Jupyter Notebook

1.1 Carregando Bibliotecas necessárias

```
In [ ]: import mysql.connector
import warnings
import base64
from IPython.display import Image, display
import pandas as pd

# MERMAID: para visualização de Diagramas no Markdown
def mm(graph):
    graph = graph.replace("\n", " ")
    graphbytes = graph.encode("ascii")
    base64_bytes = base64.b64encode(graphbytes)
    base64_string = base64_bytes.decode("ascii")
    display(Image(url="https://mermaid.ink/img/" + base64_string))

# Ignorando Warnings do Python
warnings.filterwarnings("ignore")

# Conexão com o Banco de Dados
cnx = mysql.connector.connect(user='root', password='root', database='projfbd')
```

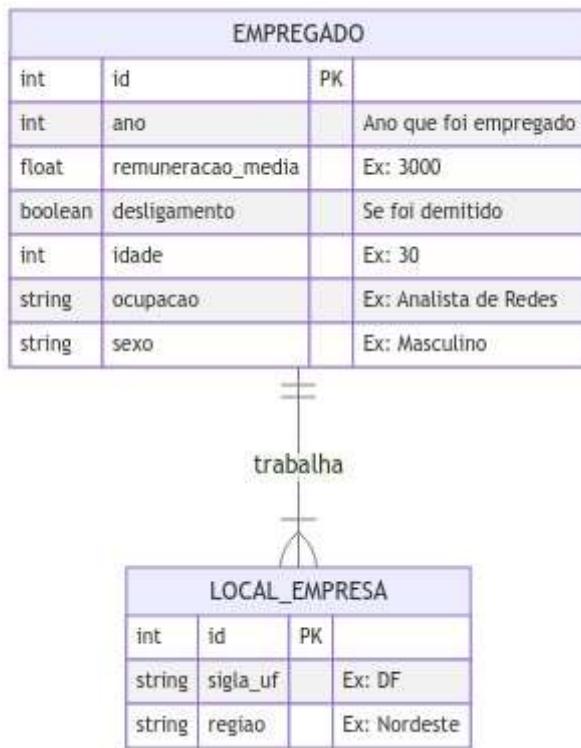
2. Modelo de dados Relacional

2.1 Modelo Conceitual

<https://mermaid.js.org/syntax/entityRelationshipDiagram.html>

Integrando jupyter com mermaid: <https://mermaid.js.org/config/Tutorials.html#jupyter-integration-with-mermaid-js>

```
In [ ]: mm"""
erDiagram
    EMPREGADO {
        int id PK
        int ano "Ano que foi empregado"
        float remuneracao_media "Ex: 3000"
        boolean desligamento "Se foi demitido"
        int idade "Ex: 30"
        string ocupacao "Ex: Analista de Redes"
        string sexo "Ex: Masculino"
    }
    LOCAL_EMPRESA {
        int id PK
        string sigla_uf "Ex: DF"
        string regiao "Ex: Nordeste"
    }
    EMPREGADO ||--|{ LOCAL_EMPRESA : trabalha
""")
```



2.2 Modelo Lógico

2.2.1 Normalização

1º FN: Uma relação está em 1FN se e somente se todos os seus atributos contêm apenas valores atômicos (simples, indivisíveis)

RESPOSTA: A relação está na 1FN, pois todos os atributos são simples e indivisíveis.

2º FN: Uma relação encontra-se na 2FN se e somente se estiver em 1FN e não contém dependências parciais.

```

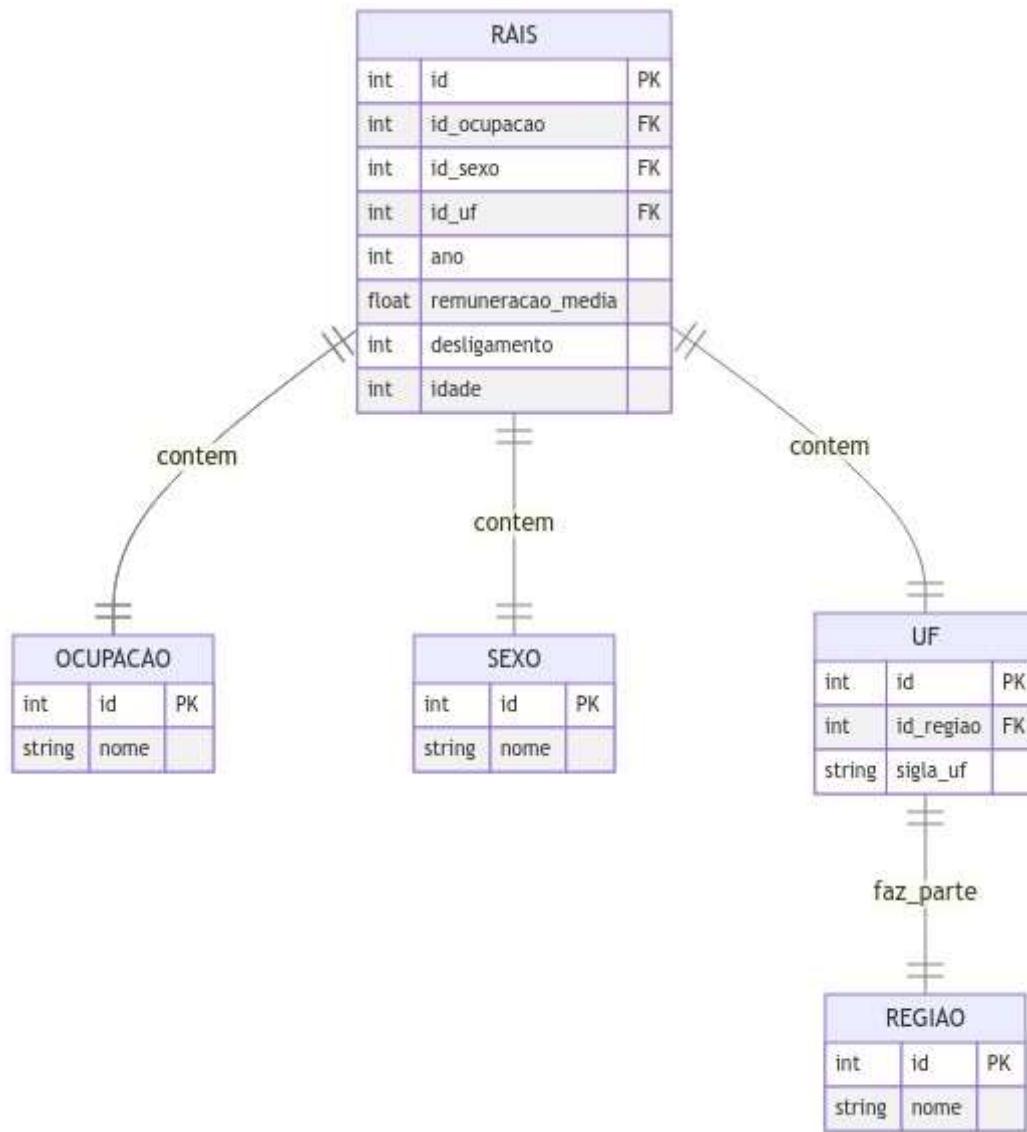
In [ ]: mm("""
erDiagram
    RAIS ||--|| OCUPACAO : contem
    RAIS ||--|| OCUPACAO : contem
    RAIS ||--|| SEXO : contem
    RAIS ||--|| UF : contem
    UF ||--|| REGIAO : faz_parte
    RAIS {
        int id PK
        int id_ocupacao FK
        int id_sexo FK
        int id_uf FK
        int ano
        float remuneracao_media
        int desligamento
        int idade
    }
    OCUPACAO {
        int id PK
        string nome
    }
    SEXO {
        int id PK
    }
""")

```

```

        string nome
    }
    UF {
        int id PK
        int id_regiao FK
        string sigla_uf
    }
    REGIAO {
        int id PK
        string nome
    }
"""
)

```



3º FN: Uma relação está em 3FN se e somente se estiver na 2FN e nenhum atributo não-primo (isto é, que não seja membro de uma chave) for transitivamente dependente da chave primária.

RESPOSTA: A relação está na 3FN, pois não há dependência transitiva.

2.2.2 Especificação do BD

- OCUPACAO (id, nome)
- SEXO (id, nome)
- REGIAO (id, nome)

- UF (id, id_regiao, sigla_uf)
 - id_regiao **REFERENCIA** REGIAO(id)
- EMPREGADO (id, id_ocupacao, id_sexo, id_uf, ano, remuneracao_media, desligamento, idade)
 - id_ocupacao **REFERENCIA** OCUPACAO(id)
 - id_sexo **REFERENCIA** SEXO(id)
 - id_uf **REFERENCIA** UF(id)

2.3 Modelo Físico

```

OCUPACAO (
    id INT NOT NULL,
    nome VARCHAR(255) NOT NULL,
    PRIMARY KEY (id)
);

SEXO (
    id INT NOT NULL,
    nome VARCHAR(9) NOT NULL,
    PRIMARY KEY (id)
);

REGIAO (
    id INT NOT NULL,
    nome VARCHAR(12) NOT NULL,
    PRIMARY KEY (id)
);

UF (
    id INT NOT NULL,
    id_regiao INT NOT NULL,
    nome VARCHAR(2) NOT NULL,
    PRIMARY KEY (id),
    FOREIGN KEY (id_regiao) REFERENCES REGIAO(id)
);

EMPREGADO (
    id INT NOT NULL AUTO_INCREMENT,
    id_ocupacao INT NOT NULL,
    id_sexo INT NOT NULL,
    id_uf INT NOT NULL,
    id_ano INT NOT NULL,
    remuneracao_media FLOAT,
    desligamento INT,
    idade INT NOT NULL,
    PRIMARY KEY (id),
    FOREIGN KEY (id_ocupacao) REFERENCES OCUPACAO(id),
    FOREIGN KEY (id_sexo) REFERENCES SEXO(id),
    FOREIGN KEY (id_ano) REFERENCES ANO(id),
    FOREIGN KEY (id_uf) REFERENCES UF(id)
);

```

3. O script SQL que gerou o banco de dados.

```
-- ===== CRIANDO DB =====

DROP DATABASE IF EXISTS projfbd;
CREATE DATABASE projfbd DEFAULT CHARACTER SET 'utf8';
USE projfbd;

-- ===== CRIANDO AS TABELAS =====

CREATE TABLE
OCUPACAO (
    id INT NOT NULL,
    nome VARCHAR(255) NOT NULL,
    PRIMARY KEY (id)
) ENGINE = InnoDB;

CREATE TABLE
SEXO (
    id INT NOT NULL,
    nome VARCHAR(9) NOT NULL,
    PRIMARY KEY (id)
) ENGINE = InnoDB;

CREATE TABLE
REGIAO (
    id INT NOT NULL,
    nome VARCHAR(12) NOT NULL,
    PRIMARY KEY (id)
) ENGINE = InnoDB;

CREATE TABLE
UF (
    id INT NOT NULL,
    id_regiao INT NOT NULL,
    nome VARCHAR(2) NOT NULL,
    PRIMARY KEY (id),
    FOREIGN KEY (id_regiao) REFERENCES REGIAO(id)
) ENGINE = InnoDB;

CREATE TABLE
EMPREGADO (
    id INT NOT NULL AUTO_INCREMENT,
    id_ocupacao INT NOT NULL,
    id_sexo INT NOT NULL,
    id_uf INT NOT NULL,
    ano INT NOT NULL,
    remuneracao_media FLOAT,
    desligamento INT,
    idade INT NOT NULL,
    PRIMARY KEY (id),
    FOREIGN KEY (id_ocupacao) REFERENCES OCUPACAO(id),
    FOREIGN KEY (id_sexo) REFERENCES SEXO(id),
    FOREIGN KEY (id_uf) REFERENCES UF(id)
) ENGINE = InnoDB;
```

4. Apresentar o processo de ETL (Extract, Transform, Load) para importação dos dados para o banco de dados.

Os passos do ETL estão contidos nos arquivos abaixo:

- Juntando os anos de 2018 a 2021 em um único arquivo.
 - Arquivo: [03_juntando.todos.anos.ipynb](#)
- Juntando os dados da RAIS com os dados de sexo e raça.
 - Arquivo: [04_fazendo_join_ocup_sexo.ipynb](#)

Resumindo os passos do ETL

Extração

1. Após filtrar os dados da tabela Relação Anual de Informações Sociais ([RAIS](#)) pelos anos de 2018 a 2021, foi feito o filtro pelos IDs de cargos de Tecnologia da informação conforme a Classificação Brasileira de Ocupações ([CBO](#)).

- 212205: Engenheiro de Aplicativos em Computacao
- 212210: Engenheiro de Equipamentos em Computacao
- 212215: Engenheiros de Sistemas Operacionais em Computacao
- 212305: Administrador de Banco de Dados
- 212310: Administrador de Redes
- 212315: Administrador de Sistemas Operacionais
- 212320: Administrador em Segurança da Informação
- 212405: Analista de Desenvolvimento de Sistemas
- 212410: Analista de Redes e de Comunicacao de Dados
- 212415: Analista de Sistemas de Automacao
- 212420: Analista de Suporte Computacional
- 317105: Programador de Internet
- 317110: Programador de Sistemas de Informacao
- 317115: Programador de Maquinas - Ferramenta com Comando Numerico
- 317120: Programador de Multimidia
- 317205: Operador de Computador (Inclusive Microcomputador)
- 317210: Tecnico de Apoio ao Usuario de Informatica (Helpdesk)

2. Em seguida, obtivemos os dados dos profissionais de TI no Brasil entre os anos de 2018 a 2019

- Quantidade **total**: 1.543.009
- Quantidade **por ano**:
 - 2021: 691.982

- 2018: 466.852
- 2020: 222.102
- 2019: 162.073

Transformação

1. Primeiro, juntamos os dados com a planilha de Sexo

1, Masculino
2, Feminino
-1, Ignorado

2. Em seguida, juntamos os dados com a planilha com o nome dos Cargos

212205, Engenheiro de Aplicativos em Computacao
212210, Engenheiro de Equipamentos em Computacao
...

3. Logo após, selecionamos apenas a colunas necessárias e as renomeamos

4. Por fim, alteramos todos os dados com "idade" = 0 para o mínimo de 14 anos (que é o menor valor, retirando o zero)

Carregamento

Fazendo INSERT dos dados das tabelas (todas menos a tabela EMPREGADO) no Banco:

```
-- Active: 1684245138463@@127.0.0.1@3306@projfbd

USE projfbd;

-- ====== INSERINDO OS DADOS (CARGA) =====

INSERT INTO OCUPACAO (id, nome)
VALUES (
    212405,
    'Analista de Desenvolvimento de Sistemas'
), (
    317110,
    'Programador de Sistemas de Informacao'
), (
    212420,
    'Analista de Suporte Computacional'
), (
    317210,
    'Tecnico de Apoio ao Usuario de Informatica (Helpdesk)'
), (
    212410,
    'Analista de Redes e de Comunicacao de Dados'
), (
    317205,
    'Operador de Computador (Inclusive Microcomputador)'
), (
    212315,
    'Administrador de Sistemas Operacionais'
), (
    212415,
    'Analista de Sistemas de Automacao'
```

```

        ),
        212310,
        'Administrador de Redes'
    ), (
        212205,
        'Engenheiro de Aplicativos em Computacao'
    ), (
        212305,
        'Administrador de Banco de Dados'
    ), (
        212320,
        'Administrador em Segurança da Informação'
    ), (
        317105,
        'Programador de Internet'
    ), (
        212215,
        'Engenheiros de Sistemas Operacionais em Computacao'
    ), (
        317115,
        'Programador de Maquinas - Ferramenta com Comando Numerico'
    ), (
        317120,
        'Programador de Multimidia'
    ), (
        212210,
        'Engenheiro de Equipamentos em Computacao'
    );

```

-- ===== TABELA: SEXO

```
INSERT INTO SEXO (id, nome) VALUES (1, 'Masculino'), (2, 'Feminino');
```

-- ===== TABELA: REGIAO

```
INSERT INTO REGIAO (id, nome)
VALUES (3, 'Sudeste'), (4, 'Sul'), (1, 'Nordeste'), (2, 'Norte'), (0, 'Centro-Oeste');
```

-- ===== TABELA: UF

```
INSERT INTO
    UF (id, id_regiao, nome)
VALUES (25, 3, 'SP'), (18, 3, 'RJ'), (10, 3, 'MG'), (22, 4, 'RS'), (17, 4, 'PR'),
(23, 4, 'SC'), (6, 0, 'DF'), (20, 2, 'RO'), (15, 1, 'PE'), (4, 1, 'BA'), (5, 1,
'CE'), (7, 3, 'ES'), (8, 0, 'GO'), (12, 0, 'MT'), (13, 2, 'PA'), (14, 1, 'PB'),
(2, 2, 'AM'), (11, 0, 'MS'), (19, 1, 'RN'), (9, 1, 'MA'), (16, 1, 'PI'), (1, 1,
'AL'), (24, 1, 'SE'), (26, 2, 'TO'), (0, 2, 'AC'), (3, 2, 'AP'), (21, 2, 'RR');
```

A tabela de EMPREGADO foi feita com o script python abaixo, pois continha mais de 1.5 milhões de registros

```
In [ ]: from __future__ import print_function
import pandas as pd
import mysql.connector
from datetime import date, datetime, timedelta

class CargaFullTabelaRAIS:
    def __init__(self, batch_size=1000, size_max=10000):
```

```

    self.batch_size = batch_size
    self.size_max = size_max
    self.cnx = mysql.connector.connect(
        user='root', password='root', database='projfbd')
    self.cursor = self.cnx.cursor()
    self.path_file_parquet = "../output/gold/rais_TODOS_ANOS_comJoin_RAIS_VINC_PUB.parquet.gz

    def carregar_CSV(self):
        # Mostrar mais colunas
        pd.set_option("display.max_columns", 100)
        pd.set_option('display.max_colwidth', 100)

        df = pd.read_parquet(self.path_file_parquet)

        qnt_total = len(df)

        print(f"""
            Quantidade de docs carregados: {qnt_total}
        """)
        return df

    def get_rais_from_pandas(self):
        df = self.carregar_CSV()
        counts = df[[
            "ocupacao_id", "sexo_id", "sigla_uf_id", "ano", "remuneracao_media", "desligamento"
        ]
        lista = counts.values.tolist()
        lista = [tuple(x) for x in lista]

        return lista

    def gerar_batch_insert_rais(self):
        print(f"""
            Inserindo os dados...
        """)

        rais_from_pandas = self.get_rais_from_pandas()

        for i in range(0, len(rais_from_pandas[:self.size_max]), self.batch_size):
            add_query = ("INSERT INTO EMPREGADO "
                        "(id_ocupacao, id_sexo, id_uf, ano, remuneracao_media, desligamento, idade"
                        "VALUES ")
            vals = ", ".join((f"({str(id_ocupacao)}, {str(id_sexo)}, {str(id_uf)}, {str(ano)}, {str(remuneracao_media)}, {str(desligamento)}, {str(idade)})"
            for id_ocupacao, id_sexo, id_uf, ano, remuneracao_media, desligamento, idade in rais_from_pandas[i:i+batch_size]))
            print(f"\n\n-- ===== INSERT a partir do dado {i}")
            # print(add_query + vals)
            try:
                self.cursor.execute(add_query + vals)
            except Exception as e:
                print(e)
                print(i)

            # ===== COMMIT =====
            self.cnx.commit()

    def init(self):
        self.gerar_batch_insert_rais()

    def __del__(self):
        self.cursor.close()
        self.cnx.close()

# ====== INICIANDO
CargaFullTabelaRAIS(batch_size=500000, size_max=2000000).init()

```

Inserindo os dados...

Quantidade de docs carregados: 1543009

```
-- === INSERT a partir do dado 0

-- === INSERT a partir do dado 500000

-- === INSERT a partir do dado 1000000

-- === INSERT a partir do dado 1500000
```

5. Utilização de pelo menos uma View.

```
CREATE VIEW VW_EMPREGADO_FULL AS
  SELECT
    e.*,
    OCUPACAO.nome AS ocupacao,
    SEXO.nome AS sexo,
    UF.nome AS uf,
    REGIAO.nome AS regiao
  FROM EMPREGADO as e
  INNER JOIN OCUPACAO ON e.id_ocupacao = OCUPACAO.id
  INNER JOIN SEXO ON e.id_sexo = SEXO.id
  INNER JOIN UF ON e.id_uf = UF.id
  INNER JOIN REGIAO ON UF.id_regiao = REGIAO.id;
```

```
In [ ]: pd.read_sql("""
SELECT * FROM VW_EMPREGADO_FULL
LIMIT 5;
""", cnx)
```

Out[]:

	id	id_ocupacao	id_sexo	id_uf	ano	remuneracao_media	desligamento	idade	ocupacao	sexo
0	7293	212410	1	6	2019	30772.5	0	61	Analista de Redes e de Comunicacao de Dados	Masculino
1	7292	212410	1	6	2019	24142.7	0	56	Analista de Redes e de Comunicacao de Dados	Masculino
2	7291	212410	1	6	2019	24667.7	0	58	Analista de Redes e de Comunicacao de Dados	Masculino
3	7290	212410	1	6	2019	23558.3	0	54	Analista de Redes e de Comunicacao de Dados	Masculino
4	7289	212410	1	6	2019	24537.7	0	57	Analista de Redes e de Comunicacao de Dados	Masculino

6. Utilização de pelo menos uma Procedure (com comandos condicionais).

DELIMITER \$\$

```
CREATE PROCEDURE IF NOT EXISTS PROC_SELECCIONAR_EMP_POR_ANO (IN ANO INT)
BEGIN
    SELECT * FROM `VW_EMPREGADO_FULL` as e WHERE ANO = e.ano;
END;

$$
DELIMITER;
```

In []:

```
pd.read_sql("""
CALL PROC_SELECCIONAR_EMP_POR_ANO(2018);
""", cnx).head(5)
```

Out[]:

	id	id_ocupacao	id_sexo	id_uf	ano	remuneracao_media	desligamento	idade	ocupacao
0	391322	212405	1	12	2018	10061.30	1	35	Analista de Desenvolvimento de Sistemas
1	391321	212315	1	12	2018	14284.80	1	34	Administrador de Sistemas Operacionais
2	391320	212320	1	12	2018	8000.01	1	45	Administrador em Segurança da Informação
3	391319	212215	1	12	2018	8000.01	1	39	Engenheiros de Sistemas Operacionais em Comput...
4	391318	212405	1	12	2018	4003.71	1	39	Analista de Desenvolvimento de Sistemas

7. Utilização de pelo menos um trigger (com comandos condicionais).

```
DELIMITER $$
```

```
CREATE TRIGGER IF NOT EXISTS TRIGGER_CHECK_INSERT_EMPREGADO
BEFORE INSERT ON EMPREGADO FOR EACH ROW BEGIN
    IF NEW.idade < 14 THEN SET NEW.idade = 14;
    END IF;
END;
```

```
$$
```

```
DELIMITER;
```

In []:

```
cursor = cnx.cursor()
cursor.execute(("INSERT INTO EMPREGADO "
                "(id_ocupacao, id_sexo, id_uf, ano, remuneracao_media, desligamento, idade) "
                "VALUES (212205, 1, 11, 2020, 111.11, 0, 11)"))

pd.read_sql("""
SELECT * FROM EMPREGADO
""", cnx).tail(1)
```

Out[]:

	id	id_ocupacao	id_sexo	id_uf	ano	remuneracao_media	desligamento	idade
1543012	1543013	212205	1	11	2020	111.11	0	14

8. No mínimo 5 Consultas SQL (a complexidade da consulta será avaliada).

8.1 Quantidade de empregos de TI por ano e gênero

```
SELECT ano, sexo, COUNT(*) as qnt_empregos
FROM `VW_EMPREGADO_FULL`
GROUP BY ano, sexo;
```

```
In [ ]: pd.read_sql("""
SELECT ano, sexo, COUNT(*) as qnt_empregos
FROM `VW_EMPREGADO_FULL`
GROUP BY ano, sexo;
""", cnx)
```

```
Out[ ]:   ano      sexo  qnt_empregos
0  2018  Feminino      101658
1  2018  Masculino     365194
2  2019  Feminino      25888
3  2019  Masculino     136185
4  2020  Feminino      44657
5  2020  Masculino     177447
6  2021  Feminino      147184
7  2021  Masculino     544798
```

8.2 Quantidade de cargos por gênero

```
SELECT
    ocupacao,
    sexo,
    COUNT(id_ocupacao) as qnt_cargos
FROM `VW_EMPREGADO_FULL`
GROUP BY id_ocupacao, sexo
ORDER BY ocupacao, sexo;
```

```
In [ ]: pd.read_sql("""
SELECT
    ocupacao,
    sexo,
    COUNT(id_ocupacao) as qnt_cargos
FROM `VW_EMPREGADO_FULL`
GROUP BY id_ocupacao, sexo
ORDER BY ocupacao, sexo;
""", cnx)
```

Out[]:

	ocupacao	sexo	qnt_cargos
0	Administrador de Banco de Dados	Feminino	3058
1	Administrador de Banco de Dados	Masculino	12033
2	Administrador de Redes	Feminino	1985
3	Administrador de Redes	Masculino	16767
4	Administrador de Sistemas Operacionais	Feminino	5190
5	Administrador de Sistemas Operacionais	Masculino	18905
6	Administrador em Segurança da Informação	Feminino	2929
7	Administrador em Segurança da Informação	Masculino	11750
8	Analista de Desenvolvimento de Sistemas	Feminino	108088
9	Analista de Desenvolvimento de Sistemas	Masculino	445439
10	Analista de Redes e de Comunicacao de Dados	Feminino	25878
11	Analista de Redes e de Comunicacao de Dados	Masculino	74867
12	Analista de Sistemas de Automacao	Feminino	4290
13	Analista de Sistemas de Automacao	Masculino	19202
14	Analista de Suporte Computacional	Feminino	48269
15	Analista de Suporte Computacional	Masculino	191574
16	Engenheiro de Aplicativos em Computacao	Feminino	2072
17	Engenheiro de Aplicativos em Computacao	Masculino	13237
18	Engenheiro de Equipamentos em Computacao	Feminino	333
19	Engenheiro de Equipamentos em Computacao	Masculino	2441
20	Engenheiros de Sistemas Operacionais em Computacao	Feminino	1340
21	Engenheiros de Sistemas Operacionais em Computacao	Masculino	9745
22	Operador de Computador (Inclusive Microcomputadores)	Feminino	23776
23	Operador de Computador (Inclusive Microcomputadores)	Masculino	59491
24	Programador de Internet	Feminino	2078
25	Programador de Internet	Masculino	11634
26	Programador de Maquinas - Ferramenta com Comando Numerico	Feminino	539
27	Programador de Maquinas - Ferramenta com Comando Numerico	Masculino	8755
28	Programador de Multimidia	Feminino	1025
29	Programador de Multimidia	Masculino	4592
30	Programador de Sistemas de Informacao	Feminino	56487
31	Programador de Sistemas de Informacao	Masculino	208124
32	Tecnico de Apoio ao Usuario de Informatica (Hardware)	Feminino	32050
33	Tecnico de Apoio ao Usuario de Informatica (Hardware)	Masculino	115068

8.3 Remuneração média por região

```

WITH remun_media AS (
    SELECT
        regiao,
        sexo,
        ROUND(AVG(remuneracao_media), 2) as media,
        ROUND(MIN(remuneracao_media), 2) as minimo,
        ROUND(MAX(remuneracao_media), 2) as maximo,
        ROUND(STD(remuneracao_media), 2) as desvio_padrao
    FROM
        `VW_EMPREGADO_FULL`
    WHERE
        remuneracao_media > 0
    GROUP BY
        regiao,
        sexo
)
SELECT
    *,
    ROUND(
        desvio_padrao - LAG(desvio_padrao, 1) OVER (
            ORDER BY
                regiao,
                sexo
        ),
        2
    ) as diff
FROM remun_media
ORDER BY regiao, sexo;

```

```

In [ ]: pd.read_sql(f"""
WITH remun_media AS (
    SELECT
        regiao,
        sexo,
        ROUND(AVG(remuneracao_media), 2) as media,
        ROUND(MIN(remuneracao_media), 2) as minimo,
        ROUND(MAX(remuneracao_media), 2) as maximo,
        ROUND(STD(remuneracao_media), 2) as desvio_padrao
    FROM
        `VW_EMPREGADO_FULL`
    WHERE
        remuneracao_media > 0
    GROUP BY
        regiao,
        sexo
)
SELECT
    *,
    ROUND(
        desvio_padrao - LAG(desvio_padrao, 1) OVER (
            ORDER BY
                regiao,
                sexo
        ),
        2
    ) as diff
FROM remun_media
ORDER BY regiao, sexo;
""", cnx)

```

Out[]:

	regiao	sexo	media	minimo	maximo	desvio_padrao	diff
0	Centro-Oeste	Feminino	6542.79	315.69	73685.48	7033.05	NaN
1	Centro-Oeste	Masculino	7186.03	111.11	85750.97	6985.15	-47.90
2	Nordeste	Feminino	3900.37	313.86	44387.25	3957.06	-3028.09
3	Nordeste	Masculino	3902.51	291.29	98543.80	3694.91	-262.15
4	Norte	Feminino	3060.56	287.07	110716.20	2904.23	-790.68
5	Norte	Masculino	4062.80	287.53	105844.80	4050.40	1146.17
6	Sudeste	Feminino	5293.50	289.87	146400.00	4765.07	714.67
7	Sudeste	Masculino	6094.63	288.80	161407.66	5462.57	697.50
8	Sul	Feminino	4123.61	302.60	88432.50	3672.33	-1790.24
9	Sul	Masculino	4745.27	288.14	84126.00	3856.57	184.24

8.4 Quantidade de demissões por região

```
WITH qnt_desligs AS (
    SELECT
        ano,
        regiao,
        sexo,
        COUNT(desligamento) as qnt_desligamento
    FROM
        `VW_EMPREGADO_FULL`
    GROUP BY
        ano,
        regiao,
        sexo
)
SELECT
    *,
    ROUND(
        qnt_desligamento - LAG(qnt_desligamento, 1) OVER (
            ORDER BY
                ano,
                regiao,
                sexo
        ),
        2
    ) as diff
FROM qnt_desligs as q
ORDER BY ano, regiao, sexo;
```

In []:

```
pd.read_sql(f"""
WITH qnt_desligs AS (
    SELECT
        ano,
        regiao,
        sexo,
        COUNT(desligamento) as qnt_desligamento
    FROM
        `VW_EMPREGADO_FULL`
    GROUP BY
        ano,
        regiao,
```

```
    sexo
)
SELECT
  *,
  ROUND(
    qnt_desligamento - LAG(qnt_desligamento, 1) OVER (
      ORDER BY
        ano,
        regiao,
        sexo
    ),
    2
  ) as diff
FROM qnt_desligs as q
ORDER BY ano, regiao, sexo;
"""", cnx)
```

Out[]:

	ano	regiao	sexo	qnt_desligamento	diff
0	2018	Centro-Oeste	Feminino	7691	NaN
1	2018	Centro-Oeste	Masculino	27957	20266.0
2	2018	Nordeste	Feminino	7960	-19997.0
3	2018	Nordeste	Masculino	35725	27765.0
4	2018	Norte	Feminino	27609	-8116.0
5	2018	Norte	Masculino	31485	3876.0
6	2018	Sudeste	Feminino	45889	14404.0
7	2018	Sudeste	Masculino	210075	164186.0
8	2018	Sul	Feminino	12509	-197566.0
9	2018	Sul	Masculino	59952	47443.0
10	2019	Centro-Oeste	Feminino	3319	-56633.0
11	2019	Centro-Oeste	Masculino	17361	14042.0
12	2019	Nordeste	Feminino	1931	-15430.0
13	2019	Nordeste	Masculino	10631	8700.0
14	2019	Norte	Feminino	832	-9799.0
15	2019	Norte	Masculino	1800	968.0
16	2019	Sudeste	Feminino	15945	14145.0
17	2019	Sudeste	Masculino	78191	62246.0
18	2019	Sul	Feminino	3861	-74330.0
19	2019	Sul	Masculino	28202	24341.0
20	2020	Centro-Oeste	Feminino	3162	-25040.0
21	2020	Centro-Oeste	Masculino	16459	13297.0
22	2020	Nordeste	Feminino	4079	-12380.0
23	2020	Nordeste	Masculino	18136	14057.0
24	2020	Norte	Feminino	4109	-14027.0
25	2020	Norte	Masculino	4252	143.0
26	2020	Sudeste	Feminino	25288	21036.0
27	2020	Sudeste	Masculino	107055	81767.0
28	2020	Sul	Feminino	8019	-99036.0
29	2020	Sul	Masculino	31545	23526.0
30	2021	Centro-Oeste	Feminino	8479	-23066.0
31	2021	Centro-Oeste	Masculino	40152	31673.0
32	2021	Nordeste	Feminino	11871	-28281.0
33	2021	Nordeste	Masculino	46354	34483.0
34	2021	Norte	Feminino	2284	-44070.0
35	2021	Norte	Masculino	12591	10307.0
36	2021	Sudeste	Feminino	98951	86360.0

ano	regiao	sexo	qnt_desligamento	diff
37	2021	Sudeste	Masculino	348584
38	2021	Sul	Feminino	25599
39	2021	Sul	Masculino	97117

8.5 Quantidades de demissões com dados acumulados por ano

```
WITH qnt_desligs AS (
    SELECT
        ano,
        sexo,
        COUNT(desligamento) as qnt_desligamento
    FROM
        `VW_EMPREGADO_FULL`
    GROUP BY ano, sexo
)
SELECT
    *,
    ROUND(
        qnt_desligamento + LAG(qnt_desligamento, 1) OVER (
            ORDER BY
                ano,
                sexo
        ),
        2
    ) as cum
FROM qnt_desligs as q
ORDER BY ano, sexo;
```

```
In [ ]: pd.read_sql(f"""
WITH qnt_desligs AS (
    SELECT
        ano,
        sexo,
        COUNT(desligamento) as qnt_desligamento
    FROM
        `VW_EMPREGADO_FULL`
    GROUP BY ano, sexo
)
SELECT
    *,
    ROUND(
        qnt_desligamento + LAG(qnt_desligamento, 1) OVER (
            ORDER BY
                ano,
                sexo
        ),
        2
    ) as cum
FROM qnt_desligs as q
ORDER BY ano, sexo;
""", cnx)
```

Out[]:

	ano	sexo	qnt_desligamento	cum
0	2018	Feminino	101658	NaN
1	2018	Masculino	365194	466852.0
2	2019	Feminino	25888	391082.0
3	2019	Masculino	136185	162073.0
4	2020	Feminino	44657	180842.0
5	2020	Masculino	177447	222104.0
6	2021	Feminino	147184	324631.0
7	2021	Masculino	544798	691982.0