Relatório DataWarehouse

Samuel Sousa 117040305

November 2021

Questão 1

Nessa questão, elaborei um script de download usando a biblioteca de requests do Python 3. A imagem abaixo representa o código que foi utilizado. Para agilizar o processo, paralelizei as três fontes de microdados, um em cada thread. Ao fazer o download, é baixado um arquivo .zip que, também utilizando a biblioteca do Python 3, foi descompacto gerando 3 pastas contendo todos os dados e suas respectivas bibliotecas.

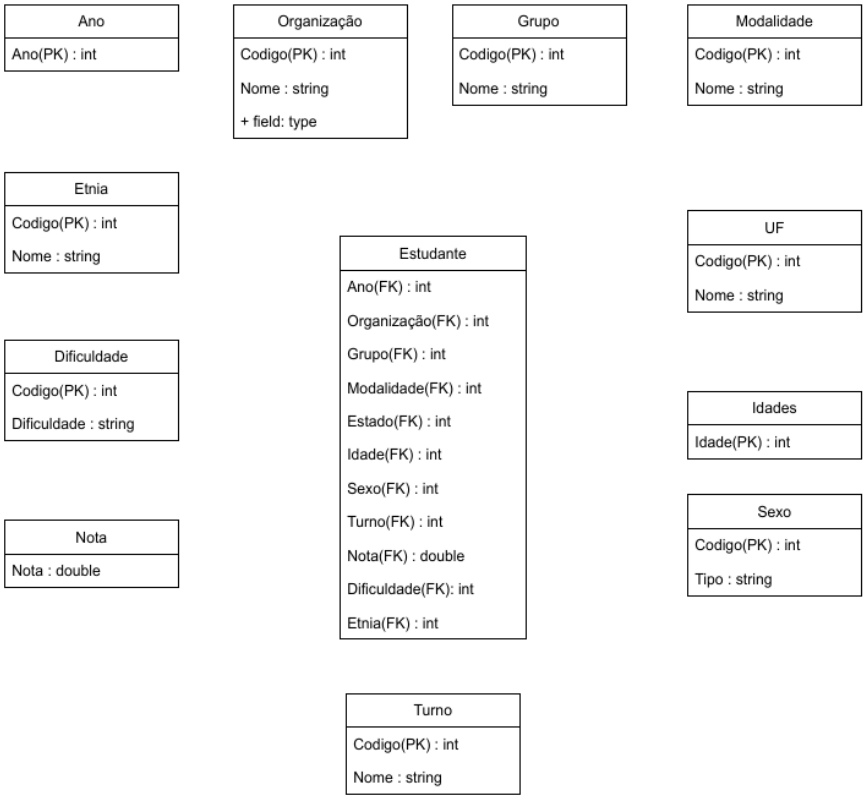
```python
def download_data(link, zip_name, save_path):
    data = requests.get(link)

    open(zip_name, 'wb').write(data.content)

    with zipfile.ZipFile(zip_name, 'r') as zip_ref:
        zip_ref.extractall(save_path)

def download_files():

    enade_2019_link = 'https://download.inep.gov.br/microdados/Enade_Microdados/microdados_enade_2019.zip'
    enade_2018_link = 'https://download.inep.gov.br/microdados/Enade_Microdados/microdados_enade_2018.zip'
    enade_2017_link = 'https://download.inep.gov.br/microdados/Enade_Microdados/microdados_Enade_2017_portal_2018.10.09.zip'


    t1 = threading.Thread(target=download_data, args=(enade_2019_link, 'enade_2019_zip', './enade_2019/'))
    t2 = threading.Thread(target=download_data, args=(enade_2018_link, 'enade_2018_zip', './enade_2018/'))
    t3 = threading.Thread(target=download_data, args=(enade_2017_link, 'enade_2017_zip', './enade_2017/'))

    t1.start()
    t2.start()
    t3.start()

    t1.join()
    t2.join()
    t3.join()

    print('Download de arquivos concluído.')
```

Logo após o download e descompactação, gerei um script para pré-processar os dados(imagem abaixo), eliminando as colunas que não serão utilizadas no modelo dimensional, assim poupando memória e agilizando o processo ao abrir e ler os arquivos csv(Esse processo ocorre só uma vez).

```python
def preprocess_data():

    columns = [
        'NU_ANO','CO_ORGACAD','CO_GRUPO',
        'CO_MODALIDADE', 'CO_UF_CURSO',
        'NU_IDADE', 'TP_SEXO',
        'CO_TURNO_GRADUACAO', 'NT_GER',
        'CO_RS_I1', 'QE_I02'
    ]

    data_2017 = pd.read_csv('./enade_2017/3.DADOS/MICRODADOS_ENADE_2017.txt',
        skipinitialspace=True, sep=';', low_memory=False)
    data_2018 = pd.read_csv('./enade_2018/2018/3.DADOS/microdados_enade_2018.txt',
        skipinitialspace=True, sep=';', low_memory=False)
    data_2019 = pd.read_csv('./enade_2019/3.DADOS/microdados_enade_2019.txt',
        skipinitialspace=True, sep=';', low_memory=False)

    data_2017 = data_2017[columns]
    data_2018 = data_2018[columns]
    data_2019 = data_2019[columns]

    data_2017.to_csv('enade_2017_preprocessed')
    data_2018.to_csv('enade_2018_preprocessed')
    data_2019.to_csv('enade_2019_preprocessed')
```

Questão 2

Utilizei o site https://app.diagrams.net/ para fazer o modelo estrela abaixo.



Questão 3

O script SQL utilizado para a geração do banco está localizado no arquivo script.sql no github deste trabalho(Arquivo extenso para ser posto como imagem). Para salvar as chaves estrangeiras, como os códigos e nomes dos cursos, juntei os dados dos dicionários de dados dos 3 anos e fiz uma remoção de duplicatas no excel e também a substituição de '=' por ',', assim consegui gerar de forma automática todos os inserts iniciais.

Questão 4

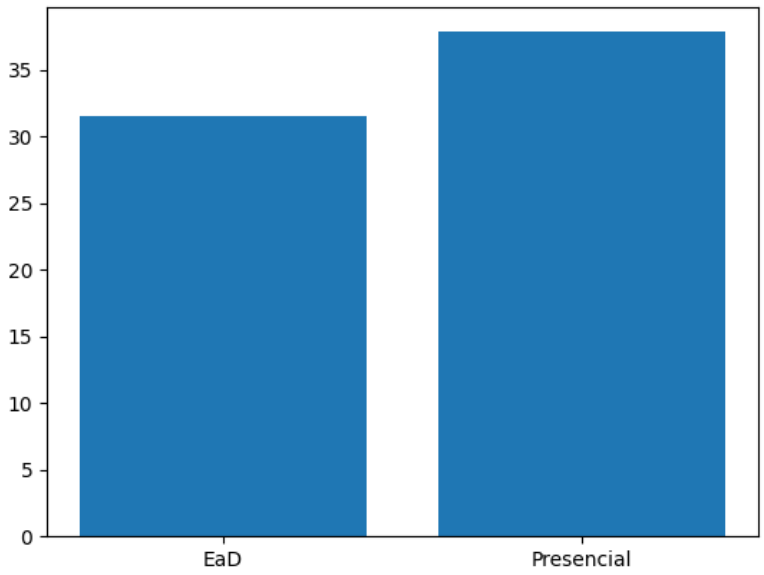
A carga de dados foi feita utilizando o seguinte código abaixo:

```
def read_insert_data():

    data_2017 = pd.read_csv('./enade_2017_preprocessed', skipinitialspace=True, sep=',', low_memory=False)
    data_2018 = pd.read_csv('./enade_2018_preprocessed', skipinitialspace=True, sep=',', low_memory=False)
    data_2019 = pd.read_csv('./enade_2019_preprocessed', skipinitialspace=True, sep=',', low_memory=False)

    sql_string = 'INSERT INTO Estudante(ano, organizacao, grupo, modalidade, uf, idade, sexo, turno, nota, dificuldade, etnia) VALUES '
    values = ''
    with open('./test', 'w') as file:
            for dataset in [data_2017, data_2018, data_2019 ]:
                for index, row in dataset.iterrows():
                    estudante = treat_data(row)
                    values += f'''  ({estudante['ano']},{estudante['organizacao']},{estudante['grupo']},
                                    {estudante['modalidade']},{estudante['uf']},{estudante['idade']},
                                    '{estudante['sexo']}',{estudante['turno']},{estudante['nota']},
                                    '{estudante['dificuldade']}', '{estudante['etnia']}' ),'''


    sql_string += values[:-1] + ';'
    con = sqlite3.connect('estudante.db')
    cursor = con.cursor()
    cursor.execute(sql_string)

    con.commit()
    con.close()
```

A ideia por trás foi ler dos arquivos pré-processados(sem as colunas que não foram utilizadas no modelo dimensional), e gerar uma string de inserção SQL. Cada linha do arquivo é tratada numa função treat_data que remove campos nulos e não identificados, substituindo por zeros, informando ao banco que aquele dado não estava disponível. Uma vez concluída a string(cerca de 1milhão de dados) a carga é feita.

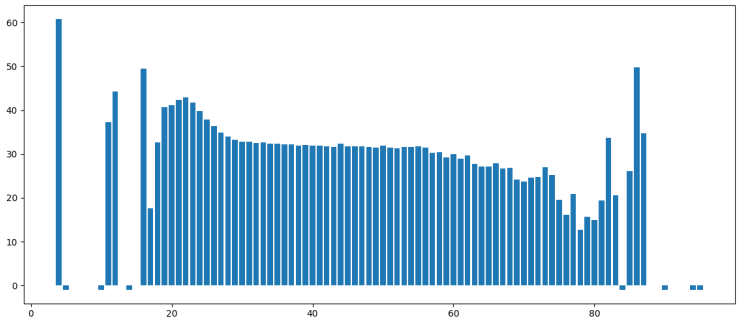
Questão 5

A primeira questão levantada foi a nota média por modalidade, isto é, presencial ou EaD.



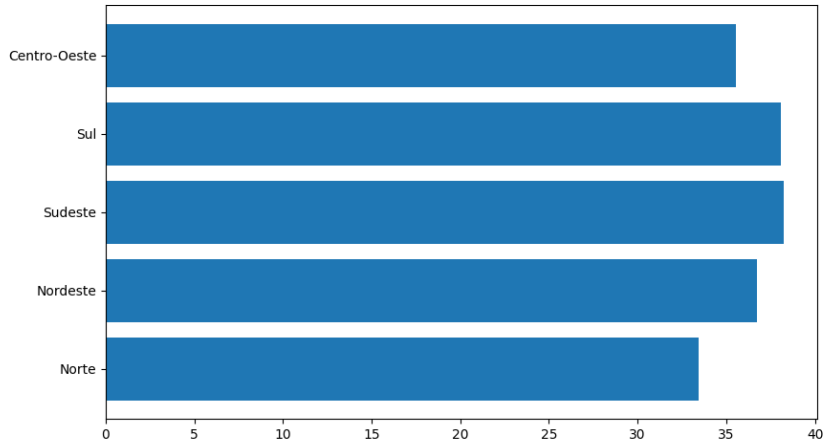
Podemos notar que pela imagem acima, o curso presencial teve uma nota média maior. Diversos fatores podem explicar este fato, dentre eles : Os cursos presenciais apresentam uma carga de estudo maior, estudantes dos cursos presenciais são mais jovens e não precisam arcar com trabalhos em tempo integral, o regime EaD costuma ser mais enxuto em certo assuntos, dentre outros.

A segunda questão é a nota média por idade.



Notamos acima que as maiores médias estão entre os estudantes mais jovens, o que também se interliga ao fato da primeira questão.

A terceira questão diz respeito a nota média por região brasileira.



Podemos observar que as regiões sul e sudeste concentram as maiores médias. Além de serem regiões populosas, elas também concentram um nível de renda, acesso a recursos entre outros fatores socio-econômicos acima das demais. O que ilustra uma deficiência que precisa ser tratada nas outras regiões.

A quarta questão responde qual a dificuldade percebida por cada etnia. Não consegui elaborar um gráfico, mas foi elaborado a tabela abaixo:

	Branca	Preta	Amarela	Parda	Indígena	Não declarado	Dado faltante
0	285	67	7	195	3	24	15
1	22879	4080	1140	14465	189	1095	85514
2	39906	8588	1998	29638	402	2287	63450
3	10424	2049	430	5928	84	746	266
4	58068	7553	2392	26055	277	2865	735
5	378158	62006	17327	232415	2339	15255	5241
6	184590	31011	8264	119769	1257	6313	2729
7	29386	5673	1363	20352	199	1243	504

Onde os índices a esquerda representam, respectivamente, Muito fácil, Fácil, Médio, Difícil, Muito difícil, Resposta anulada, Sem resposta, Dado faltante

A quinta e ultima questão responde os turnos por etnia dos participantes, essa questão busca esclarecer algumas condições socioeconômicas entre aqueles que trabalham e estudam e os que só estudam.

	Branca	Preta	Amarela	Parda	Indígena	Não declarado	Dado faltante
0	9719	2238	478	7873	97	463	3348
1	83572	15392	4004	52916	569	4290	19256
2	15016	4395	1004	15392	221	1073	5262
3	238134	35561	11044	142348	1462	10191	36917
4	377255	63441	16391	230288	2401	13811	93671

Questão 6

Na tentativa do aprendizado, utilizei as variáveis de estado, curso e idade para poder prever a nota do candidato. No entanto não tive um resultado satisfatório, tendo um score de regressão linear muito baixo. Acredito que os tratamentos de dados e as repetições de alguns valores puderam influenciar nesse resultado. Abaixo está o trecho do código utilizado no aprendizado:

```
def machine_learn():
    con = sqlite3.connect('estudante.db')
    cursor = con.cursor()

    df = pd.DataFrame([row for row in cursor.execute('SELECT * FROM Estudante') if row[8] != -1])
    X = df[[4, 2, 5]]
    y = df[[8]]

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=2)

    LR = LinearRegression()
    result  = LR.fit(X_train, y_train)

    print(result.score(X_test,y_test))
```

Questão 7

As ferramentas utilizadas foram :

1. Python : https://www.python.org/

   O uso do python foi escolhido pela gama de ferramentas da linguagem, documentação e o meu conhecimento a respeito.

2. SQLite3 : https://www.sqlite.org/index.html

   O SQLite é um banco de dados simples e local, reduzindo parte dos problemas de configurações iniciais de um BD.

3. Matplotlib.Pyplot : https://matplotlib.org/stable/tutorials/introductory/pyplot.html

   O pyplot foi escolhido por ser uma ferramenta simples de manipulação de gráficos.

4. Numpy : https://numpy.org/

   O numpy foi escolhido pois este interage bem com os modelos de aprendizado de máquina e gráficos do pyplot.

5. Sklearn : https://scikit-learn.org/

   O Scikit Learn foi escolhido por ter vários modelos de aprendizado de máquina e funções já preparadas para o tipo de dado que estava sendo trabalhado, além de ter um tutorial de fácil uso.

6. Pandas : https://pandas.pydata.org/

   O Pandas foi escolhido por ser a melhor ferramenta para tratar um grande volume de dados, além de possuir ferramentas integradas com Numpy, Sklearn e Pyplot.