ST IT Cloud - Data and Analytics Test LV.4 ¶

Esse teste deve avaliar alguns conceitos de big data e a qualidade técnica na manipulacão de dados, otimização de performance, trabalho com arquivos grandes e tratamento de qualidade.

Passo a passo

- Parte teórica: responda as questões abaixo preenchendo as células em branco.
- Parte prática: disponibilizamos aqui 2 cases para, leia os enunciados dos problemas, desenvolver os programas, utilizando a stack definida durante o processo seletivo, para entregar os dados de acordo com os requisitos descritos abaixo.

Faz parte dos critérios de avaliacão a pontualidade da entrega. Implemente até onde for possível dentro do prazo acordado.

Os dados de pessoas foram gerados de forma aleatória, utilizando a biblioteca FakerJS, FakerJS-BR e Faker

LEMBRE-SE: A entrega deve conter TODOS os passos para o avaliador executar o programa (keep it simple).

Questão 1 - Descreva de forma detalhada quais são as etapas na construção de um pipeline de dados, sem considerar ferramentas específicas, imagine que é seu primeiro contato com o cliente e você precisa entender a demanda dele e explicar quais são os passos que você terá que implementar para entregar a demanda.

Deve-se realizar verificar com o o usuario do sistema qual o objetivo do cliente (quais dados o cliente quer apresentado). Logo apos verificar a fonte dos dados, bem como se a coleta sera Stream ou Batch. Logo então realiza-se uma analise de requisitos (ferramentas, custos, mão de obra) e um planejamento pre-eliminar e os apresenta ao cliente. Se o cliente achar o investimento e o planejamento condizente avançamos para o desenvolvimento.

Questão 2 - Defina com suas palavras um processamento em streaming e processamento em batch. Qual sua experiência com cada uma delas.

Bathc: Os dados são carregados obedecendo periodos de latencia (hora em hora, dias em dias e etc). Já desenvolvi um sistema de sensoriamento desde a projeção eletrica dos sensores até a arquitetura e envio dos dados, durante a estadia no IST, em que dados de um modulo de sensor era enviados usando um protocolo de alta latencia (MQTT). O servidor armazenava os dados em uma base de dados relacional, e um job capturava os dados diariamente e realizava o processo de enviar para a area de stagging em um servidor no EC2 onde era aplicado o ETL aos dados.

Stremming: Os dados são enviados em tempo real para o processo de ETL. Meu trabalho de conclusão de curso foi um multimedidor trifasico que realiza a medição da rede trifasica ou monofasica e envia os dados via MQTT para um servidor pessoal, onde usando um processo incremental de ETL, trata os dados e os apresenta em um dashboard. Alem disso na Cogna e na Data Strategy trabalho com preocesso straming de tratamento de dados, extraindo os dados do SAP e os enviando para uma arquitetura desenvolvida na AWS.

Questão 3 - Quais são as camadas de um Data Lake?

Um data lake geralmente é composto por tres camadas que ganham diferentes nomeclaturas de acordo com

seus gestores.

1 - Transient , stagging , landing: É a camada onde os dados brutos aterrissam com pouco ou quase nenhum tratamento

2 - Raw: Os dados são tratados e homogenizados.

3 - Integration ou trusted (zona confiavel): Aqui os dados já foram tratados e homogenizados agora a a necessidade de validar a qualidade dos dados.

Questão 4 - Quais as diferenças de um Data Lake e um DW?

Data Lake é o repositorio de dados massissos sem tratativas, que apos passarem pelo processo de ETL é convertido em um Data Warehouse (DW). A diferença se da tanto na modelagem dos dados onde um DW é apresentado em fatos e dimensoes enquando um data lake pode ser apresentado de n formas, quando nos dados, no qual em um data lake não ha necessidade de uma homogenização dos dados, enquanto no DW os dados preferencialmente devem ser homogenizados.

Questão 5 - O que é arquitetura Lambda e Kappa? Descreva com suas palavras.

São arquiteturas que apresentam tratativas de processamento para dados de alta e baixa prioridade de processamento. Enquanto a Lambda propoem dois caminhos distintos para o processamento, com requesitos de latencia diferentes, o Kappa propoe que os dados passem pelo mesmo caminho no entanto obedecendo os criterios de prioridade.

Questão 6 - O que é Data Quality para você e como você implementa isso nos seus processos?

Data Quality como o proprio nome infere é a validação da qualidade e da confiabilidade dos dados os quais estou tratando. Sempre procuro analizar os dados a fim de verificar se não ha anomalias. Uma boa tecnica é verificar o desvio padrão de amostras de dados numericos.

Questão 7 - Em uma escala de 0 a 10, qual seria seu nível de experiência com PySpark?

Considero que tenho muito mais experiencia utilizando Pandas que PySpark, porem tenho um amplo conhecimento, já desenvolvi pipelines que ainda rodam em produção durante a minha estadia no IST bem como na DataStrategy e na Cogna trabalho conrinuamente com PySpark. Considero em uma escala de 0 a 10 que tenho 6 de experiencia com PySpark

Questão 8 - Em uma escala de 0 a 10, qual seria seu nível de experiência com SQL?

De 0 a 10 considero que tenho 7 de experiencia com SQL.

Questão 9 - Descreva suas expeciências com banco de dados SQL e NoSQL.

SQL: Já trabalhei e desenvolvi projetos com Oracle (PL/SQL) , SQL Server (Transact), Postesgress, Firebird, Sqlite, MySql. Tenho experiencia desde 2014 com SQL.

NoSQL: Já desenvolvi projetos com Mongo DB, outras tecnologias como Cassandra tive contato apenas acabemico.

Em todas as tecnologias utilizei juntamente linguagens como Java, React Native, Python e C++.

Questão 10 - Tem experiência com versionamento de código? Com quais ferramentas já trabalhou? Descreva.

Sim, já trabalhei com github, bitbucket e azure devops, todas usando git, tambem já trabalhei com versionamento e manutenção de servidores privados porem open source como o gitblit.

Questão 11 - Tem experiência em desenvolvimento em cloud? Se sim, especifique a(s) plataforma(s) que já trabalhou e suas principais implementações e conhecimentos em cada serviço.

Sim, já trabalhei com AWS durante a minha estadia no IST, na DataStrategy e na Cogna (ec2,Elastic Beanstalk, s3, athena, cloud9, EMR, Lambda, StepFuntions'). E já trabalhei com cloud em um serviço proprietario da Totvs (Cofre) famoso por hospedar clientes como Boticario.

Questão 12 - Tem experiência com metodologia ágil? Qual?

Sim, já trabalhei com o Scrum e Kanbam na Totvs, bem como na DataStrategy. Geralmente aplico uma metodologia mista de scrum e kanbam em minhas atividades pessoais. Tenho experiencia com diversas ferramentas como o Trail e Jira, bem como ferramentas mais simples como Planner e Trello.

TESTE PRÁTICO

Problema 1: Você está recebendo o arquivo 'dados_cadastrais_fake.csv' que contem dados cadastrais de clientes, mas para que análises ou relatórios sejam feitos é necessário limpar e normalizar os dados. Além disso, existe uma coluna com o número de cpf e outra com cnpj, você precisará padronizar deixando apenas dígitos em formato string (sem caracteres especiais), implementar uma forma de verificar se tais documentos são válidos sendo que a informação deve se adicionada ao dataframe em outras duas novas colunas.

Após a normalização, gere reports que respondam as seguintes perguntas:

- Quantos clientes temos nessa base?
- Qual a média de idade dos clientes?
- Quantos clientes nessa base pertencem a cada estado?
- Quantos CPFs válidos e inválidos foram encontrados?
- Quantos CNPJs válidos e inválidos foram encontrados?

Ao final gere um arquivo no formato csv e um outro arquivo no formato parquet chamado (problema1_normalizado), eles serão destinados para pessoas distintas.

EXTRA: executar as mesmas validações no *1E8.csv.gz

In [17]:

```
! pip install tqdm #Util para apresentar a barra de progresso
! pip install numpy #Util para gerar vetores
! pip install pandas # Utilizada no tratamento dos dados
! pip install unicodedata #Utilizada para homogenizar dados com codificação Unicode
! pip install pyarrow #Necessaria para o pandas gerar .parquet
! pip install fastparquet #Necessaria para o pandas gerar .parquet
! pip install pyunpack #Necessaria para extrair arquivos 7z
! pip install patool #Necessaria para extrair arquivos 7z
```

```
Requirement already satisfied: tqdm in c:\users\lucas.rsoares\appdata\local
\programs\python\python39\lib\site-packages (4.62.2)
Requirement already satisfied: colorama in c:\users\lucas.rsoares\appdata\lo
cal\programs\python\python39\lib\site-packages (from tqdm) (0.4.4)

WARNING: You are using pip version 21.1.3; however, version 21.2.4 is availa
ble.
You should consider upgrading via the 'c:\users\lucas.rsoares\appdata\local
\programs\python\python39\python.exe -m pip install --upgrade pip' command.

Requirement already satisfied: numpy in c:\users\lucas.rsoares\appdata\local
\programs\python\python39\lib\site-packages (1.21.0)

WARNING: You are using pip version 21.1.3; however, version 21.2.4 is availa
ble.
You should consider upgrading via the 'c:\users\lucas.rsoares\appdata\local
\programs\python\python39\python.exe -m pip install --upgrade pip' command.

Requirement already satisfied: pandas in c:\users\lucas.rsoares\appdata\loca
l\programs\python\python39\lib\site-packages (1.3.0)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\lucas.rsoa
res\appdata\local\programs\python\python39\lib\site-packages (from pandas)
(2.8.2)
Requirement already satisfied: pytz>=2017.3 in c:\users\lucas.rsoares\appdat
a\local\programs\python\python39\lib\site-packages (from pandas) (2021.1)
Requirement already satisfied: numpy>=1.17.3 in c:\users\lucas.rsoares\appda
ta\local\programs\python\python39\lib\site-packages (from pandas) (1.21.0)
Requirement already satisfied: six>=1.5 in c:\users\lucas.rsoares\appdata\lo
cal\programs\python\python39\lib\site-packages (from python-dateutil>=2.7.3-
>pandas) (1.16.0)

WARNING: You are using pip version 21.1.3; however, version 21.2.4 is availa
ble.
You should consider upgrading via the 'c:\users\lucas.rsoares\appdata\local
\programs\python\python39\python.exe -m pip install --upgrade pip' command.
ERROR: Could not find a version that satisfies the requirement unicodedata
(from versions: none)
ERROR: No matching distribution found for unicodedata
WARNING: You are using pip version 21.1.3; however, version 21.2.4 is availa
ble.
You should consider upgrading via the 'c:\users\lucas.rsoares\appdata\local
\programs\python\python39\python.exe -m pip install --upgrade pip' command.


Requirement already satisfied: pyarrow in c:\users\lucas.rsoares\appdata\loc
al\programs\python\python39\lib\site-packages (5.0.0)
Requirement already satisfied: numpy>=1.16.6 in c:\users\lucas.rsoares\appda
ta\local\programs\python\python39\lib\site-packages (from pyarrow) (1.21.0)
```

```
WARNING: You are using pip version 21.1.3; however, version 21.2.4 is avai
lable.
You should consider upgrading via the 'c:\users\lucas.rsoares\appdata\loca
l\programs\python\python39\python.exe -m pip install --upgrade pip' comman
d.
WARNING: You are using pip version 21.1.3; however, version 21.2.4 is avai
lable.
You should consider upgrading via the 'c:\users\lucas.rsoares\appdata\loca
l\programs\python\python39\python.exe -m pip install --upgrade pip' comman
d.

Requirement already satisfied: fastparquet in c:\users\lucas.rsoares\appdata
\local\programs\python\python39\lib\site-packages (0.7.1)
Requirement already satisfied: pandas>=1.1.0 in c:\users\lucas.rsoares\appda
ta\local\programs\python\python39\lib\site-packages (from fastparquet) (1.3.
0)
Requirement already satisfied: thrift>=0.11.0 in c:\users\lucas.rsoares\appd
ata\local\programs\python\python39\lib\site-packages (from fastparquet) (0.1
3.0)
Requirement already satisfied: cramjam>=2.3.0 in c:\users\lucas.rsoares\appd
ata\local\programs\python\python39\lib\site-packages (from fastparquet) (2.
3.2)
Requirement already satisfied: fsspec in c:\users\lucas.rsoares\appdata\loca
l\programs\python\python39\lib\site-packages (from fastparquet) (2021.8.1)
Requirement already satisfied: numpy>=1.18 in c:\users\lucas.rsoares\appdata
\local\programs\python\python39\lib\site-packages (from fastparquet) (1.21.
0)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\lucas.rsoa
res\appdata\local\programs\python\python39\lib\site-packages (from pandas>=
1.1.0->fastparquet) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in c:\users\lucas.rsoares\appdat
a\local\programs\python\python39\lib\site-packages (from pandas>=1.1.0->fast
parquet) (2021.1)
Requirement already satisfied: six>=1.5 in c:\users\lucas.rsoares\appdata\lo
cal\programs\python\python39\lib\site-packages (from python-dateutil>=2.7.3-
>pandas>=1.1.0->fastparquet) (1.16.0)

ERROR: Could not find a version that satisfies the requirement zipfile (from
versions: none)
ERROR: No matching distribution found for zipfile
WARNING: You are using pip version 21.1.3; however, version 21.2.4 is availa
ble.
You should consider upgrading via the 'c:\users\lucas.rsoares\appdata\local
\programs\python\python39\python.exe -m pip install --upgrade pip' command.

Requirement already satisfied: pyunpack in c:\users\lucas.rsoares\appdata\lo
cal\programs\python\python39\lib\site-packages (0.2.2)
Requirement already satisfied: easyprocess in c:\users\lucas.rsoares\appdata
\local\programs\python\python39\lib\site-packages (from pyunpack) (0.3)
Requirement already satisfied: entrypoint2 in c:\users\lucas.rsoares\appdata
\local\programs\python\python39\lib\site-packages (from pyunpack) (0.2.4)

WARNING: You are using pip version 21.1.3; however, version 21.2.4 is availa
ble.
You should consider upgrading via the 'c:\users\lucas.rsoares\appdata\local
\programs\python\python39\python.exe -m pip install --upgrade pip' command.

Collecting patool
  Downloading patool-1.12-py2.py3-none-any.whl (77 kB)
Installing collected packages: patool
Successfully installed patool-1.12
```

```
WARNING: You are using pip version 21.1.3; however, version 21.2.4 is avai
lable.
You should consider upgrading via the 'c:\users\lucas.rsoares\appdata\loca
l\programs\python\python39\python.exe -m pip install --upgrade pip' comman
d.
```

In [1]:

```
import pandas as pd
import re #utilizadas para trabalhar com expressoes regulars
import unicodedata
from tqdm import tqdm
import requests  #utilizada para realizar o dowload do arquivo do problema 2
import os #Necessaria para criar pastas e diretorios
from datetime import datetime #Necessaria para homogenizar as datas
import numpy as np
from pyunpack import Archive
```

In [2]:

```
def verificaDocumento(numero):
  if len(numero)==11:

    cpf = ''.join(re.findall('\d', str(numero)))

    if (not cpf) or (len(cpf) < 11):
        return False
    inteiros = list(map(int, cpf))
    novo = inteiros[:9]
    while len(novo) < 11:
        r = sum([(len(novo)+1-i)*v for i,v in enumerate(novo)]) % 11
        if r > 1:
            f = 11 - r
        else:
            f = 0
        novo.append(f)
    if novo == inteiros:
        return True
  elif len(numero)==14:
    cnpj = ''.join(re.findall('\d', str(numero)))
    if (not cnpj) or (len(cnpj) < 14):
        return False
    inteiros = list(map(int, cnpj))
    novo = inteiros[:12]

    prod = [5, 4, 3, 2, 9, 8, 7, 6, 5, 4, 3, 2]
    while len(novo) < 14:
        r = sum([x*y for (x, y) in zip(novo, prod)]) % 11
        if r > 1:
            f = 11 - r
        else:
            f = 0
        novo.append(f)
        prod.insert(0, 6)
    if novo == inteiros:
        return True
  return False
```

In [3]:

```
def normatizaUF(uf):
  if len(uf)>2:
    uf = ''.join(caracter for caracter in unicodedata.normalize('NFKD', uf)
    if not unicodedata.combining(caracter)).replace(' ','').upper()

    listReplace = [['AC','ACRE'], ['AL','ALAGOAS'], ['AP','AMAPA'], ['AM','AMAZONAS'],
    ['BA','BAHIA'], ['CE','CEARA'],['ES','ESPIRITOSANTO'], ['GO','GOIAS'], ['MA','MARANHAO'
    ['MT','MATOGROSSO'], ['MS','MATOGROSSODOSUL'], ['MG','MINASGERAIS'],['PA','PARA'],
    ['PB','PARAIBA'], ['PR','PARANA'], ['PE','PERNAMBUCO'],['PI','PIAUI'],['RJ','RIODEJANEI
    ['RN','RIOGRANDEDONORTE'], ['RS','RIOGRANDEDOSUL'],['RO','RONDONIA'], ['RR','RORAIMA'],
    ['SC','SANTACATARINA'], ['SP','SAOPAULO'],['SE','SERGIPE'], ['TO','TOCANTINS'],
    ['DF','DISTRITOFEDERAL']]

    for sigla,estado in listReplace:
      if uf in estado:
        return sigla
    print(uf)
  else:
    return uf.upper()
```

In [4]:

```
def homogenizaDocumento(numDoc):
  if len(numDoc) == 11:
    return '{}{}{}.{}{}{}.{}{}{}-{}{}'.format(*numDoc)
  elif len(numDoc) == 14:
    return '{}{}.{}{}{}.{}{}{}/{}{}{}{}-{}{}'.format(*numDoc)
  else:
    print(numdoc)
    return numdoc
```

In [6]:

```
dados = pd.read_csv("dados_cadastrais_fake.csv",sep = ';')
renameDict = {'nomes': 'NOM_CLIENTE', 'idade': 'VLR_IDADE',
              'cidade': 'NOM_CIDADE', 'estado': 'SGL_UF',
              'cpf':"NUM_CPF",'cnpj':"NUM_CNPJ"}
dados.rename(renameDict, axis=1, inplace=True)
array = np.arange(len(dados))
dados['ID'] = array
dados = dados.set_index('ID')
dadosdict = dados.to_dict('records')
for row in tqdm(dadosdict):
    cpf = row['NUM_CPF'].replace('.','').replace('-','').replace('/','').replace('\\','')
    cnpj = row['NUM_CNPJ'].replace('.','').replace('-','').replace('/','').replace('\\','')
    row['NUM_CPF'] = homogenizaDocumento(cpf)
    row['FLAG_CPF_VALIDO'] = verificaDocumento(cpf)
    row['NUM_CNPJ'] = homogenizaDocumento(cnpj)
    row['FLAG_CNPJ_VALIDO'] = verificaDocumento(cnpj)
    row['SGL_UF'] = normatizaUF(row['SGL_UF'])

dados = pd.DataFrame(dadosdict)
```

```
100%|
| 10000/10000 [00:00<00:00, 52089.89it/s]
```

1: Report da analise do dataframe

In [7]:

```
countClientes = dados[dados.columns[0]].count()
avgIdade = dados["VLR_IDADE"].mean()
cpfsValidos = dados[dados["FLAG_CPF_VALIDO"]==True].count()["FLAG_CPF_VALIDO"]
cnpjsValidos = dados[dados["FLAG_CNPJ_VALIDO"]==True].count()["FLAG_CNPJ_VALIDO"]
report = {"Quantidade de Clientes": [int(countClientes)],
          "Media de idade dos clientes": ['%.2f' % float(avgIdade)],
          "CPF's validos": [int(cpfsValidos)] ,
          "CPF's invalidos":[int(countClientes-cpfsValidos)],
          "CNPJ's validos": [int(cnpjsValidos)],
          "CNPJ's invalidos": [int(countClientes-cnpjsValidos)]
          }
dfReport = pd.DataFrame(report).T
display(dfReport)
```

	0
Quantidade de Clientes	10000
Media de idade dos clientes	53.78
CPF's validos	10000
CPF's invalidos	0
CNPJ's validos	10000
CNPJ's invalidos	0

2: Clientes por estado

In [8]:

```
clientesUf = dados.groupby(['SGL_UF']).size().reset_index(name='N° Clientes')
display(clientesUf)
```

	SGL_UF	N° Clientes
0	AC	371
1	AL	371
2	AM	371
3	AP	371
4	BA	371
5	CE	371
6	DF	371
7	ES	371
8	GO	371
9	MA	371
10	MG	370
11	MS	370
12	MT	370
13	PA	370
14	PB	370
15	PE	370
16	PI	370
17	PR	370
18	RJ	370
19	RN	370
20	RO	370
21	RR	370
22	RS	370
23	SC	370
24	SE	370
25	SP	370
26	TO	370

In [9]:

```
path_Inicial = 'Problema1'
if not os.path.isdir(path_Inicial):
    os.makedirs(path_Inicial)
dados.to_csv(path_Inicial+'/problema1_normalizado.csv', sep=';', encoding='utf-8')
dados.to_parquet(path_Inicial+'/problema1_normalizado.parquet')
```

Problema 2: Você deverá implementar um programa, para ler, tratar e particionar os dados.

O arquivo fonte está disponível em `https://st-it-cloud-public.s3.amazonaws.com/people-v2_1E6.csv.gz`

Data Quality

- Higienizar e homogenizar o formato da coluna `document`
- Detectar através da coluna `document` se o registro é de uma Pessoa Física ou Pessoa Jurídica, adicionando uma coluna com essa informação
- Higienizar e homogenizar o formato da coluna `birthDate`
- Existem duas colunas nesse dataset que em alguns registros estão trocadas. Quais são essas colunas?

  R: Haviam registros em que o JobType estava no campo telefone e o Telefone estava no campo JobType, foi corrigido na etapa de homogenização dos dados.
- Corrigir os dados com as colunas trocadas
- Além desses pontos, existem outras tratamentos para homogenizar esse dataset. Aplique todos que conseguir.

Agregação dos dados

- Quais são as 5 PF que mais gastaram (`totalSpent`)?
- Qual é o valor de gasto médio por estado (`state`)?
- Qual é o valor de gasto médio por `jobArea` ?
- Qual é a PF que gastou menos (`totalSpent`)?
- Quantos nomes e documentos repetidos existem nesse dataset?
- Quantas linhas existem nesse dataset?

Particionamento de dados tratados com as regras descritas em **`DATA QUALITY`**

- Particionar em arquivos PARQUET por estado (`state`)
- Particionar em arquivos CSV por ano/mes/dia de nascimento (`birthDate`)

Data Quality

In [10]:

```
#dowload arquivo
response = requests.get("https://st-it-cloud-public.s3.amazonaws.com/people-v2_1E6.csv.gz",

with open("people-v2_1E6.csv.gz", "wb") as handle:
    for data in tqdm(response.iter_content()):
        handle.write(data)
```

```
49918255it [05:17, 157043.20it/s]
```

In [12]:

```
#descompactar arquivo
path_Inicial = 'Problema2'
if not os.path.isdir(path_Inicial):
    os.makedirs(path_Inicial)

Archive('people-v2_1E6.csv.gz').extractall("Problema2")
```

In [5]:

```
#importar arquivo
dados = pd.read_csv("Problema2/people-v2_1E6.csv",sep = ';')
array = np.arange(len(dados))
renameDict = {'document': 'NUM_DOCUMENTO', 'name': 'NOM_CLIENTE',
              'job':'NOM_CARGO','jobArea':'NOM_AREA_TRABALHO',
              'jobType':'NOM_TIPO_TRABALHO','phoneNumber':'NUM_TELEFONE',
              'birthDate':"DATA_ANIVERSARIO",
              'city': 'NOM_CIDADE', 'state': 'SGL_UF',
              'totalSpent':'VLR_TOTAL_GASTO'}
dados.rename(renameDict, axis=1, inplace=True)
dados['ID'] = array
dados = dados.set_index('ID')
```

In [6]:

```
def tipoPessoa(numDoc):
  if len(numDoc) == 11:
    return 'PF'
  elif len(numDoc) == 14:
    return 'PJ'
  else:
    return '-'
```

In [7]:

```
def homogenizaData(stringData):
  stringData = stringData.replace('.','/').replace(',','/').replace('-','/').replace('\\','
  dataFormats = ['%d/%m/%Y','%m/%d/%Y','%d/%b/%Y','%b/%d/%Y','%a/%d/%m/%Y','%a/%m/%d/%Y','%
                 '%Y%m%d','%Y%m%d/']
  for format in dataFormats:
    try:
      data = (datetime.strptime(stringData,format )).strftime('%d/%m/%Y')
      return data
    except:
      continue
```

In [8]:

```
def homogenizaPhoneNumber(texto):
  fone = texto.replace('+','').replace('(','').replace(')','').replace('-','').replace(' ',
  if fone.isdigit():
    if len(fone) == 13:
      return '+{}{} ({}{}) {} {}{}{}{}-{}{}{}{}'.format(*fone)
    elif len(fone) == 12:
      return '+{}{} ({}{}) {}{}{}{}-{}{}{}{}'.format(*fone)
    elif len(fone) == 11:
      return '+55 ({}{}) {}{}{}{}-{}{}{}{}'.format(*fone)
    elif len(fone) == 10:
      return '+55 ({}{}) {}{}{}{}-{}{}{}{}'.format(*fone)
    else:
      print(fone)
  else:
    return ''
```

In [9]:

```
#aplica homogenização e correção nos dados
dadosdict = dados.to_dict('records')
for row in tqdm(dadosdict):
  documento = row['NUM_DOCUMENTO'].replace('.','').replace('-','').replace('/','').replace(
  telefone = row['NUM_TELEFONE'] if not pd.isna(row['NUM_TELEFONE']) else ''
  areaTrabalho = row['NOM_AREA_TRABALHO'] if not pd.isna(row['NOM_AREA_TRABALHO']) else ''
  uf = row['SGL_UF'] if not pd.isna(row['SGL_UF']) else ''
  dataAniversario = row['DATA_ANIVERSARIO'] if not pd.isna(row['DATA_ANIVERSARIO']) else ''

  row['NUM_DOCUMENTO'] = homogenizaDocumento(documento)
  row['FLAG_TIPO_PESSOA']= tipoPessoa(documento)
  row['FLAG_DOCUMENTO_VALIDO'] = verificaDocumento(documento)
  row['DATA_ANIVERSARIO']= homogenizaData(dataAniversario)
  row['SGL_UF']=normatizaUF(uf)
  if (homogenizaPhoneNumber(telefone) == "") and (not homogenizaPhoneNumber(areaTrabalho) =
    row['NUM_TELEFONE']=homogenizaPhoneNumber(areaTrabalho)
    row['NOM_AREA_TRABALHO']=telefone
  else:
    row['NUM_TELEFONE']=homogenizaPhoneNumber(telefone)

  if not homogenizaPhoneNumber(row['NOM_AREA_TRABALHO'])  == "":
    row['NOM_AREA_TRABALHO']=''

dataframe = pd.DataFrame(dadosdict)
```

```
100%|
1000000/1000000 [01:59<00:00, 8355.66it/s]
```

Agregação dos dados

1: 5 Pessoas Fisicas que mais gastaram

In [22]:

```
dataframe.loc[dataframe['FLAG_TIPO_PESSOA'] == 'PF'].nlargest(5,'VLR_TOTAL_GASTO')
```

Out[22]:

	NUM_DOCUMENTO	NOM_CLIENTE	NOM_CARGO	NOM_AREA_TRABALHO	NOM_TIPC
464764	187.416.888-13	Sra. Rocio Martins	Especialista Mobilidade Sênior	Configuração	
968481	821.908.647-55	Euvanderson Costa	Supervisor Branding Sênior	Branding	I
4807	589.531.476-70	Valeria Souza	Gerente Paradigma Chefe	Contas	
233482	171.950.127-00	Regina Melo	Especialista Identidade Frente	Mobilidade	I
280776	614.545.514-45	Wandir Martins	NaN	Otimização	

2: Gasto medio por estado

In [23]:

```
gastoUf = dataframe[['SGL_UF','VLR_TOTAL_GASTO']].groupby(['SGL_UF']).mean()
display(gastoUf)
```

SGL_UF	VLR_TOTAL_GASTO
AC	502.478424
AL	500.515519
AM	498.770663
AP	504.259137
BA	498.774140
CE	499.300300
DF	499.107366
ES	501.631747
GO	501.292493
MA	500.435299
MG	499.532094
MS	499.509741
MT	500.264610
PA	499.230847
PB	499.200399
PE	502.405773
PI	499.076521
PR	500.644200
RJ	502.438569
RN	499.756018
RO	498.237318
RR	500.956398
RS	500.630983
SC	501.345612
SE	498.240099
SP	498.694480
TO	501.901563

3: Gasto medio por area de trabalho

In [24]:

```
gastoJobArea = dataframe[['NOM_AREA_TRABALHO','VLR_TOTAL_GASTO']].groupby(['NOM_AREA_TRABAL
display(gastoJobArea)
```

NOM_AREA_TRABALHO	VLR_TOTAL_GASTO
	500.397910
A infraestrutura	499.722667
Aplicações	501.907173
Branding	499.845293
Comunicações	498.327090
Configuração	502.580277
Contas	499.084772
Criativo	499.669854
Dados	500.888762
Diretivas	499.936022
Divisão	501.003315
Fatores	498.326736
Funcionalidade	503.706068
Garantia	497.888706
Grupo	500.761461
Identidade	498.858893
Implementação	501.646314
Integração	503.211706
Interações	498.849805
Intranet	499.643178
Marca	500.513447
Marketing	499.712373
Mercados	495.362091
Mobilidade	501.566734
Métricas	499.061580
Operações	495.447901
Otimização	501.036381
Paradigma	502.820945
Pesquisa	498.784515
Prestação de contas	498.813095
Programa	502.797252
Qualidade	501.121881

NOM_AREA_TRABALHO	VLR_TOTAL_GASTO
Rede	500.888008
Resposta	499.565322
Segurança	503.086734
Soluções	501.481668
Táticas	498.320258
Usabilidade	506.531420

4: Pessoa Fisica que menos gastou (Não gastou nada)

In [25]:

```
dataframe.loc[dataframe['FLAG_TIPO_PESSOA'] == 'PF'].nsmallest(1,'VLR_TOTAL_GASTO')
```

Out[25]:

	NUM_DOCUMENTO	NOM_CLIENTE	NOM_CARGO	NOM_AREA_TRABALHO	NOM_TIPC
247394	588.689.262-10	Sr. Enio Souza	Consultor Marketing Distrito	Comunicações	

5.1: Documentos repetidos (Existem 430 repetiçoes de nomes)

In [26]:

```
reptidos = dataframe[['NUM_DOCUMENTO']].groupby(['NUM_DOCUMENTO']).size().reset_index(name=
reptidos = reptidos.loc[reptidos['REPETIÇOES'] >1 ]
print('Quantidade de Registros: ',len(reptidos))
display(reptidos)
```

```
Quantidade de Registros:  430
```

	NUM_DOCUMENTO	REPETIÇOES
9227	012.062.948-84	2
18294	024.417.647-72	2
18758	025.334.714-96	2
24382	031.248.722-31	2
26580	035.472.532-76	2
...	...	...
989752	973.832.252-95	2
991669	977.647.971-54	2
991945	978.201.713-24	2
993346	981.826.427-44	2
994849	984.715.476-78	2

430 rows × 2 columns

5.2: Nomes repetidos (Existem 173513 repetiçoes de nomes)

In [27]:

```
clireptidos = dataframe[['NOM_CLIENTE']].groupby(['NOM_CLIENTE']).size().reset_index(name='
clireptidos = clireptidos.loc[clireptidos['REPETIÇOES'] >1 ]
print('Quantidade de Registros: ',len(clireptidos))
display(clireptidos)
```

```
Quantidade de Registros:  173513
```

	NOM_CLIENTE	REPETIÇOES
0	Abdias	27
1	Abdias Albuquerque	4
2	Abdias Albuquerque Filho	2
3	Abdias Barros	4
5	Abdias Barros Jr.	2
...	...	...
455005	Érico Saraiva Neto	2
455007	Érico Silva	3
455009	Érico Silva Neto	3
455010	Érico Souza	4
455013	Érico Xavier	5

173513 rows × 2 columns

5.3: Nomes e documentos repetidos simultaneamente (Não existe ocorrencias de um mesmo nome e um mesmo documento se repetindo simultaneamente)

In [28]:

```
combrepetidos = dataframe[['NOM_CLIENTE','NUM_DOCUMENTO']].groupby(['NOM_CLIENTE','NUM_DOCU
combrepetidos = combrepetidos.loc[combrepetidos['REPETIÇOES'] >1 ]
print('Quantidade de Registros: ',len(combrepetidos))
display(combrepetidos)
```

```
Quantidade de Registros:  0
```

	NOM_CLIENTE	NUM_DOCUMENTO	REPETIÇOES

6: Quantidade de registros no dataframe

In [29]:

```
print('Quantidade de Registros: ',len(dataframe))
display(dataframe)
```

```
Quantidade de Registros:  1000000
```

	NUM_DOCUMENTO	NOM_CLIENTE	NOM_CARGO	NOM_AREA_TRABALHO	NOM_TIPO
0	766.841.487-87	Charlleny Braga	Oficial Criativo Dinâmico	Configuração	
1	857.048.557-33	Newton Saraiva	Administrador Comunicações Internacional	Prestação de contas	
2	15.664.328/3733-77	Dr. Sr. Solange Macedo	Designer Identidade Direto	Métricas	
3	02.328.238/0877-86	Celina Carvalho Jr.	NaN	Qualidade	
4	30.073.687/4087-40	Aurilo Martins	Especialista Paradigma Internacional	Programa	
...	...	...	...	...	
999995	202.970.542-01	Aurilo Franco Neto	Agente Funcionalidade Investidor	Otimização	
999996	76.245.248/0354-84	Iata Martins	Técnico A infraestrutura Central	Rede	
999997	15.787.105/2002-06	Erica Melo	Associado Operações Cliente	Garantia	
999998	129.226.816-61	Jurema Carvalho Filho Neto	Agente Operações Legado	Programa	
999999	971.454.113-18	Sr. Jair Martins	Produtor Comunicações Frente	Interações	

1000000 rows × 12 columns

Particionamento dos dados

1: Parquet por estados

In [10]:

```python
dataframeStates = dataframe
path_Inicial = 'Problema2/ParquetPorEstados'
if not os.path.isdir(path_Inicial):
    os.makedirs(path_Inicial)
estados = dataframeStates['SGL_UF'].drop_duplicates().values.tolist()
for uf in tqdm(estados):
    perStateData = dataframeStates.loc[dataframeStates['SGL_UF'] == uf]
    perStateData.to_parquet(path_Inicial+'/'+uf+'.parquet')
```

```
100%|██████████████████████████████████████████████████
████████| 27/27 [00:08<00:00,  3.11it/s]
```

2: CSV por Ano, mes e dia (Operação demorada, trabalhar com CSV's não é uma boa pratica pois eles não possuem datatype como o parquet, o que os torna pesados e lentos de trabalhar)

In [31]:

```python
dataframeAniversario = dataframe
path_Inicial = 'Problema2/CsvPorData'
if not os.path.isdir(path_Inicial):
    os.makedirs(path_Inicial)
datas = dataframe['DATA_ANIVERSARIO'].drop_duplicates().values.tolist()
for dat in tqdm(datas):
    path =path_Inicial+'/'+(datetime.strptime(dat,'%d/%m/%Y' )).strftime('%Y/%m/%d')
    if not os.path.isdir(path):
        os.makedirs(path)
    dataf = dataframeAniversario.loc[dataframeAniversario['DATA_ANIVERSARIO'].str.contains(
    dataf.to_csv(path+'/'+dat.replace('/','-')+'.csv', sep=';', encoding='utf-8')
```

```
100%|██████████████████████████████████████████████████
███| 10249/10249 [1:06:02<00:00,  2.59it/s]
```

In [ ]:

```python

```