DESAFIO INDIVIDUAL

Nivel Infra O Dataset deve ser salvo em ambiente cloud(Cloud Storage) O arquivo original e tratado deve ser salvo em MongoDB Atlas em coleções diferentes Os DataFrames devem ser obrigatoriamente salvos em uma bucket do CloudStorage

Nivel Pandas O arquivo está em outra linguagem e deve ter seus dados traduzidos para Português-BR Realizar a extração corretamente para um dataframe Verificar a existência de dados inconsistentes e realizar a limpeza para NaN ou NA Realizar o drop(se necessário) de colunas do dataframe realizando o comentário do porque da exclusão Todos os passos devem ser comentados

Nivel PySpark (Funções básicas vistas em aula) Deverá ser montada a estrutura do DataFrame utilizando o StructType. Verificar a existência de dados inconsistentes, nulos e realizar a limpeza. Verificar a necessidade de drop em colunas ou linhas. Caso seja necessário, fazer comentário do porque. Realizar a mudança de nome de pelo menos 2 colunas Deverá criar pelo menos duas novas colunas contendo alguma informação relevante sobre as outras colunas já existentes (Funções de Agrupamento, Agregação ou Joins). (Use a sua capacidade analítica) Deverá utilizar filtros, ordenação e agrupamento, trazendo dados relevantes para o negócio em questão. (Use a sua capacidade analítica) Utilizar pelo menos duas Window Functions Nivel SparkSQL Utilizar no minimo 5 consultas diferentes utilizando o SparkSQL, comentando o porquê de ter escolhido essas funções e explicando o que cada consulta faz. Nível DataStudio Construir um dashboard (maximo 1 pagina) para apresentação dos insights

Ferramentas

Colab ou Ides | Google Cloud | Data Studio

Observação : o que será analisado são os tópicos cumpridos , na ocasião de findar o tempo e algum(ns) não forem contemplados , realize a entrega do que foi concluído . Pedimos que nos conceda acesso ao email professores.bcw4@soulcodeacademy.org e professoresbc17@soulcodeacademy.org para seu projeto no google cloud , adicione um usuário : soulcode senha : a1b2c3 no mongo atlas e compartilhe junto a key de autenticação para acessarmos seu mongo atlas e seja enviado juntamente com o codigo realizado

LEITURA DO ARQUIVO VIA PANDAS COM A CRIAÇÃO DOS CONECTORES

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
#Importando os modulos e os dados do arquivo CSV e salvando em um DataFrame com pandas
```

```
import pandas
import pandas as pd
```

```
# Leitura de dados e maximizando exibição
df = pd.read_csv("marketing_campaign.csv", sep = ";")
pd.options.display.max_rows = 3000
pd.options.display.max_columns = 29
df
```

```
from google.cloud import storage
# Credenciais de acesso json(key)
client = storage.Client.from_service_account_json(json_credentials_path='boreal-quarter-349018-4f9a
#Indicando a bucket que vai ser usada
bucket = client.get_bucket('def_individual')
#Montando o nome do arquivo
object_name_in_gcs_bucket = bucket.blob('marketing_campaign.csv')
#Fazendo o upload para o bucket
object_name_in_gcs_bucket.upload_from_filename('marketing_campaign.csv')
```

```
#conector para mongodb
import pymongo
myclient = pymongo.MongoClient("mongodb+srv://root:Eddiesp16@cluster0.4u1yb.mongodb.net/filialb.fil
mydb = myclient["filialb"]
mycol = mydb["filialb"]
```

```
#inserindo os dados no banco mongo convertendo para dicionarios
data = df.to_dict(orient = "records")
db = myclient["def_ind0"]
db.def_ind0.insert_many(data)
```

```
<pymongo.results.InsertManyResult at 0x1ba6f75f400>
```

Nivel Pandas O arquivo está em outra linguagem e deve ter seus dados traduzidos para Português-BR Realizar a extração corretamente para um dataframe Verificar a existência de dados inconsistentes e realizar a limpeza para NaN ou NA Realizar o drop(se necessário) de colunas do dataframe realizando o comentário do porque da exclusão Todos os passos devem ser comentados

```
# Exibir informações sobre dados
df.info()
df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 29 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   ID                   2240 non-null   int64  
 1   Year_Birth           2240 non-null   int64  
 2   Education            2240 non-null   object 
 3   Marital_Status       2240 non-null   object 
 4   Income               2216 non-null   float64
 5   Kidhome              2240 non-null   int64  
 6   Teenhome             2240 non-null   int64  
 7   Dt_Customer          2240 non-null   object 
 8   Recency              2240 non-null   int64  
 9   MntWines             2240 non-null   int64  
 10  MntFruits            2240 non-null   int64  
 11  MntMeatProducts      2240 non-null   int64  
 12  MntFishProducts      2240 non-null   int64  
 13  MntSweetProducts     2240 non-null   int64  
 14  MntGoldProds         2240 non-null   int64  
 15  NumDealsPurchases    2240 non-null   int64  
 16  NumWebPurchases      2240 non-null   int64  
 17  NumCatalogPurchases  2240 non-null   int64  
 18  NumStorePurchases    2240 non-null   int64  
 19  NumWebVisitsMonth    2240 non-null   int64  
 20  AcceptedCmp3         2240 non-null   int64  
 21  AcceptedCmp4         2240 non-null   int64  
 22  AcceptedCmp5         2240 non-null   int64  
 23  AcceptedCmp1         2240 non-null   int64  
 24  AcceptedCmp2         2240 non-null   int64  
 25  Complain             2240 non-null   int64  
 26  Z_CostContact        2240 non-null   int64  
 27  Z_Revenue            2240 non-null   int64  
 28  Response             2240 non-null   int64  
dtypes: float64(1), int64(25), object(3)
memory usage: 507.6+ KB
```

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Cust
0	5524	1957	Graduation	Single	58138.0	0	0	2012-(

```
#Renomeando as colunas em português
df.rename(columns={'Year_Birth':'Ano_nascimento',
                   'Education':'Escolaridade',
                   'Marital_Status':'Estado_civil',
                   'Income':'Renda',
                   'Kidhome':'Qtd_criancasnacasa',
                   'Teenhome':'Qtd_adoslecentesnacasa',
                   'Dt_Customer':'Data_cadastro',
                   'Recency':'Dias_ultimacompra',
                   'MntWines':'Gastos_vinhos',
                   'MntFruits':'Gastos_frutas',
                   'MntMeatProducts':'Gastos_carnes',
                   'MntFishProducts':'Gastos_peixes',
                   'MntSweetProducts':'Gastos_doces',
                   'MntGoldProds':'Gastos_ouro',
                   'NumDealsPurchases':'Qtd_compras_descontos',
                   'NumWebPurchases':'Qtd_compras_site',
                   'NumCatalogPurchases':'Qtd_compras_catalogo',
                   'NumStorePurchases':'Qtd_compras_loja',
                   'NumWebVisitsMonth':'Qtd_visitas_site_mes',
```

```
                    'AcceptedCmp3':'Aceito_cmp3',
                    'AcceptedCmp4':'Aceito_cmp4',
                    'AcceptedCmp5':'Aceito_cmp5',
                    'AcceptedCmp1':'Aceito_cmp41',
                    'AcceptedCmp2':'Aceito_cmp2',
                    'Complain':'Reclamacao',
                    'Z_CostContact':'Custo_contato',
                    'Z_Revenue':'Imposto',
                    'Response':'Resposta',
                    }, inplace=True)

df.head()


df['Escolaridade']


#Renomeando as linhas da coluna escolaridade
df.replace(['Graduation'], 'Graduado', inplace=True)
df.replace(['PhD'], 'Doutorado', inplace=True)
df.replace(['Master'], 'Mestrado', inplace=True)
df.replace(['Basic'], 'Básico', inplace=True)
df.replace(['2n Cycle'], 'Pós_gradução', inplace=True)


#Exibir a coluna Estado Civil
df['Estado_civil']


#Trocando os valores das linhas e traduzindo
df.replace(['Single'], 'Solteiro', inplace=True)
df.replace(['Together'], 'Juntos', inplace=True)
df.replace(['Married'], 'Casado', inplace=True)
df.replace(['Divorced'], 'Divorciado', inplace=True)
df.replace(['Widow'], 'Viúvo', inplace=True)
df.replace(['Alone'], 'Solteiro', inplace=True)
df.replace(['Absurd'], 'Solteiro', inplace=True)
df.replace(['YOLO'], 'Solteiro', inplace=True)


df.head(1)


#Exibe a coluna Data_cadastro
df['Data_cadastro']


#Muda o valor da coluna para data
df['Data_cadastro'] = pd.to_datetime(df.Data_cadastro)


# monstra a contagem dos valores nulos
df.isnull().sum()


#Exbibe a coluna renda
df['Renda']


# Backup da variavel df atribuindo outra variavel
df2 = df
```

```
#Remove a linha com valores nulos
df.dropna(inplace=True)
```

```
#Exibe os valores nulos no dataframe
df.isna()
```

```
#Exibe todo o dataframe
df
```

```
#Converte o arquivo pandas em csv
df.to_csv('tratadapandas.csv')
```

```
df = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/def/tratadapandas.csv", sep = ";")
```

```
from pymongo import MongoClient
```

```
#conector para mongodb
import pymongo
myclient = pymongo.MongoClient("mongodb+srv://root:Eddiesp16@cluster0.4u1yb.mongodb.net/def_ind_fin
mydb = myclient["def_ind_final"]
mycol = mydb["def_ind_final"]
```

```
/usr/local/lib/python3.7/dist-packages/pymongo/common.py:787: UserWarning: Unknow
  warnings.warn(str(exc))
```

```
#inserindo os dados no banco mongo convertendo para dicionarios
data = df.to_dict(orient = "records")
db = myclient["def_ind_final"]
db.def_ind_final.insert_many(data)
```

```
from google.cloud import storage
# Credenciais de acesso json(key)
client = storage.Client.from_service_account_json(json_credentials_path='boreal-quarter-349018-4f9a
#Indicando a bucket que vai ser usada
bucket = client.get_bucket('def_individual')
#Montando o nome do arquivo
object_name_in_gcs_bucket = bucket.blob('marketing_campaign.csv')
#Fazendo o upload para o bucket
object_name_in_gcs_bucket.upload_from_filename('tratadapandas.csv')
```

```
#Instalando o pyspark
!pip install pyspark
```

```
#Instalando o gcsfs para usar o cloud storage
!pip install gcsfs
```

```
#CHAMAR AS BIBLIOTECAS/MÓDULOS NECESSÁRIAS
import pyspark
```

```
from pyspark.sql import SparkSession
from pyspark import SparkConf
import pyspark.sql.functions as F



#CHAMAR AS BIBLIOTECAS/MÓDULOS NECESSÁRIAS
from google.cloud import storage
import os


#CONFIGURAÇÃO DA CHAVE DE SEGURANÇA
serviceAccount = '/content/drive/MyDrive/Colab Notebooks/def/boreal-quarter-349018-4f9adad0db3c.jso
os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = serviceAccount


#CÓDIGO QUE ACESSA A BUCKET CRIADA E FAZ O DOWNLOAD DOS ARQUIVOS VIA PYSPARK
client = storage.Client()

#CRIAR VARIÁVEL PARA RECEBER O NOME DA BUCKET
bucket = client.get_bucket('def_individual')

#USA O MÉTODO BLOB PARA RETORNAR O NOME DO ARQUIVO
bucket.blob('marketing_campaign.csv')

#CRIA A VARIÁVEL PATH COLOCANDO O CAMINHO DE URI
path = 'gs://def_individual/marketing_campaign.csv'


#CRIAR A SPARK SESSION E LER O ARQUIVO VIA PYSPARK
spark = (
    SparkSession.builder
                .master('local')
                .appName('spark-gcs')
                .config('spark.ui.port', '4050')
                .config("spark.jars", 'https://storage.googleapis.com/hadoop-lib/gcs/gcs-connector-
                .getOrCreate()
)


# usa o objeto StructType
from pyspark.sql.types import StructType, StructField, IntegerType, StringType, DateType, DoubleTyp
schema_df = StructType([
    StructField("ID", IntegerType()),
    StructField("Year_Birth", IntegerType()),
    StructField("Education", StringType()),
    StructField("Marital_Status", StringType()),
    StructField("Income", IntegerType()),
    StructField("Kidhome", IntegerType()),
    StructField("Teenhome", IntegerType()),
    StructField("Dt_Customer", StringType()),
    StructField("Recency", IntegerType()),
    StructField("MntWines", IntegerType()),
    StructField("MntMeatProducts", IntegerType()),
    StructField("MntFishProducts", IntegerType()),
    StructField("MntSweetProducts", IntegerType()),
    StructField("MntGoldProds", IntegerType()),
    StructField("NumDealsPurchases", IntegerType()),
    StructField("NumWebPurchases", IntegerType()),
    StructField("NumCatalogPurchases", IntegerType()),
    StructField("NumStorePurchases", IntegerType()),
    StructField("NumWebVisitsMonth", IntegerType()),
```

```
    StructField("AcceptedCmp3", IntegerType()),
    StructField("AcceptedCmp4", IntegerType()),
    StructField("AcceptedCmp5", IntegerType()),
    StructField("AcceptedCmp1", IntegerType()),
    StructField("AcceptedCmp2", IntegerType()),
    StructField("Complain", IntegerType()),
    StructField("Z_CostContact", IntegerType()),
    StructField("Z_Revenue", IntegerType()),
    StructField("Response", IntegerType())


])

#CRIAR UM DATAFRAME PELO PYSPARK com base no structtype
df_spark = ( spark.read.format('csv')
                      .option('delimiter', ';')
                      .option('header', 'true')
                      .option('inferschema', 'true')
                      .load('gs://def_individual/marketing_campaign.csv', schema = schema_df)

)


#Verificar os tipos das colunas
df_spark.printSchema()
```

```
root
 |-- ID: integer (nullable = true)
 |-- Year_Birth: integer (nullable = true)
 |-- Education: string (nullable = true)
 |-- Marital_Status: string (nullable = true)
 |-- Income: integer (nullable = true)
 |-- Kidhome: integer (nullable = true)
 |-- Teenhome: integer (nullable = true)
 |-- Dt_Customer: string (nullable = true)
 |-- Recency: integer (nullable = true)
 |-- MntWines: integer (nullable = true)
 |-- MntMeatProducts: integer (nullable = true)
 |-- MntFishProducts: integer (nullable = true)
 |-- MntSweetProducts: integer (nullable = true)
 |-- MntGoldProds: integer (nullable = true)
 |-- NumDealsPurchases: integer (nullable = true)
 |-- NumWebPurchases: integer (nullable = true)
 |-- NumCatalogPurchases: integer (nullable = true)
 |-- NumStorePurchases: integer (nullable = true)
 |-- NumWebVisitsMonth: integer (nullable = true)
 |-- AcceptedCmp3: integer (nullable = true)
 |-- AcceptedCmp4: integer (nullable = true)
 |-- AcceptedCmp5: integer (nullable = true)
 |-- AcceptedCmp1: integer (nullable = true)
 |-- AcceptedCmp2: integer (nullable = true)
 |-- Complain: integer (nullable = true)
 |-- Z_CostContact: integer (nullable = true)
 |-- Z_Revenue: integer (nullable = true)
 |-- Response: integer (nullable = true)
```

```
df_spark.show(1)
```

```
+----+----------+----------+--------------+------+-------+--------+-----------+--
|  ID|Year_Birth| Education|Marital_Status|Income|Kidhome|Teenhome|Dt_Customer|Re
+----+----------+----------+--------------+------+-------+--------+-----------+--
|5524|      1957|Graduation|        Single| 58138|      0|       0| 2012-09-04|
+----+----------+----------+--------------+------+-------+--------+-----------+--
only showing top 1 row
```

```
#Exibe os soma dos nulos
df_spark.toPandas().isna().sum()
```

```
ID                     0
Year_Birth             0
Education              0
Marital_Status         0
Income                24
Kidhome                0
Teenhome               0
Dt_Customer            0
Recency                0
MntWines               0
MntMeatProducts        0
MntFishProducts        0
MntSweetProducts       0
MntGoldProds           0
NumDealsPurchases      0
NumWebPurchases        0
NumCatalogPurchases    0
NumStorePurchases      0
NumWebVisitsMonth      0
AcceptedCmp3           0
AcceptedCmp4           0
AcceptedCmp5           0
AcceptedCmp1           0
AcceptedCmp2           0
Complain               0
Z_CostContact          0
Z_Revenue              0
Response               0
dtype: int64
```

```
# remove qualquer linha nula de qualquer coluna
df_spark = df_spark.na.drop()

# Foi removido as linhas com os valores nulos equivalente a 1,07% do total


#Exibe os soma dos nulos
df_spark.toPandas().isna().sum()
```

```
ID                     0
Year_Birth             0
Education              0
Marital_Status         0
Income                 0
Kidhome                0
Teenhome               0
```

```
Dt_Customer            0
Recency                0
MntWines               0
MntMeatProducts        0
MntFishProducts        0
MntSweetProducts       0
MntGoldProds           0
NumDealsPurchases      0
NumWebPurchases        0
NumCatalogPurchases    0
NumStorePurchases      0
NumWebVisitsMonth      0
AcceptedCmp3           0
AcceptedCmp4           0
AcceptedCmp5           0
AcceptedCmp1           0
AcceptedCmp2           0
Complain               0
Z_CostContact          0
Z_Revenue              0
Response               0
dtype: int64
```

```
df_spark.show(1)
```

```
+----+----------+----------+--------------+------+-------+--------+-----------+--
|  ID|Year_Birth| Education|Marital_Status|Income|Kidhome|Teenhome|Dt_Customer|Re
+----+----------+----------+--------------+------+-------+--------+-----------+--
|5524|      1957|Graduation|        Single| 58138|      0|       0| 2012-09-04|
+----+----------+----------+--------------+------+-------+--------+-----------+--
only showing top 1 row
```

```
#Renomeando as colunas Education e Marital_Status
df_spark = df_spark.withColumnRenamed('Education', 'Escolaridade')
df_spark = df_spark.withColumnRenamed('Marital_Status', 'Estado_civel')
```

```
# Exibe os valores alterados da coluna
df_spark.show(1)
```

```
+----+----------+------------+------------+------+-------+--------+-----------+--
|  ID|Year_Birth|Escolaridade|Estado_civel|Income|Kidhome|Teenhome|Dt_Customer|Re
+----+----------+------------+------------+------+-------+--------+-----------+--
|5524|      1957|  Graduation|      Single| 58138|      0|       0| 2012-09-04|
+----+----------+------------+------------+------+-------+--------+-----------+--
only showing top 1 row
```

```
#Criando uma nova coluna com a soma das colunas Kidhome e Teenhome
#soma total das crianças e adolescentes
df_spark = df_spark.withColumn('Kidhomesum', F.col('Kidhome') + F.col('Teenhome'))
#Criando uma nova coluna com o total de gastos com a soma do tipos de itens gastos ao longo de 2 an
df_spark = df_spark.withColumn('Totalgastos', F.col('MntWines') + F.col('MntMeatProducts')+ F.col('
```

```
df_spark.show(5)
```

```
+----+----------+------------+------------+------+-------+--------+-----------+--
|  ID|Year_Birth|Escolaridade|Estado_civel|Income|Kidhome|Teenhome|Dt_Customer|Re
+----+----------+------------+------------+------+-------+--------+-----------+--
|5524|      1957|  Graduation|      Single| 58138|      0|       0| 2012-09-04|
|2174|      1954|  Graduation|      Single| 46344|      1|       1| 2014-03-08|
|4141|      1965|  Graduation|    Together| 71613|      0|       0| 2013-08-21|
|6182|      1984|  Graduation|    Together| 26646|      1|       0| 2014-02-10|
|5324|      1981|         PhD|     Married| 58293|      1|       0| 2014-01-19|
+----+----------+------------+------------+------+-------+--------+-----------+--
only showing top 5 rows
```

```
#exibe a soma de rendas por escolaridade
df_spark.groupBy('Escolaridade').sum('Income').show()
```

```
+------------+-----------+
|Escolaridade|sum(Income)|
+------------+-----------+
|    2n Cycle|    9526638|
|         PhD|   27005896|
|      Master|   19314900|
|  Graduation|   58835937|
|       Basic|    1096538|
+------------+-----------+
```

```
#exibe a soma de Total gasto por escolaridade
df_spark.groupBy('Escolaridade').sum('Totalgastos').show()
```

```
+------------+----------------+
|Escolaridade|sum(Totalgastos)|
+------------+----------------+
|    2n Cycle|           89610|
|         PhD|          309923|
|      Master|          207896|
|  Graduation|          637239|
|       Basic|            3184|
+------------+----------------+
```

```
#exibe a média de Total gasto por escolaridade
df_spark.groupBy('Escolaridade').mean('Totalgastos').show()
```

```
+------------+-----------------+
|Escolaridade| avg(Totalgastos)|
+------------+-----------------+
|    2n Cycle|           448.05|
|         PhD|644.3305613305613|
|      Master|569.5780821917808|
|  Graduation| 571.002688172043|
|       Basic|58.96296296296296|
+------------+-----------------+
```

```
df_spark.show(10)
```

```
+----+----------+------------+------------+------+-------+--------+-----------+--
|  ID|Year_Birth|Escolaridade|Estado_civel|Income|Kidhome|Teenhome|Dt_Customer|Re
+----+----------+------------+------------+------+-------+--------+-----------+--
|5524|      1957|  Graduation|      Single| 58138|      0|       0| 2012-09-04|
|2174|      1954|  Graduation|      Single| 46344|      1|       1| 2014-03-08|
|4141|      1965|  Graduation|    Together| 71613|      0|       0| 2013-08-21|
|6182|      1984|  Graduation|    Together| 26646|      1|       0| 2014-02-10|
|5324|      1981|         PhD|     Married| 58293|      1|       0| 2014-01-19|
|7446|      1967|      Master|    Together| 62513|      0|       1| 2013-09-09|
| 965|      1971|  Graduation|    Divorced| 55635|      0|       1| 2012-11-13|
|6177|      1985|         PhD|     Married| 33454|      1|       0| 2013-05-08|
|4855|      1974|         PhD|    Together| 30351|      1|       0| 2013-06-06|
|5899|      1950|         PhD|    Together|  5648|      1|       1| 2014-03-13|
+----+----------+------------+------------+------+-------+--------+-----------+--
only showing top 10 rows
```

```
#Importando os modulos do window functions
from pyspark.sql.window import Window
from pyspark.sql.functions import col,avg,sum,min,max,row_number


# Criando a windows a partir do dataframe
windowPartitionAgg  = Window.partitionBy("Escolaridade")

# função agragada

#mostra mínimo particionado com escolaridade
df_spark.withColumn("Min",
             min(col("Income")).over(windowPartitionAgg))



# Criando a windows a partir do dataframe
windowPartitionAgg2  = Window.partitionBy("Escolaridade")

# função agragada

#mostra máximo particionado com escolaridade
df_spark.withColumn("Max",
             max(col("Income")).over(windowPartitionAgg)).show()


#Carregando o arquivo para o sparksql
df_test = (spark
      .read
      .format("csv")
      .option("header", "true")
      .option("inferschema", "true")
      .option("delimiter", ";")
      .load("gs://def_individual/marketing_campaign.csv")
      .createOrReplaceTempView("teste"))


#Exibe as primeiras 2 linhas
spark.sql('''SELECT * FROM teste''').show(2)
```

```
+----+----------+----------+--------------+------+-------+--------+-----------+--
|  ID|Year_Birth| Education|Marital_Status|Income|Kidhome|Teenhome|Dt_Customer|Re
+----+----------+----------+--------------+------+-------+--------+-----------+--
|5524|      1957|Graduation|        Single| 58138|      0|       0| 2012-09-04|
|2174|      1954|Graduation|        Single| 46344|      1|       1| 2014-03-08|
+----+----------+----------+--------------+------+-------+--------+-----------+--
only showing top 2 rows
```

```
#exibe Education e Income ordenado em descrecente
spark.sql('''
            SELECT Income
            FROM teste
            ORDER BY Income DESC;
          ''').show(10)
```

```
+------+
|Income|
+------+
|666666|
|162397|
|160803|
|157733|
|157243|
|157146|
|156924|
|153924|
|113734|
|105471|
+------+
only showing top 10 rows
```

```
#exibe Education e Income ordenado em ascendente
spark.sql('''
            SELECT Income
            FROM teste
            where Income >= 0
            ORDER BY Income ASC;
          ''').show(30)
```

```
+------+
|Income|
+------+
|  1730|
|  2447|
|  3502|
|  4023|
|  4428|
|  4861|
|  5305|
|  5648|
|  6560|
|  6835|
|  7144|
|  7500|
|  7500|
```

```
|  7500|
|  7500|
|  7500|
|  7500|
|  7500|
|  7500|
|  7500|
|  7500|
|  7500|
|  7500|
|  8028|
|  8820|
|  8940|
|  9255|
|  9548|
|  9722|
| 10245|
+------+
only showing top 30 rows
```

```
#exibe a quantidade de cadastro agrupado pela coluna Education
spark.sql('''
           SELECT
    Education,
    COUNT(Education) headcount
FROM
    teste
GROUP BY
    Education;
           ;
         ''').show()
```

```
+----------+---------+
| Education|headcount|
+----------+---------+
|  2n Cycle|      203|
|       PhD|      486|
|    Master|      370|
|Graduation|     1127|
|     Basic|       54|
+----------+---------+
```

```
#exibe a quantidade de cadastro agrupado pela coluna Education
spark.sql('''
           SELECT

    COUNT(Education) headcount
FROM
    teste
GROUP BY
    Education;
           ;
         ''').show()
```

```
#exibe da tabela
spark.sql('''DESCRIBE teste''').show()
```

```
+------------------+---------+-------+
|          col_name|data_type|comment|
+------------------+---------+-------+
|                ID|      int|   null|
|        Year_Birth|      int|   null|
|         Education|   string|   null|
|    Marital_Status|   string|   null|
|            Income|      int|   null|
|           Kidhome|      int|   null|
|          Teenhome|      int|   null|
|       Dt_Customer|   string|   null|
|           Recency|      int|   null|
|          MntWines|      int|   null|
|         MntFruits|      int|   null|
|   MntMeatProducts|      int|   null|
|   MntFishProducts|      int|   null|
|  MntSweetProducts|      int|   null|
|      MntGoldProds|      int|   null|
| NumDealsPurchases|      int|   null|
|   NumWebPurchases|      int|   null|
|NumCatalogPurchases|      int|   null|
| NumStorePurchases|      int|   null|
| NumWebVisitsMonth|      int|   null|
+------------------+---------+-------+
only showing top 20 rows
```