<?xml version="1.0" encoding="UTF-8"?>

UNIVERSIDADE
VILA VELHA
ESPÍRITO SANTO

PROJETO INTEGRADOR DE CIÊNCIA DE DADOS

WORKFLOW

Link para meu projeto: https://colab.research.google.com/drive/1suP4Meb561I6RBkglu_5ej0vPD0HWx62?usp=sharing (https://colab.research.google.com/drive/1suP4Meb561I6RBkglu_5ej0vPD0HWx62?usp=sharing)

Aluno : Thiago Rosemberg Jager

Curso: Tecnologia em Ciência de Dados-UVV

Email: thijager@gmail.com

Contato: 27 99787-6070

In [ ]:

ECOSSISTEMA PYTHON:

Link: Aqui!!! (https://colab.research.google.com/drive/10FW5w5QbxRZYkaj79fhqu9ujl8wY-Uuy?usp=sharing)

BIBLIOTECAS E FUNÇÕES

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from statsmodels.tsa.stattools import adfuller
from datetime import datetime as dt
import pytz
import json
import os
```

In [2]:
```
#bibliotecas específicas
```

In [3]:
```
# MODELO DE CLASSIFICACAO
from sklearn.model_selection import train_test_split
from sklearn import datasets
from sklearn import svm

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import requests
import json
import scipy.stats as stats

# Example of the Normality Test
from scipy.stats import anderson
from scipy.stats import shapiro
from scipy.stats import normaltest

# Example of the Correlation Tests
from scipy.stats import pearsonr
from scipy.stats import spearmanr
from scipy.stats import kendalltau
from scipy.stats import chi2_contingency

# Parametric Statistical Hypothesis Tests

# Example of the Student's t-test
from scipy.stats import ttest_ind
# Example of the Paired Student's t-test
from scipy.stats import ttest_rel
# Example of the Analysis of Variance Test: ANOVA
from scipy.stats import f_oneway

# Nonparametric Statistical Hypothesis Tests

# Example of the Mann-Whitney U Test
from scipy.stats import mannwhitneyu
# Example of the Wilcoxon Signed-Rank Test
from scipy.stats import wilcoxon
# Example of the Kruskal-Wallis H Test
from scipy.stats import kruskal
# Example of the Friedman Test
from scipy.stats import friedmanchisquare

# Example of the Augmented Dickey-Fuller unit root test
from statsmodels.tsa.stattools import adfuller

import seaborn as sns
```

In [4]:
```
def Normality_Test(data):
        result = anderson(data)
        print(f'DISTRIBUIÇÃO AMOSTRAL: stat = {result.statistic:.3f}')
        for i in range(len(result.critical_values)):
                sl, cv = result.significance_level[i], result.critical_values[i]
                if result.statistic < cv:
                        print(f'Intervalo de Confiança ({sl:.1f} %): Probably Gaussia
n.')
                else:
                        print(f'Intervalo de Confiança ({sl:.1f} %): Probably Not Gau
ssian.')
```

In [5]:
```
def statistic(_dataset):
  """
  DOCSTRING:
  Describe (without count) + Insert range: max - min
  """
  _describe = _dataset.describe()
  _describe = _describe.drop('count')
  _describe.loc['range'] = _describe.max() - _describe.min()
  _describe = np.around(_describe.sort_values(), 2)
  return _describe
```

In [6]:
```
def PlotarStatistic(ts, yLabel = None):
  plt.title(f'ESTATÍSTICA DESCRITIVA DO DATASET: Mean ($\mu$) = {ts.mean(): .2f}')
  _describe = statistic(ts)
  ts.agg(statistic).plot(figsize = [15, 6])
  plt.ylabel(yLabel)
  plt.show()
  print('ESTATÍSTICA DESCRITIVA DO DATASET:')
  print(_describe)
```

In [7]:
```
def PlotarSerie(ts, title = None, yLabel = None, xLabel = None):
  ts.plot(figsize = [15, 6])
  resultados = Estacionariedade(ts)
  plt.title(f'SÉRIE TEMPORAL: $p$ Value: {resultados[1]: .5f}: Mean ($\mu$): {ts.mean
(): .2f}')
  plt.ylabel(yLabel)
  plt.xlabel(xLabel)
  plt.hlines(ts.mean(), ts.index[0], ts.index[len(ts) - 1], colors='red', linestyles
= '--')
  plt.show()
```

In [8]:
```
def PlotarSerieMultiplas(ts, title = None, yLabel = None, xLabel = None):
  ts.plot(figsize = [15, 6])
  plt.title(f'SÉRIE TEMPORAL MÚLTIPLAS')
  plt.ylabel(yLabel)
  plt.xlabel(xLabel)
  plt.show()
```

In [9]:
```
def PlotarPredicaoMultiplas(ts, minutos = 2, title = None, yLabel = None, xLabel = No
ne):
  ts_m = ts.rolling(minutos, min_periods=0).corr()
  ts_m.plot(figsize = [15, 6])
  plt.title(title)
  plt.ylabel(yLabel)
  plt.xlabel(xLabel)
  plt.show()
```

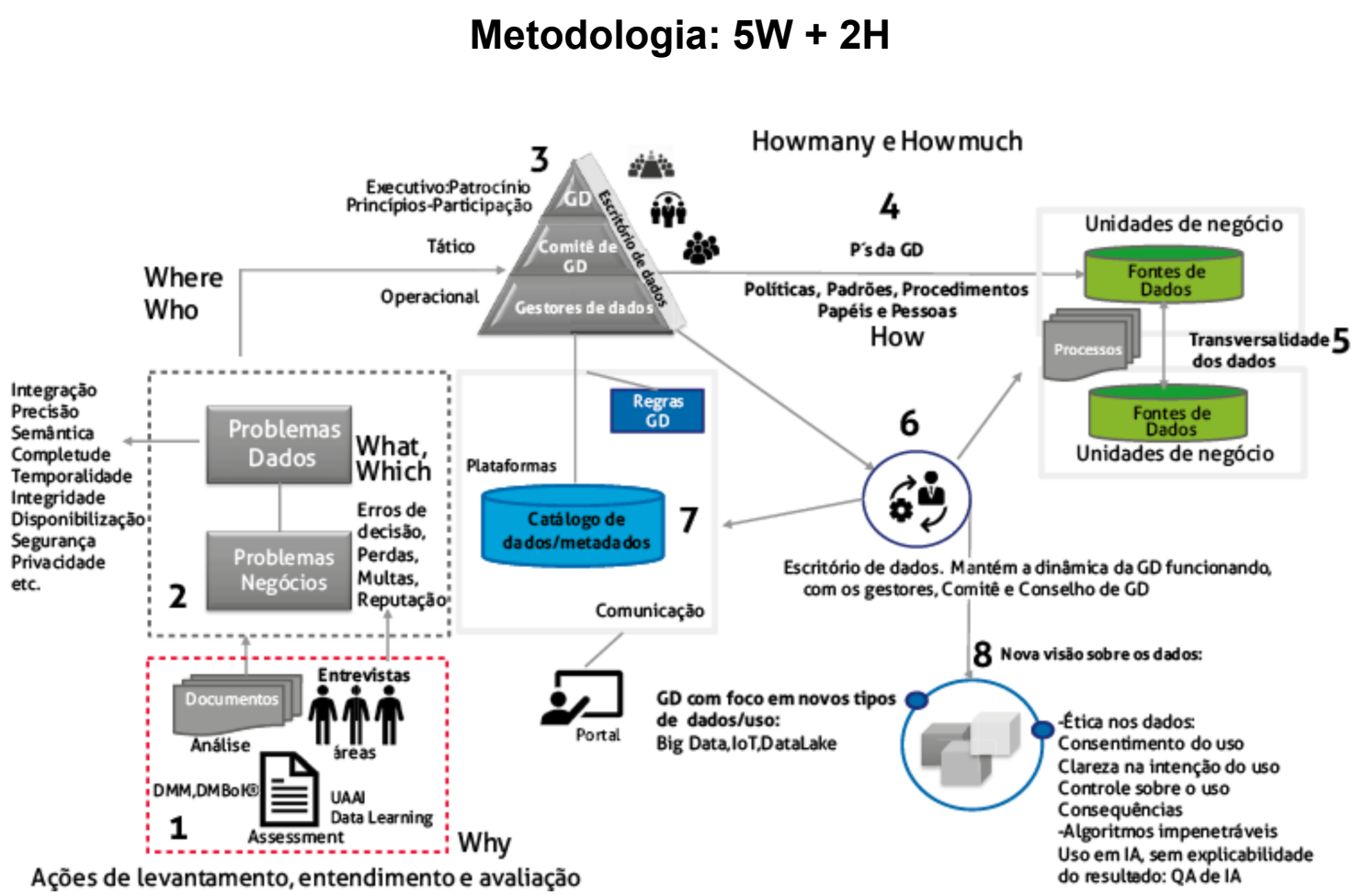
In [10]:
```
def PlotarSerieAcumulada(ts, title = None, yLabel = None, xLabel = None, janelaMovel
= 1):

  # VALOR ACUMULADO DA SÉRIE TEMPORAL:
  array_ts = np.zeros(61, dtype=int)
  for index, carros in enumerate(ts[:'2021-10-01 12:59:00']):
    array_ts[index + 1] = array_ts[index] + carros

  rng = pd.date_range('10/1/2021 12:00', periods=61, freq='T')
  ts_acumulado = pd.Series(array_ts, index=rng)

  # JANELA MÓVEL
  indexer = pd.api.indexers.FixedForwardWindowIndexer(window_size=janelaMovel)
  rolling_mean = ts_acumulado.rolling(window=indexer, min_periods=janelaMovel).mean()
  rolling_mean = rolling_mean.shift(janelaMovel) # Deslocamento da Janela = janelaMov
el
  rolling_mean.plot(label=f"$\hatY$: VALOR ESTIMADO DA SÉRIE TEMPORAL - JANELA MÓVEL
 (MINUTOS) = {janelaMovel}")

  ts_acumulado.plot(figsize = [15, 6], label='$Y$: VALOR REAL DA SÉRIE TEMPORAL')
  resultados = Estacionariedade(ts_acumulado)
  plt.title(f'SÉRIE TEMPORAL: $p$ Value: {resultados[1]: .5f}: Mean ($\mu$): {ts_acum
ulado.mean(): .2f}')
  plt.ylabel(yLabel)
  plt.xlabel(xLabel)
  plt.legend()
  plt.hlines(ts_acumulado.mean(), ts_acumulado.index[0], ts_acumulado.index[len(ts_ac
umulado) - 1], colors='red', linestyles = '--')
  plt.show()
```

In [11]:
```
def Estacionariedade(ts):
  resultados = adfuller(ts, autolag='AIC')
  print('DADOS ESTATÍSTICOS IMPORTANTES:')
  print(f'Estatística ADS: {resultados[0]: .2f}')
  print(f'Valor p: {resultados[1]: .10f}')
  print(f'Número de Atrasos: {resultados[2]}')
  print('Valores Críticos: ')
  for key, value in resultados[4].items():
    print(f'{key}: {value: .2f}')
  return resultados
```

In [12]:
```
def Normalizacao(group):
  group['normed_total'] = (group.total / group.total.sum()) * 100
  return group
```

In [13]:
```
def PlotarDistrit(dataset, bins = 16):
  num_bins = bins

  fig, ax = plt.subplots()

  # the histogram of the data
  n, bins, patches = ax.hist(dataset, num_bins, density = True, cumulative = False)

  # add a 'best fit' line
  mu = np.mean(dataset)
  Q2 = np.median(dataset)
  sigma = np.std(dataset)


  y = ((1 / (np.sqrt(2 * np.pi) * sigma)) *
       np.exp(-0.5 * (1 / sigma * (bins - mu)) ** 2))
  ax.plot(bins, y, '--', color='red')
  ax.set_xlabel('Total Gasto pelo Cliente')
  ax.set_ylabel('Probability density')
  ax.set_title(f'Histogram of IQ: $\mu =$ {mu: .1f}, $Q_{2}=$ {Q2: .1f}, $\sigma =${s
igma: .1f}')

  #
  fig.tight_layout()
  plt.show()
```

Governança de Dados



Referência bibliográfica:

BARBIERI, Carlos. Governança de dados. [Digite o Local da Editora]: Editora Alta Books, 2020. 9788550815435. E-book. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788550815435/ (https://integrada.minhabiblioteca.com.br/#/books/9788550815435/). Acesso em: 05 ago. 2022.

Metodologia para Análise de Dados

Para fazer uma análise de dados de forma correta, concisa e que agregue valor ao negócio, é necessário seguir algumas etapas para evitar problemas e extrair resultados ruins, como já diz o ditado:

"Dados ruins, Resultados ruins" do inglês "garbage in, garbage out".

Para isso, já existem algumas metodologias/processos para se fazer análise de dados,

Workflow:

Data Science Workflow: Ciclo de vida:

![image.png]

(

ETAPA 1: ETL - Extract, Transform and Load

Atividade 1 (A1): Pesquisar o dataset

DATASET - Colocar seu link aqui: https://www.kaggle.com/datasets/gabrielramos87/an-online-shop-business?select=Sales+Transaction+v.4a.csv (https://www.kaggle.com/datasets/gabrielramos87/an-online-shop-business?select=Sales+Transaction+v.4a.csv)

DESCRIÇÃO DAS CARACTERÍSTICAS (**OBRIGATÓRIAS**) DO DATASET:

1. REQUISITOS 1: Dados (COLUNAS) sobre **BUSINESS**: CLIENTE (primary key) + PRODUTOS ou SERVIÇOS (Desejável) + Coluna (**OBRIGATÓRIA**) de Dinheiro (R$ ou US\$ ou qualquer moeda).
2. REQUISITOS 2: No mínimo ou acima de **[5000-500000] LINHAS**: Eventos: Clientes + ...
3. REQUISITOS 3: **NÃO** é permitido simular os dados de entrada X/saída: $/F(X)$ do conjunto.
4. REQUISITOS 4: Dados **posteriores a 2020** até atual.

```
In [14]: dataset=pd.read_excel('/content/Sales Transaction3.5 v.4a.xlsx')
```

In [15]:
```
dataset
```

Out[15]:

	TransactionNo	Date	ProductNo	ProductName	CustomerNo	Country	Cost	Unity	Quantity
0	581482	2019-09-12 00:00:00	22485	Set Of 2 Wooden Market Crates	17490.0	United Kingdom	0.43	1.79	12
1	581475	2019-09-12 00:00:00	22596	Christmas Star Wish List Chalkboard	13069.0	United Kingdom	0.19	0.30	36
2	581475	2019-09-12 00:00:00	23235	Storage Tin Vintage Leaf	13069.0	United Kingdom	0.50	0.96	12
3	581475	2019-09-12 00:00:00	23272	Tree T-Light Holder Willie Winkie	13069.0	United Kingdom	0.08	0.89	12
4	581475	2019-09-12 00:00:00	23239	Set Of 4 Knick Knack Tins Poppies	13069.0	United Kingdom	0.17	1.99	6
...	...	...	...	...	...	...	...	...	..
49993	577855	11/22/2019	84828	Jungle Popsicles Ice Lolly Moulds	15877.0	United Kingdom	0.74	6.13	1
49994	577855	11/22/2019	22262	Felt Egg Cosy Chicken	15877.0	United Kingdom	0.78	7.29	1
49995	577855	11/22/2019	21383	Pack Of 12 Sticky Bunnies	15877.0	United Kingdom	0.36	7.29	1
49996	577855	11/22/2019	22530	Magic Drawing Slate Dolly Girl	15877.0	United Kingdom	0.22	7.29	1
49997	577855	11/22/2019	23392	Spaceboy Rocket Lolly Makers	15877.0	United Kingdom	0.99	7.29	1

49998 rows × 16 columns

In [16]:
```
# RENOMEANDO A COLUNA DATE PARA "DATA"
```

In [17]:
```
dataset.rename(columns = {'Date':'Data'}, inplace = True)
```

In [18]:
```
dataset
```

Out[18]:

	TransactionNo	Data	ProductNo	ProductName	CustomerNo	Country	Cost	Unity	Quantity
0	581482	2019-09-12 00:00:00	22485	Set Of 2 Wooden Market Crates	17490.0	United Kingdom	0.43	1.79	12
1	581475	2019-09-12 00:00:00	22596	Christmas Star Wish List Chalkboard	13069.0	United Kingdom	0.19	0.30	36
2	581475	2019-09-12 00:00:00	23235	Storage Tin Vintage Leaf	13069.0	United Kingdom	0.50	0.96	12
3	581475	2019-09-12 00:00:00	23272	Tree T-Light Holder Willie Winkie	13069.0	United Kingdom	0.08	0.89	12
4	581475	2019-09-12 00:00:00	23239	Set Of 4 Knick Knack Tins Poppies	13069.0	United Kingdom	0.17	1.99	6
...	...	...	...	...	...	...	...	...	..
49993	577855	11/22/2019	84828	Jungle Popsicles Ice Lolly Moulds	15877.0	United Kingdom	0.74	6.13	1
49994	577855	11/22/2019	22262	Felt Egg Cosy Chicken	15877.0	United Kingdom	0.78	7.29	1
49995	577855	11/22/2019	21383	Pack Of 12 Sticky Bunnies	15877.0	United Kingdom	0.36	7.29	1
49996	577855	11/22/2019	22530	Magic Drawing Slate Dolly Girl	15877.0	United Kingdom	0.22	7.29	1
49997	577855	11/22/2019	23392	Spaceboy Rocket Lolly Makers	15877.0	United Kingdom	0.99	7.29	1

49998 rows × 16 columns

In [19]:
```
# TRANSFORMANDO A COLUNA Age (idade)  QUE ESTÁ COMO FLOAT EM STRING ( POIS É UMA COLUNA DE VARIÁVEL QUALITATIVA)
```

Atividade 2 (A2): Descrição / Classificação de todas as colunas

DESCRIÇÃO - (INFORMAÇÕES DA TABELA)

In [20]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49998 entries, 0 to 49997
Data columns (total 16 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   TransactionNo  49998 non-null  object 
 1   Data           49998 non-null  object 
 2   ProductNo      49998 non-null  object 
 3   ProductName    49998 non-null  object 
 4   CustomerNo     49996 non-null  float64
 5   Country        49998 non-null  object 
 6   Cost           49998 non-null  float64
 7   Unity          49998 non-null  float64
 8   Quantity       49998 non-null  int64  
 9   Price          49998 non-null  float64
 10  Genre          49998 non-null  object 
 11  Age            49998 non-null  int64  
 12  Payment        49998 non-null  object 
 13  Total          49998 non-null  float64
 14  Profit         49998 non-null  float64
 15  Profile        49998 non-null  int64  
dtypes: float64(6), int64(3), object(7)
memory usage: 6.1+ MB
```

In [21]:
```
#TRANSFORMAR COLUNA STRING EM FLOAT
#dataset['profit'] = dataset['Profit'].str.replace(',','.').astype(float)
```

In [22]:
```
# CLASSIFICAÇÕES DAS COLUNAS DA TABELA
```

CLASSIFICAÇÃO - QUALITATIVA (NOMINAL OU ORDINAL) OU QUANTITATIVA (DISCRETA OU CONTÍNUA):

TransactionNo (categorica -Qualitativa): Número de 6 dígitos que define cada transação. A letra "c" no código indica cancelamento.

Date (numerico-Qualitatva): data em que cada transação foi gerada.

ProductNo (categorica -Qualitativa): 5 ou 6 dígitos foi usado como caractéres para identificar produtos únicos.

Product (categorica - Qualitativa): nome do ítem/produto.

Price (numerica- Quantitativa Contínua): O preço de cada produto ou unidade em Libras Esterlinas

Quantity (numerica -Quantitativa Discreta): Quantidade de cada produto por transação. Valores negativos são relatados como transações canceladas.

CustomerNo (categorica - Quantitativa): Um dígito de 5 ou 6 números que define cada cliente.

Country (categorica -Qualitativa Nominal): Nome dos países onde cada cliente reside.

In [23]:
```
# COLUNAS QUE PODERIAM SER CRIADAS PARA REALIZAR MAIS PESQUISAS DE MACHNE LEARNING E
 OUTROS GRÁFICOS.

#**Age** (Categórica - Qualititativa: refere-se a idade)

#**Payment** ( Categória- Qaualitativa: refere-se ao tipo de pagamento)

#**Total** (Numérica- Quantitativa : rerefere-se ao total da transação.)

#**Profit** (Numerica Quantitativa: referese ao lucro.)

#**profile** (Categórica -Qualitativa: refere-se ao tipo de perfil do cliente)

#**Unity** ( Numérica Quantitativa: refere-se ao valor unitário.

#**Genre** (Categórica Qualitativa: refere-se ao genero, masculino ou feminino.
```

ETAPA 2: Data Exploration

Atividade (A1): Definir / Eliminar (Dropar) colunas NÃO relevantes ao Projeto.

In [24]:
```
display(dataset)
```

	TransactionNo	Data	ProductNo	ProductName	CustomerNo	Country	Cost	Unity	Quantity
0	581482	2019-09-12 00:00:00	22485	Set Of 2 Wooden Market Crates	17490.0	United Kingdom	0.43	1.79	12
1	581475	2019-09-12 00:00:00	22596	Christmas Star Wish List Chalkboard	13069.0	United Kingdom	0.19	0.30	36
2	581475	2019-09-12 00:00:00	23235	Storage Tin Vintage Leaf	13069.0	United Kingdom	0.50	0.96	12
3	581475	2019-09-12 00:00:00	23272	Tree T-Light Holder Willie Winkie	13069.0	United Kingdom	0.08	0.89	12
4	581475	2019-09-12 00:00:00	23239	Set Of 4 Knick Knack Tins Poppies	13069.0	United Kingdom	0.17	1.99	6
...	...	...	...	...	...	...	...	...	..
49993	577855	11/22/2019	84828	Jungle Popsicles Ice Lolly Moulds	15877.0	United Kingdom	0.74	6.13	1
49994	577855	11/22/2019	22262	Felt Egg Cosy Chicken	15877.0	United Kingdom	0.78	7.29	1
49995	577855	11/22/2019	21383	Pack Of 12 Sticky Bunnies	15877.0	United Kingdom	0.36	7.29	1
49996	577855	11/22/2019	22530	Magic Drawing Slate Dolly Girl	15877.0	United Kingdom	0.22	7.29	1
49997	577855	11/22/2019	23392	Spaceboy Rocket Lolly Makers	15877.0	United Kingdom	0.99	7.29	1

49998 rows × 16 columns

In [25]:
```
#DROPANDO AS COLUNAS MENOS IMPORTANTES
datasetLimpo=dataset.drop(['TransactionNo', 'ProductNo','CustomerNo', 'ProductName',
'Country'],axis=1)
```

In [26]:
```
datasetLimpo
```

Out[26]:

	Data	Cost	Unity	Quantity	Price	Genre	Age	Payment	Total	Profit	Profile
0	2019-09-12 00:00:00	0.43	1.79	12	21.47	Female	56	Debit	257.64	257.21	6
1	2019-09-12 00:00:00	0.19	0.30	36	10.65	Female	26	Credit	383.40	383.21	7
2	2019-09-12 00:00:00	0.50	0.96	12	11.53	Female	70	Debit	138.36	137.86	4
3	2019-09-12 00:00:00	0.08	0.89	12	10.65	Female	26	Debit	127.80	127.72	4
4	2019-09-12 00:00:00	0.17	1.99	6	11.94	Male	57	Debit	71.64	71.47	3
...	...	...	...	...	...	...	...	...	...	...	...
49993	11/22/2019	0.74	6.13	1	6.13	Male	54	Credit	6.13	5.39	1
49994	11/22/2019	0.78	7.29	1	7.29	Male	32	Credit	7.29	6.51	1
49995	11/22/2019	0.36	7.29	1	7.29	Female	29	Credit	7.29	6.93	1
49996	11/22/2019	0.22	7.29	1	7.29	Male	33	Credit	7.29	7.07	1
49997	11/22/2019	0.99	7.29	1	7.29	Female	70	Debit	7.29	6.30	1

49998 rows × 11 columns

Atividade (A2): Pesquisar / Eliminar (Dropar) os Valores duplicados (linhas duplicadas)

Para isso, faça:

1. Aplicar uma transformação com o método **STR.UPPER** (tudo maiúsculo) em todas as colunas Quanlitativas (str).
2. Contador a frequência dos valores únicos: método **UNIQUE**.
3. Aplicar o método **DUPLICATED** eliminando as linhas duplicadas do dataset.

In [27]:
```
#APLICANDO MÉTODO STR.UPPER (COLUNAS MAIÚSCULAS)
```

In [28]:
```
dataset["Country"]=dataset["Country"].str.upper()
dataset["ProductName"]=dataset["ProductName"].str.upper()
```

In [29]:
```
dataset
```

Out[29]:

	TransactionNo	Data	ProductNo	ProductName	CustomerNo	Country	Cost	Unity	Quant
0	581482	2019-09-12 00:00:00	22485	SET OF 2 WOODEN MARKET CRATES	17490.0	UNITED KINGDOM	0.43	1.79	
1	581475	2019-09-12 00:00:00	22596	CHRISTMAS STAR WISH LIST CHALKBOARD	13069.0	UNITED KINGDOM	0.19	0.30	
2	581475	2019-09-12 00:00:00	23235	STORAGE TIN VINTAGE LEAF	13069.0	UNITED KINGDOM	0.50	0.96	
3	581475	2019-09-12 00:00:00	23272	TREE T-LIGHT HOLDER WILLIE WINKIE	13069.0	UNITED KINGDOM	0.08	0.89	
4	581475	2019-09-12 00:00:00	23239	SET OF 4 KNICK KNACK TINS POPPIES	13069.0	UNITED KINGDOM	0.17	1.99	
...	...	...	...	...	...	...	...	...	
49993	577855	11/22/2019	84828	JUNGLE POPSICLES ICE LOLLY MOULDS	15877.0	UNITED KINGDOM	0.74	6.13	
49994	577855	11/22/2019	22262	FELT EGG COSY CHICKEN	15877.0	UNITED KINGDOM	0.78	7.29	
49995	577855	11/22/2019	21383	PACK OF 12 STICKY BUNNIES	15877.0	UNITED KINGDOM	0.36	7.29	
49996	577855	11/22/2019	22530	MAGIC DRAWING SLATE DOLLY GIRL	15877.0	UNITED KINGDOM	0.22	7.29	
49997	577855	11/22/2019	23392	SPACEBOY ROCKET LOLLY MAKERS	15877.0	UNITED KINGDOM	0.99	7.29	

49998 rows × 16 columns

In [30]:
```
#VERIFICANDO VALORES ÚNICOS
dataset.nunique()
```

Out[30]:
```
TransactionNo    1836
Data               16
ProductNo        2690
ProductName      2690
CustomerNo       1191
Country            23
Cost              101
Unity             292
Quantity          151
Price              98
Genre               2
Age                53
Payment             2
Total             823
Profit           6956
Profile             8
dtype: int64
```

In [31]:
```
#VERIFICANDO OS VALORES DUPLICADOS
dataset.duplicated()
```

Out[31]:
```
0        False
1        False
2        False
3        False
4        False
         ...
49993    False
49994    False
49995    False
49996    False
49997    False
Length: 49998, dtype: bool
```

In [32]:
```
#DROPANDO VALORE DUPLICADOS
datasetLimpo=dataset.drop_duplicates()
```

In [33]:
```
#VERIFICANDO SE HOUVE ALTERAÇÃO
datasetLimpo
```

Out[33]:

	TransactionNo	Data	ProductNo	ProductName	CustomerNo	Country	Cost	Unity	Quant
0	581482	2019-09-12 00:00:00	22485	SET OF 2 WOODEN MARKET CRATES	17490.0	UNITED KINGDOM	0.43	1.79	
1	581475	2019-09-12 00:00:00	22596	CHRISTMAS STAR WISH LIST CHALKBOARD	13069.0	UNITED KINGDOM	0.19	0.30	
2	581475	2019-09-12 00:00:00	23235	STORAGE TIN VINTAGE LEAF	13069.0	UNITED KINGDOM	0.50	0.96	
3	581475	2019-09-12 00:00:00	23272	TREE T-LIGHT HOLDER WILLIE WINKIE	13069.0	UNITED KINGDOM	0.08	0.89	
4	581475	2019-09-12 00:00:00	23239	SET OF 4 KNICK KNACK TINS POPPIES	13069.0	UNITED KINGDOM	0.17	1.99	
...	...	...	...	...	...	...	...	...	
49993	577855	11/22/2019	84828	JUNGLE POPSICLES ICE LOLLY MOULDS	15877.0	UNITED KINGDOM	0.74	6.13	
49994	577855	11/22/2019	22262	FELT EGG COSY CHICKEN	15877.0	UNITED KINGDOM	0.78	7.29	
49995	577855	11/22/2019	21383	PACK OF 12 STICKY BUNNIES	15877.0	UNITED KINGDOM	0.36	7.29	
49996	577855	11/22/2019	22530	MAGIC DRAWING SLATE DOLLY GIRL	15877.0	UNITED KINGDOM	0.22	7.29	
49997	577855	11/22/2019	23392	SPACEBOY ROCKET LOLLY MAKERS	15877.0	UNITED KINGDOM	0.99	7.29	

49998 rows × 16 columns

In [34]:
```
#NÃO HÁ VALORES DUPLICADOS, OS VALORES SÃO ÚNICOS.
```

Atividade (A3): Pesquisar / Eliminar (Dropar) os Valores de Sentinela (NaN: NotANumber)

In [35]:
```
#VERIFICANDO SE HÁ VALORES AUSENTES(NaN)
datasetLimpo.isna()
```

Out[35]:

	TransactionNo	Data	ProductNo	ProductName	CustomerNo	Country	Cost	Unity	Quantity	Pri
0	False	False	False	False	False	False	False	False	False	Fa
1	False	False	False	False	False	False	False	False	False	Fa
2	False	False	False	False	False	False	False	False	False	Fa
3	False	False	False	False	False	False	False	False	False	Fa
4	False	False	False	False	False	False	False	False	False	Fa
...	...	...	...	...	...	...	...	...	...	
49993	False	False	False	False	False	False	False	False	False	Fa
49994	False	False	False	False	False	False	False	False	False	Fa
49995	False	False	False	False	False	False	False	False	False	Fa
49996	False	False	False	False	False	False	False	False	False	Fa
49997	False	False	False	False	False	False	False	False	False	Fa

49998 rows × 16 columns

In [36]:
```
#FILTRANDO VALORES  (NA)
datasetLimpo=datasetLimpo.dropna()
```

In [37]:
```
datasetLimpo
```

Out[37]:

	TransactionNo	Data	ProductNo	ProductName	CustomerNo	Country	Cost	Unity	Quant
0	581482	2019-09-12 00:00:00	22485	SET OF 2 WOODEN MARKET CRATES	17490.0	UNITED KINGDOM	0.43	1.79	
1	581475	2019-09-12 00:00:00	22596	CHRISTMAS STAR WISH LIST CHALKBOARD	13069.0	UNITED KINGDOM	0.19	0.30	
2	581475	2019-09-12 00:00:00	23235	STORAGE TIN VINTAGE LEAF	13069.0	UNITED KINGDOM	0.50	0.96	
3	581475	2019-09-12 00:00:00	23272	TREE T-LIGHT HOLDER WILLIE WINKIE	13069.0	UNITED KINGDOM	0.08	0.89	
4	581475	2019-09-12 00:00:00	23239	SET OF 4 KNICK KNACK TINS POPPIES	13069.0	UNITED KINGDOM	0.17	1.99	
...	...	...	...	...	...	...	...	...	
49993	577855	11/22/2019	84828	JUNGLE POPSICLES ICE LOLLY MOULDS	15877.0	UNITED KINGDOM	0.74	6.13	
49994	577855	11/22/2019	22262	FELT EGG COSY CHICKEN	15877.0	UNITED KINGDOM	0.78	7.29	
49995	577855	11/22/2019	21383	PACK OF 12 STICKY BUNNIES	15877.0	UNITED KINGDOM	0.36	7.29	
49996	577855	11/22/2019	22530	MAGIC DRAWING SLATE DOLLY GIRL	15877.0	UNITED KINGDOM	0.22	7.29	
49997	577855	11/22/2019	23392	SPACEBOY ROCKET LOLLY MAKERS	15877.0	UNITED KINGDOM	0.99	7.29	

49996 rows × 16 columns

In [38]:
```
#CONCLUSÃO: UM VALOR  AUSENTES NA EXLUÍDO
```

In [39]:
```
#VERIFICANDO VALORES NULL ( NULOS)
datasetLimpo.isnull().sum()
```

Out[39]:
```
TransactionNo    0
Data             0
ProductNo        0
ProductName      0
CustomerNo       0
Country          0
Cost             0
Unity            0
Quantity         0
Price            0
Genre            0
Age              0
Payment          0
Total            0
Profit           0
Profile          0
dtype: int64
```

In [40]:
```
#NÃO HÁ VALORES DE SENTINELA PARA TRATAR.
```

ETAPA 3: Data Evoluation

ETAPA 3.1: Análise exploratória das Variáveis Qualitativas

Atividade (A1): Criar os Gráficos (Plotar)

In [41]:
```
datasetLimpo
```

Out[41]:

	TransactionNo	Data	ProductNo	ProductName	CustomerNo	Country	Cost	Unity	Quant
0	581482	2019-09-12 00:00:00	22485	SET OF 2 WOODEN MARKET CRATES	17490.0	UNITED KINGDOM	0.43	1.79	
1	581475	2019-09-12 00:00:00	22596	CHRISTMAS STAR WISH LIST CHALKBOARD	13069.0	UNITED KINGDOM	0.19	0.30	
2	581475	2019-09-12 00:00:00	23235	STORAGE TIN VINTAGE LEAF	13069.0	UNITED KINGDOM	0.50	0.96	
3	581475	2019-09-12 00:00:00	23272	TREE T-LIGHT HOLDER WILLIE WINKIE	13069.0	UNITED KINGDOM	0.08	0.89	
4	581475	2019-09-12 00:00:00	23239	SET OF 4 KNICK KNACK TINS POPPIES	13069.0	UNITED KINGDOM	0.17	1.99	
...	...	...	...	...	...	...	...	...	
49993	577855	11/22/2019	84828	JUNGLE POPSICLES ICE LOLLY MOULDS	15877.0	UNITED KINGDOM	0.74	6.13	
49994	577855	11/22/2019	22262	FELT EGG COSY CHICKEN	15877.0	UNITED KINGDOM	0.78	7.29	
49995	577855	11/22/2019	21383	PACK OF 12 STICKY BUNNIES	15877.0	UNITED KINGDOM	0.36	7.29	
49996	577855	11/22/2019	22530	MAGIC DRAWING SLATE DOLLY GIRL	15877.0	UNITED KINGDOM	0.22	7.29	
49997	577855	11/22/2019	23392	SPACEBOY ROCKET LOLLY MAKERS	15877.0	UNITED KINGDOM	0.99	7.29	

49996 rows × 16 columns

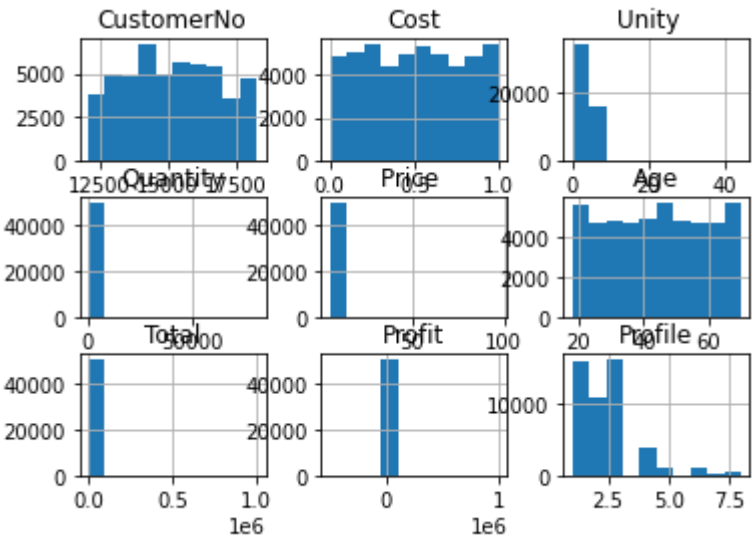
In [42]:
```
#TABELA ESTATÍSTICAS
datasetLimpo.describe()
```

Out[42]:

	CustomerNo	Cost	Unity	Quantity	Price	Age	Tota
count	49996.000000	49996.000000	49996.000000	49996.000000	49996.000000	49996.000000	4.999600e+0
mean	15160.610169	0.500249	3.121135	12.173054	6.527987	44.004920	8.819518e+0
std	1700.482296	0.291292	2.694891	516.160394	1.323589	15.246976	5.029240e+0
min	12067.000000	0.000000	0.000000	1.000000	5.130000	18.000000	5.130000e+0
25%	13777.000000	0.250000	0.720000	1.000000	6.190000	31.000000	7.240000e+0
50%	15152.000000	0.500000	2.060000	3.000000	6.190000	44.000000	1.857000e+0
75%	16566.750000	0.750000	6.190000	9.000000	6.390000	57.000000	6.040000e+0
max	18283.000000	1.000000	44.370000	80995.000000	97.370000	70.000000	1.002718e+0

In [43]:
```
#GRÁFICO DE HISTOGRAMA
datasetLimpo.hist()
```

Out[43]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f84fd2df210>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f84fce81790>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f84fda61d90>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f84fe5913d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f84fe1fc9d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f84fe03cb10>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f84fe8d11d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f84fd9edd50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f84fe98da90>]],
      dtype=object)
```

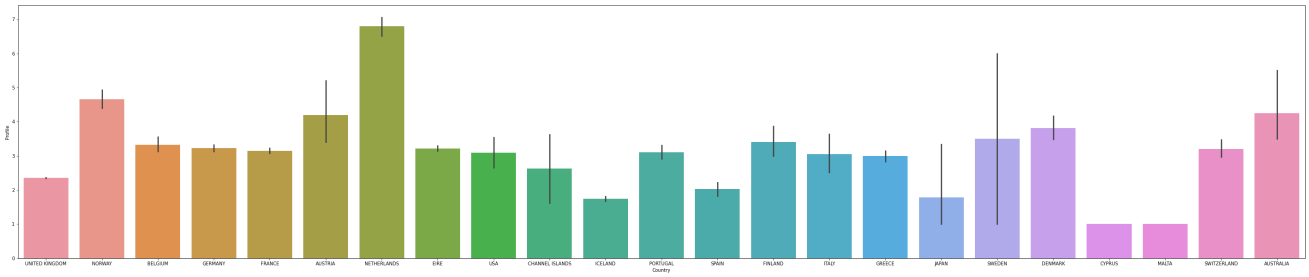


In [44]:
```
datasetLimpo.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 49996 entries, 0 to 49997
Data columns (total 16 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   TransactionNo  49996 non-null  object 
 1   Data           49996 non-null  object 
 2   ProductNo      49996 non-null  object 
 3   ProductName    49996 non-null  object 
 4   CustomerNo     49996 non-null  float64
 5   Country        49996 non-null  object 
 6   Cost           49996 non-null  float64
 7   Unity          49996 non-null  float64
 8   Quantity       49996 non-null  int64  
 9   Price          49996 non-null  float64
 10  Genre          49996 non-null  object 
 11  Age            49996 non-null  int64  
 12  Payment        49996 non-null  object 
 13  Total          49996 non-null  float64
 14  Profit         49996 non-null  float64
 15  Profile        49996 non-null  int64  
dtypes: float64(6), int64(3), object(7)
memory usage: 6.5+ MB
```

In [45]:
```
#ANÁLISE DE PERFIL  POR PÁIS EM  GRÁFICO de BARRAS
# CLIQUE NO GRAFICO PARA AUMENTAR O TAMANHO
import matplotlib.pyplot as plt
plt.figure(figsize = (50,10))
sns.barplot(data=datasetLimpo, x="Country", y="Profile" )
```

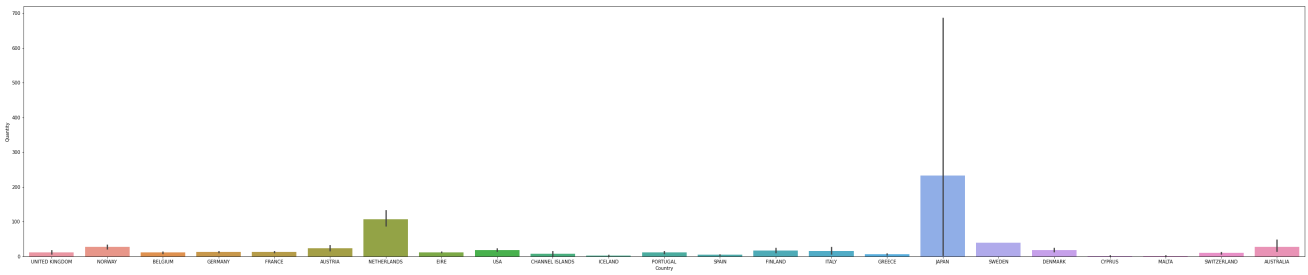
Out[45]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f84fd9ed390>
```



In [46]:
```
#CONCLUSÃO: NetherLands, Dinamarca e Australia  tem gasto maior no comércio eletrônic
o
```

In [47]:
```
#GRÁFICO DE BARRAS QUANTIDADE POR PAÍS
import matplotlib.pyplot as plt
plt.figure(figsize = (50,10))
sns.barplot(data=datasetLimpo, x="Country", y="Quantity" )
```

Out[47]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f84fd5adb50>
```

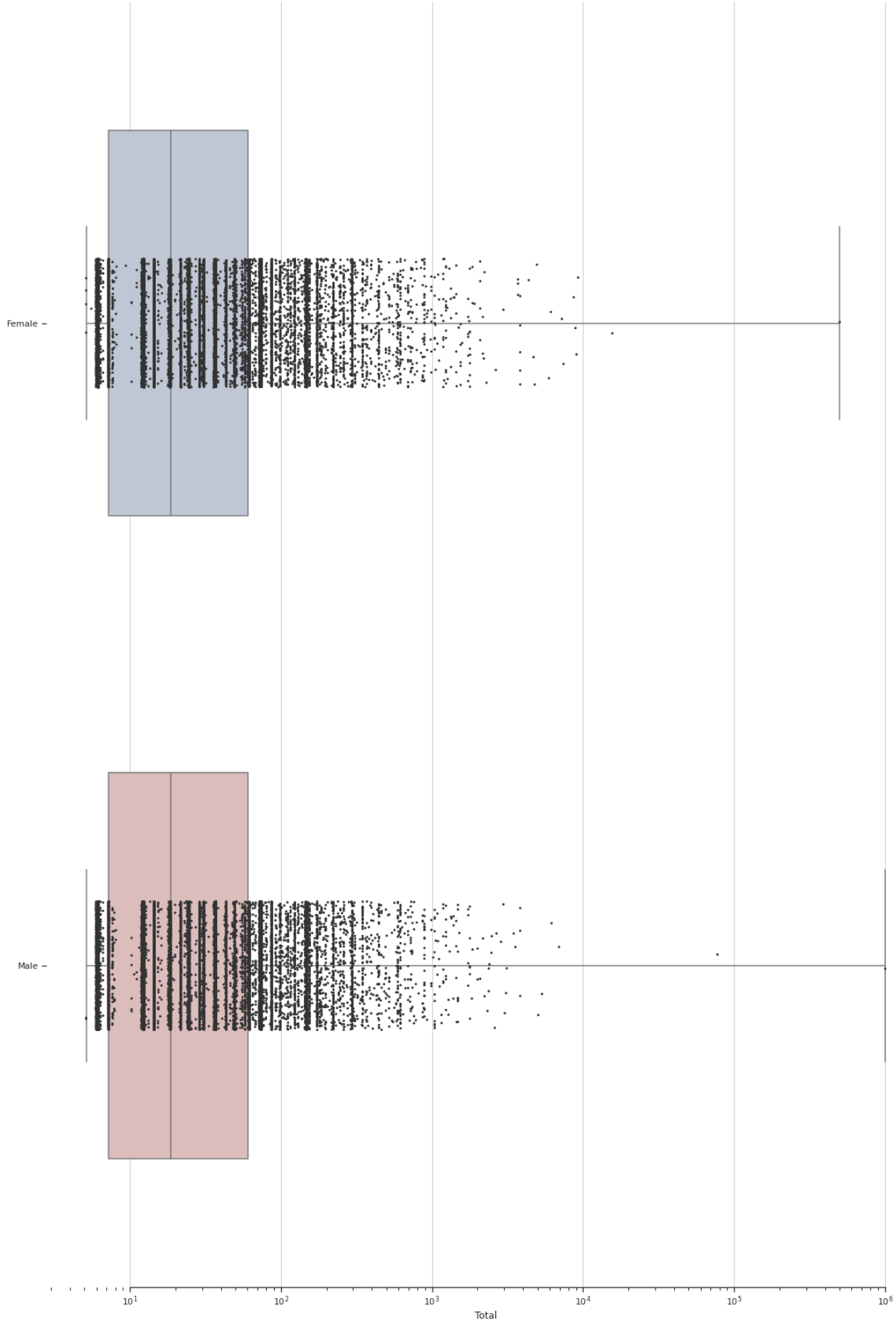


In [48]:
```
#CONCLUSÃO: NetherLands e Japão tem mais quantidade de transações no comércio eletrôn
ico
```

In [49]:
```
#GRÁFICO BOXPLOT DE COMPARAÇÃO DE TOTAL POR GENERO
```

In [50]:

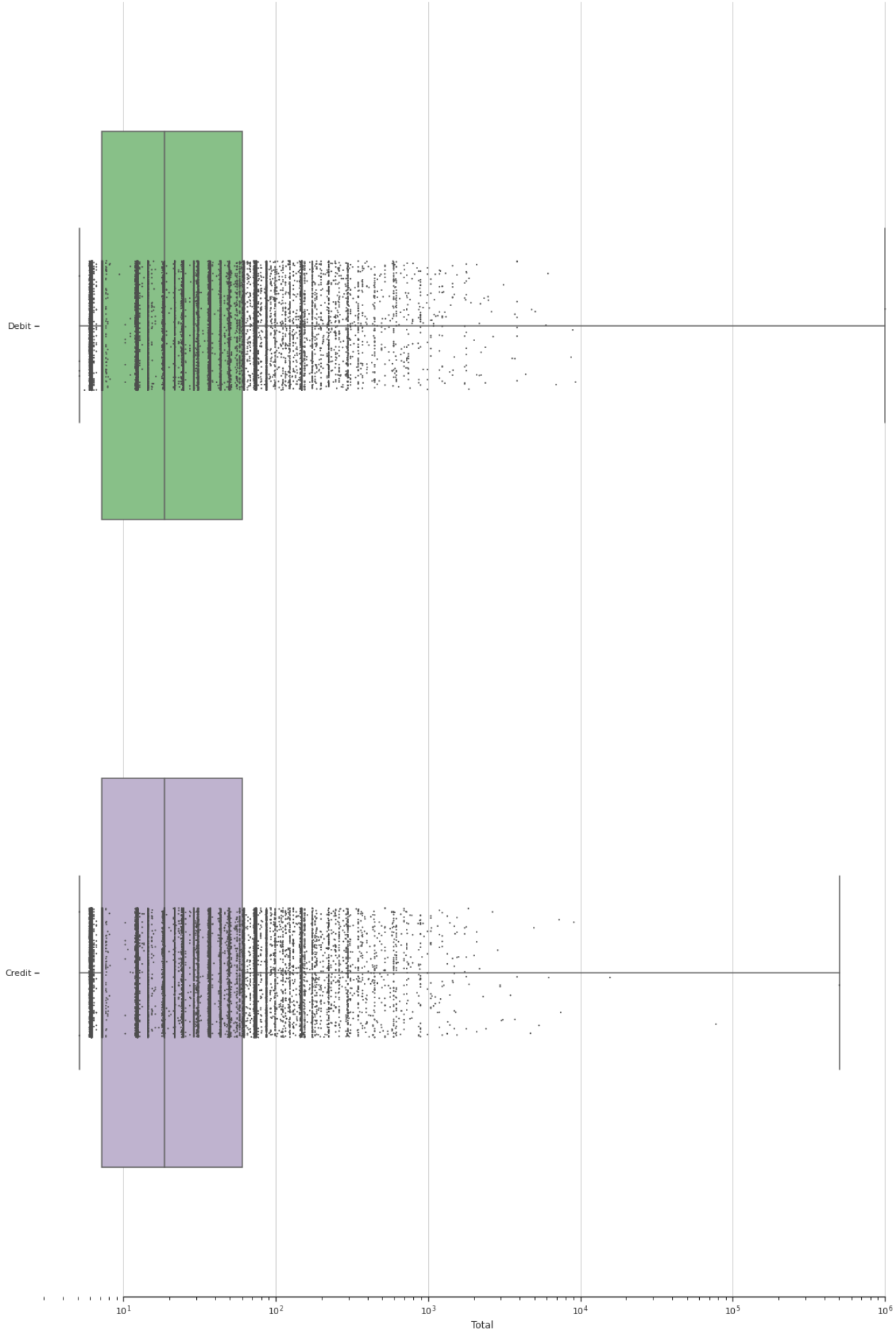
```
sns.set_theme(style="ticks")

# Iniciando figura com eixo logarítimico
f, ax = plt.subplots(figsize=(20, 30))
ax.set_xscale("log")


# Plotando comparação entre preço e País
sns.boxplot(x="Total", y="Genre", data=datasetLimpo,
            whis=[0, 100], width=.6, palette="vlag")

# Adicionando pontos de cada observação em cada amostra
sns.stripplot(x="Total", y="Genre", data=datasetLimpo,
              size=3, color=".2", linewidth=0)

# Tweak the visual presentation
ax.xaxis.grid(True)
ax.set(ylabel="")
sns.despine(trim=True, left=True)
```



In [51]:
```
#CONCLUSÁO: Os outliers acima no gráfico boxplot não devem ser considerados "lixos" p
ois
#demonstram apenas transações de preço e quantidade por país.
```

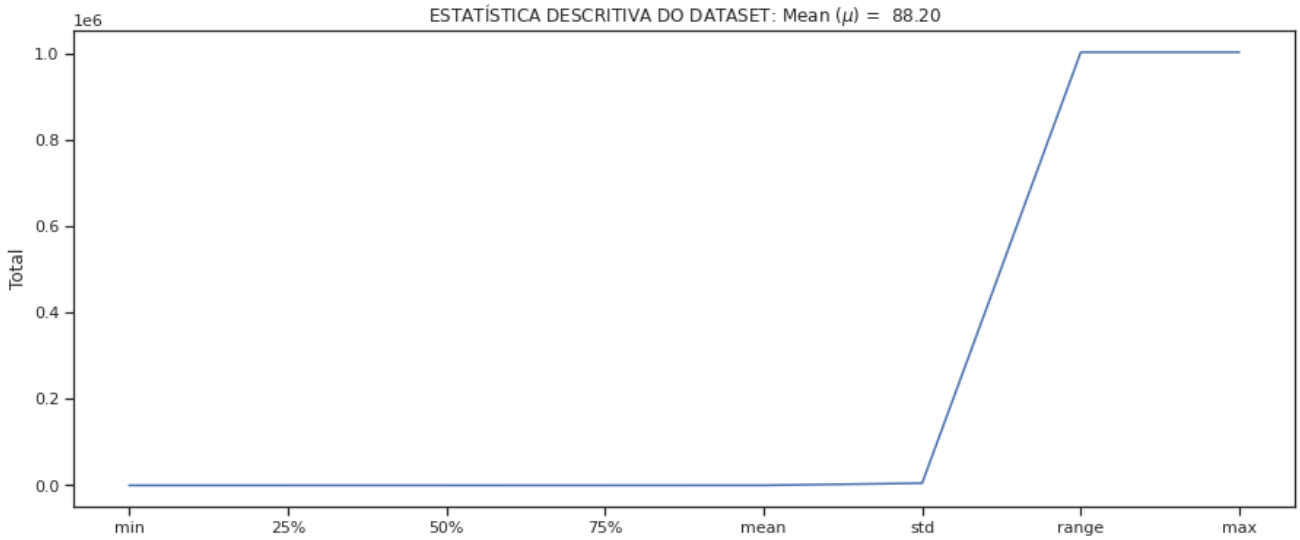
In [52]:

```
#Gráfico BoxPlot comparando TOTAL POR TIPO DE PAGAMENTO ( DEBITO OU CRÉDITO)

# Iniciando figura com eixo logarítimico
f, ax = plt.subplots(figsize=(20, 30))
ax.set_xscale("log")


# Plotando comparação entre preço e País
sns.boxplot(x="Total", y="Payment", data=datasetLimpo,
            whis=[0, 100], width=.6, palette="Accent")

# Adicionando pontos em cada observação
sns.stripplot(x="Total", y="Payment", data=datasetLimpo,
              size=2, color=".3", linewidth=0)

# Tweak the visual presentation
ax.xaxis.grid(True)
ax.set(ylabel="")
sns.despine(trim=True, left=True)
```



In [53]:
```
#CONCLUSÁO: Os outliers acima no gráfico boxplot não devem ser considerados "lixos" p
ois demonstram apenas trasações altas de quantidade por país.
```

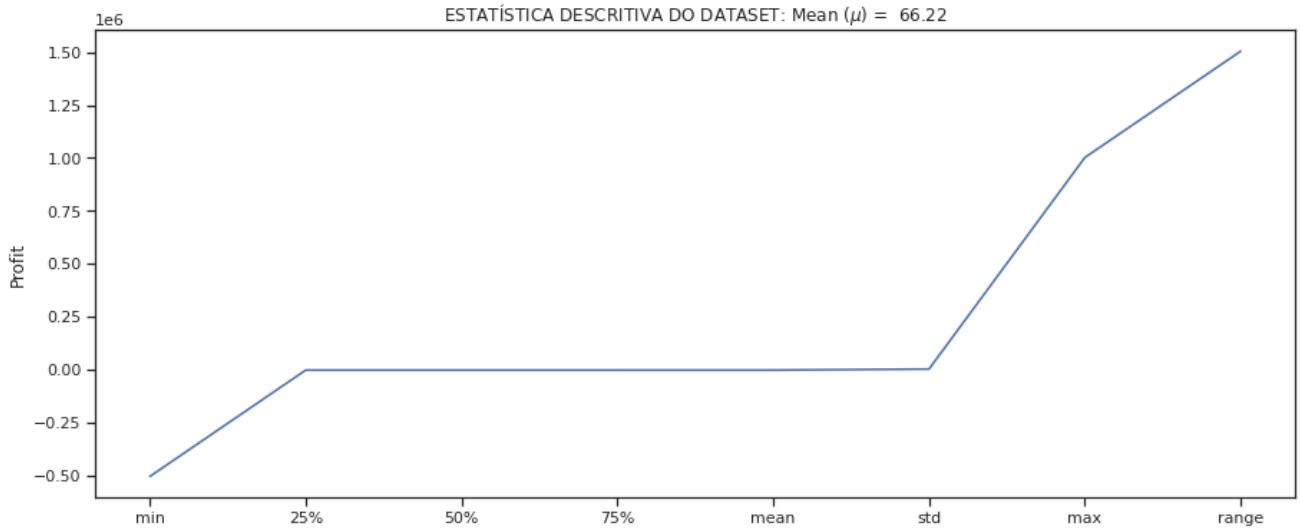
In [54]:
```
#ESTATÍSTICA DESCRITIVA da coluna TOTAL.
PlotarStatistic(datasetLimpo.Total, "Total")
```




```
ESTATÍSTICA DESCRITIVA DO DATASET:
min            5.13
25%            7.24
50%           18.57
75%           60.40
mean          88.20
std         5029.24
range    1002712.97
max      1002718.10
Name: Total, dtype: float64
```

In [54]:
```

```

In [55]:
```
#ESTATÍSITCAS descritiva da Coluna LUCRO
PlotarStatistic(datasetLimpo.Profit, "Profit")
```




```
ESTATÍSTICA DESCRITIVA DO DATASET:
min      -501359.12
25%           6.29
50%          17.84
75%          57.83
mean         66.22
std        5029.57
max     1002718.02
range   1504077.14
Name: Profit, dtype: float64
```

In [56]:
```
# PREPARANDO O DATASET PARA USAR COMO GRÁFICO DE SÉRIE
# EXCLUINDO COLUNAS IRRELEVANTES
```

In [57]:
```
datasetSerie=datasetLimpo.drop(['ProductNo', 'CustomerNo', 'TransactionNo', 'ProductName', 'Country', ], axis=1)
```

In [58]:
```
datasetSerie
```

Out[58]:

	Data	Cost	Unity	Quantity	Price	Genre	Age	Payment	Total	Profit	Profile
0	2019-09-12 00:00:00	0.43	1.79	12	21.47	Female	56	Debit	257.64	257.21	6
1	2019-09-12 00:00:00	0.19	0.30	36	10.65	Female	26	Credit	383.40	383.21	7
2	2019-09-12 00:00:00	0.50	0.96	12	11.53	Female	70	Debit	138.36	137.86	4
3	2019-09-12 00:00:00	0.08	0.89	12	10.65	Female	26	Debit	127.80	127.72	4
4	2019-09-12 00:00:00	0.17	1.99	6	11.94	Male	57	Debit	71.64	71.47	3
...	...	...	...	...	...	...	...	...	...	...	...
49993	11/22/2019	0.74	6.13	1	6.13	Male	54	Credit	6.13	5.39	1
49994	11/22/2019	0.78	7.29	1	7.29	Male	32	Credit	7.29	6.51	1
49995	11/22/2019	0.36	7.29	1	7.29	Female	29	Credit	7.29	6.93	1
49996	11/22/2019	0.22	7.29	1	7.29	Male	33	Credit	7.29	7.07	1
49997	11/22/2019	0.99	7.29	1	7.29	Female	70	Debit	7.29	6.30	1

49996 rows × 11 columns

In [59]:
```
#TRANSFORMANDO AS COLUNAS EM MAIUSCULAS ( modo inplace mantém maiúsculas)
datasetMaiuscula=datasetSerie.rename(columns={'Date': 'DATA','Price': 'PREÇO','Quantity':'QUANTITY','Unity':'UNITY','Genre':'GENERO','Age': 'IDADE', 'Payment':'PAGAMENTO','Total': 'TOTAL', 'Profit':'LUCRO',' Profile': 'PERFIL', 'Cost':'CUSTO UNITÁRIO'}, inplace=True)
```

In [60]:
```
datasetSerie
```

Out[60]:

	Data	CUSTO UNITÁRIO	Unity	QUANTITY	Price	GENERO	IDADE	PAGAMENTO	TOTAL	LUCRO
0	2019-09-12 00:00:00	0.43	1.79	12	21.47	Female	56	Debit	257.64	257.21
1	2019-09-12 00:00:00	0.19	0.30	36	10.65	Female	26	Credit	383.40	383.21
2	2019-09-12 00:00:00	0.50	0.96	12	11.53	Female	70	Debit	138.36	137.86
3	2019-09-12 00:00:00	0.08	0.89	12	10.65	Female	26	Debit	127.80	127.72
4	2019-09-12 00:00:00	0.17	1.99	6	11.94	Male	57	Debit	71.64	71.47
...	...	...	...	...	...	...	...	...	...	...
49993	11/22/2019	0.74	6.13	1	6.13	Male	54	Credit	6.13	5.39
49994	11/22/2019	0.78	7.29	1	7.29	Male	32	Credit	7.29	6.51
49995	11/22/2019	0.36	7.29	1	7.29	Female	29	Credit	7.29	6.93
49996	11/22/2019	0.22	7.29	1	7.29	Male	33	Credit	7.29	7.07
49997	11/22/2019	0.99	7.29	1	7.29	Female	70	Debit	7.29	6.30

49996 rows × 11 columns

In [61]:
```
datasetSerie.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 49996 entries, 0 to 49997
Data columns (total 11 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Data            49996 non-null  object 
 1   CUSTO UNITÁRIO  49996 non-null  float64
 2   Unity           49996 non-null  float64
 3   QUANTITY        49996 non-null  int64  
 4   Price           49996 non-null  float64
 5   GENERO          49996 non-null  object 
 6   IDADE           49996 non-null  int64  
 7   PAGAMENTO       49996 non-null  object 
 8   TOTAL           49996 non-null  float64
 9   LUCRO           49996 non-null  float64
 10  Profile         49996 non-null  int64  
dtypes: float64(5), int64(3), object(3)
memory usage: 5.6+ MB
```

In [62]:
```
#TRANSFORMANDO O DATASET EM DATAFRAME PARA PLOTAR O GRÁFICO
#graficoSerie= pd.DataFrame(datasetSerie)
```

In [63]:
```
grafico_serie=datasetSerie.set_index('Data')
```

In [64]:
```
#datasetSerie.Date=pd.to_datetime(datasetSerie.Date)
```

In [65]:
```
grafico_serie
```

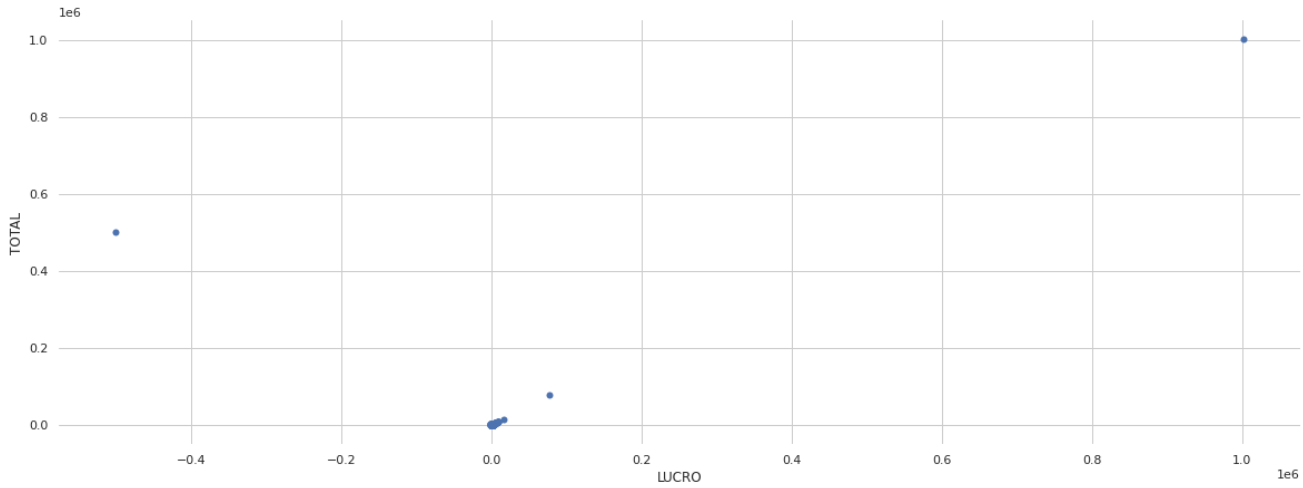
Out[65]:

Data	CUSTO UNITÁRIO	Unity	QUANTITY	Price	GENERO	IDADE	PAGAMENTO	TOTAL	LUCRO	Profile
2019-09-12 00:00:00	0.43	1.79	12	21.47	Female	56	Debit	257.64	257.21	6
2019-09-12 00:00:00	0.19	0.30	36	10.65	Female	26	Credit	383.40	383.21	7
2019-09-12 00:00:00	0.50	0.96	12	11.53	Female	70	Debit	138.36	137.86	4
2019-09-12 00:00:00	0.08	0.89	12	10.65	Female	26	Debit	127.80	127.72	4
2019-09-12 00:00:00	0.17	1.99	6	11.94	Male	57	Debit	71.64	71.47	3
...	...	...	...	...	...	...	...	...	...	...
11/22/2019	0.74	6.13	1	6.13	Male	54	Credit	6.13	5.39	1
11/22/2019	0.78	7.29	1	7.29	Male	32	Credit	7.29	6.51	1
11/22/2019	0.36	7.29	1	7.29	Female	29	Credit	7.29	6.93	1
11/22/2019	0.22	7.29	1	7.29	Male	33	Credit	7.29	7.07	1
11/22/2019	0.99	7.29	1	7.29	Female	70	Debit	7.29	6.30	1

49996 rows × 10 columns

In [66]:
```
#GRÁFICO DE DISPERSÃO
#CONCLUSÃO: existe correlação entre as variáveis LUCRO e TOTAL
sns.set_theme(style="whitegrid")
f, ax = plt.subplots(figsize=(20, 7))
sns.despine(f, left=True, bottom=True)
clarity_ranking = ["LUCRO", "TOTAL"]
sns.scatterplot(x="LUCRO", y="TOTAL",

                palette="ch:r=-.2,d=.3_r",
                hue_order=clarity_ranking,
                sizes=(1, 8), linewidth=0,
                data=grafico_serie, ax=ax)
```

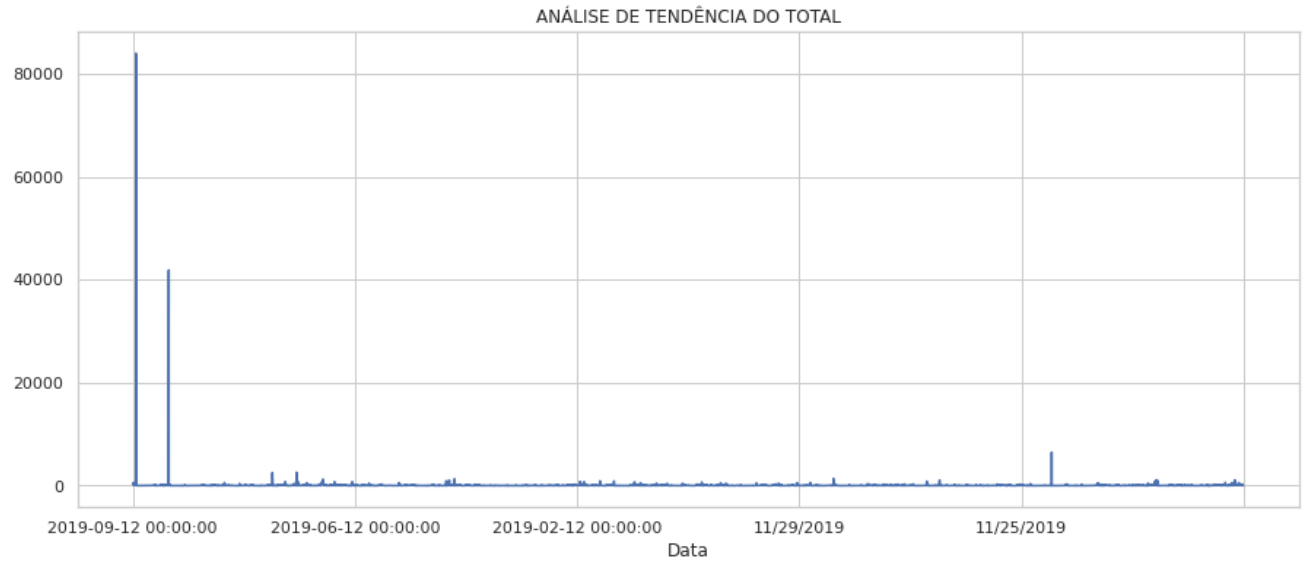
Out[66]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f84fd1ee8d0>
```



In [67]:
```
# GRÁFICO DE ANÁLISE DE TENDÊNCIA DA COLUNA TOTAL
tendenciaTOTAL = grafico_serie.TOTAL.rolling(12).mean()
tendenciaTOTAL
```

Out[67]:
```
Data
2019-09-12 00:00:00          NaN
2019-09-12 00:00:00          NaN
2019-09-12 00:00:00          NaN
2019-09-12 00:00:00          NaN
2019-09-12 00:00:00          NaN
                            ...
11/22/2019             10.727500
11/22/2019             10.824167
11/22/2019             10.920833
11/22/2019             11.017500
11/22/2019             10.092500
Name: TOTAL, Length: 49996, dtype: float64
```

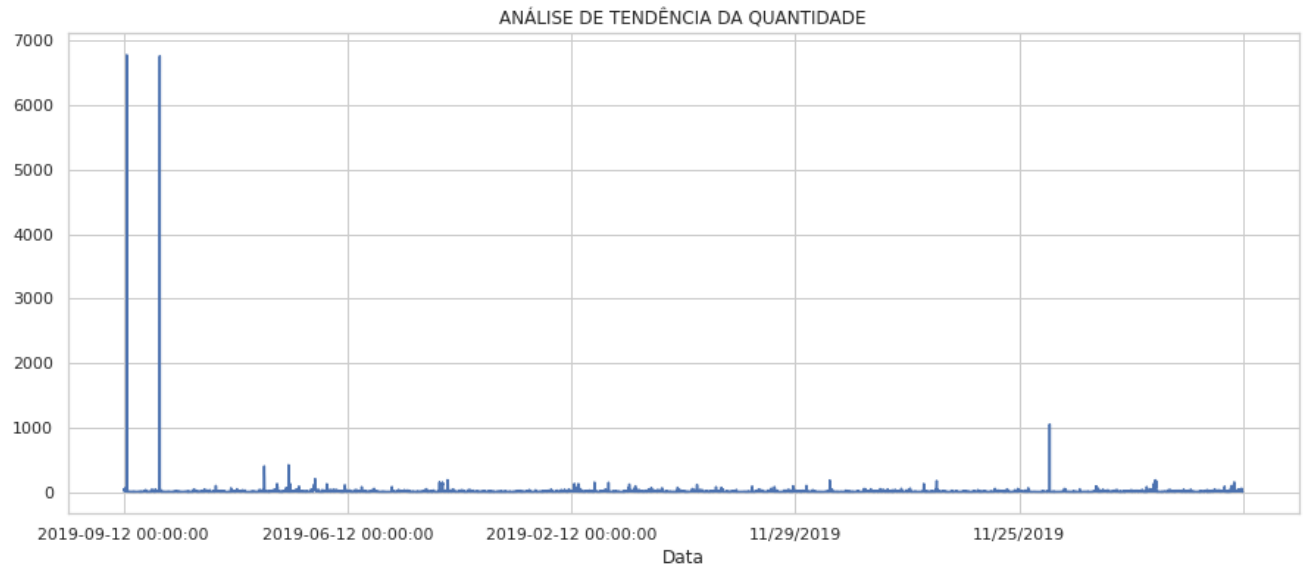
In [68]:
```
tendenciaTOTAL.plot(figsize = [15, 6])
plt.title('ANÁLISE DE TENDÊNCIA DO TOTAL')
plt.show()
```



In [69]:
```
# GRÁFICO DE ANÁLISE DE TENDÊNCIA DA COLUNA QUANTITY (QUANTIDADE)
tendenciaQuantity = grafico_serie.QUANTITY.rolling(12).mean()
tendenciaQuantity
```

Out[69]:
```
Data
2019-09-12 00:00:00         NaN
2019-09-12 00:00:00         NaN
2019-09-12 00:00:00         NaN
2019-09-12 00:00:00         NaN
2019-09-12 00:00:00         NaN
                            ...
11/22/2019             1.750000
11/22/2019             1.750000
11/22/2019             1.750000
11/22/2019             1.750000
11/22/2019             1.583333
Name: QUANTITY, Length: 49996, dtype: float64
```

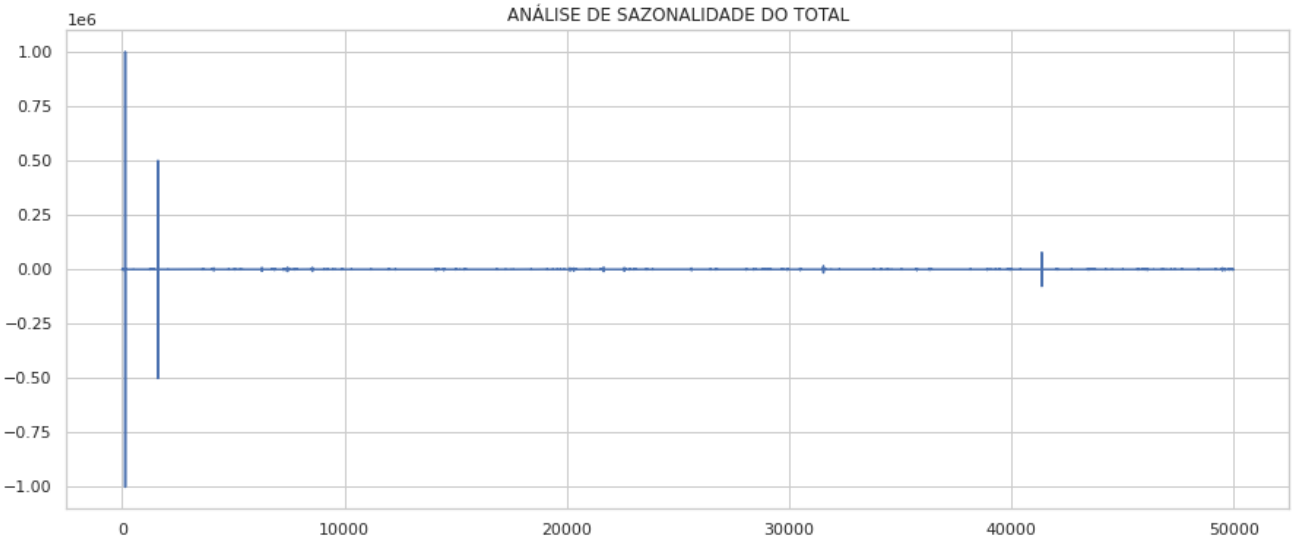
In [70]:
```
tendenciaQuantity.plot(figsize = [15, 6])
plt.title('ANÁLISE DE TENDÊNCIA DA QUANTIDADE')
plt.show()
```



In [71]:
```
#ANÁLISE DE SAZONALIDADE DO TOTAL
sazonalidadeTOTAL = datasetSerie.TOTAL.diff()
sazonalidadeTOTAL
```

Out[71]:
```
0           NaN
1        125.76
2       -245.04
3        -10.56
4        -56.16
          ...
49993      0.00
49994      1.16
49995      0.00
49996      0.00
49997      0.00
Name: TOTAL, Length: 49996, dtype: float64
```

In [72]:
```
sazonalidadeTOTAL.plot(figsize = [15, 6])
plt.title('ANÁLISE DE SAZONALIDADE DO TOTAL')
plt.show()
```



In [73]:
```
#PlotarSerie(grafico_serie.PREÇO)
```

Atividade (A2): Criar as Tabelas: Tabulação Cruzada (Método crosstab):

→ Incluir uma tabela com valores RELATIVOS (%).

In [74]:
```
dataset
```

Out[74]:

	TransactionNo	Data	ProductNo	ProductName	CustomerNo	Country	Cost	Unity	Quant
0	581482	2019-09-12 00:00:00	22485	SET OF 2 WOODEN MARKET CRATES	17490.0	UNITED KINGDOM	0.43	1.79	
1	581475	2019-09-12 00:00:00	22596	CHRISTMAS STAR WISH LIST CHALKBOARD	13069.0	UNITED KINGDOM	0.19	0.30	
2	581475	2019-09-12 00:00:00	23235	STORAGE TIN VINTAGE LEAF	13069.0	UNITED KINGDOM	0.50	0.96	
3	581475	2019-09-12 00:00:00	23272	TREE T-LIGHT HOLDER WILLIE WINKIE	13069.0	UNITED KINGDOM	0.08	0.89	
4	581475	2019-09-12 00:00:00	23239	SET OF 4 KNICK KNACK TINS POPPIES	13069.0	UNITED KINGDOM	0.17	1.99	
...	...	...	...	...	...	...	...	...	
49993	577855	11/22/2019	84828	JUNGLE POPSICLES ICE LOLLY MOULDS	15877.0	UNITED KINGDOM	0.74	6.13	
49994	577855	11/22/2019	22262	FELT EGG COSY CHICKEN	15877.0	UNITED KINGDOM	0.78	7.29	
49995	577855	11/22/2019	21383	PACK OF 12 STICKY BUNNIES	15877.0	UNITED KINGDOM	0.36	7.29	
49996	577855	11/22/2019	22530	MAGIC DRAWING SLATE DOLLY GIRL	15877.0	UNITED KINGDOM	0.22	7.29	
49997	577855	11/22/2019	23392	SPACEBOY ROCKET LOLLY MAKERS	15877.0	UNITED KINGDOM	0.99	7.29	

49998 rows × 16 columns

In [75]:
```
#TABELA CROSSTAB DE TOTAL POR PAÍS
pd.crosstab(dataset['Country'], dataset['Total'], margins=True)
```

Out[75]:

Total	5.13	5.55	5.97	6.02	6.03	6.04	6.13	6.17	6.19	6.39	...	7428.0	8690.76	8913.6	9
Country															
AUSTRALIA	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
AUSTRIA	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
BELGIUM	0	0	0	1	0	1	0	0	4	0	...	0	0	0	
CHANNEL ISLANDS	0	0	0	0	0	0	0	0	3	0	...	0	0	0	
CYPRUS	0	0	0	0	0	0	0	0	2	0	...	0	0	0	
DENMARK	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
EIRE	0	0	0	0	0	0	2	0	7	0	...	0	0	0	
FINLAND	0	0	0	0	0	1	0	0	0	0	...	0	0	0	
FRANCE	0	0	0	0	0	2	6	0	11	0	...	0	0	0	
GERMANY	0	0	0	0	0	0	10	0	14	1	...	0	0	0	
GREECE	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
ICELAND	1	0	3	1	0	12	43	0	176	0	...	0	0	0	
ITALY	0	0	0	0	0	0	0	0	1	1	...	0	0	0	
JAPAN	0	0	0	0	0	0	0	0	2	0	...	0	0	0	
MALTA	0	0	1	0	0	3	2	0	0	0	...	0	0	0	
NETHERLANDS	0	0	0	0	0	0	0	0	0	0	...	0	0	1	
NORWAY	0	0	0	0	0	0	0	0	1	0	...	0	0	0	
PORTUGAL	0	0	0	0	0	2	0	0	8	0	...	0	0	0	
SPAIN	0	0	2	0	0	0	10	0	22	1	...	0	0	0	
SWEDEN	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
SWITZERLAND	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
UNITED KINGDOM	5	1	265	88	2	684	1748	2	8706	97	...	1	1	0	
USA	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
All	6	1	271	90	2	705	1821	2	8957	100	...	1	1	1	

24 rows × 824 columns

In [76]:
```
#TABELA CROSSTAB DE QUANTIDADE POR PAÍS
pd.crosstab(dataset['Country'], dataset['Quantity'], margins=True)
```

Out[76]:

Quantity / Country	1	2	3	4	5	6	7	8	9	10	...	1000	1120	1200	1404
AUSTRALIA	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0
AUSTRIA	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0
BELGIUM	6	7	2	13	0	11	0	5	0	10	...	0	0	0	0
CHANNEL ISLANDS	3	0	0	1	0	2	0	0	0	0	...	0	0	0	0
CYPRUS	2	2	0	0	0	0	0	0	0	0	...	0	0	0	0
DENMARK	0	1	0	0	0	2	0	1	0	0	...	0	0	0	0
EIRE	10	61	19	40	2	79	0	41	4	25	...	0	0	0	0
FINLAND	2	3	4	1	0	5	0	2	0	5	...	0	0	0	0
FRANCE	20	57	33	67	9	101	2	52	1	50	...	0	0	0	0
GERMANY	31	44	34	53	4	73	0	39	1	46	...	0	0	0	0
GREECE	0	3	3	10	0	7	0	0	0	4	...	0	0	0	0
ICELAND	312	127	63	28	20	14	9	9	3	8	...	0	0	0	0
ITALY	4	6	3	8	0	4	0	0	3	1	...	0	0	0	0
JAPAN	2	1	2	0	0	0	1	0	0	0	...	0	0	0	0
MALTA	6	2	2	0	0	0	0	0	0	0	...	0	0	0	0
NETHERLANDS	0	0	0	2	0	0	0	0	0	2	...	0	0	0	0
NORWAY	1	11	0	11	0	9	0	19	2	5	...	0	0	0	0
PORTUGAL	23	12	5	8	7	13	1	6	1	15	...	0	0	0	0
SPAIN	43	27	0	8	3	8	0	0	1	6	...	0	0	0	0
SWEDEN	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0
SWITZERLAND	0	3	1	6	0	8	0	1	0	1	...	0	0	0	0
UNITED KINGDOM	15318	7780	3673	3294	1240	2783	385	963	244	1546	...	2	1	2	1
USA	0	0	0	0	0	0	0	4	0	4	...	0	0	0	0
All	15783	8147	3844	3550	1285	3119	398	1142	260	1728	...	2	1	2	1

24 rows × 152 columns

In [77]:
```
#TABELA PIVOT TABLE DE PREÇO POR PAÍS
dataset.pivot_table(['Total'], index=['Country'],
                    aggfunc=np.max)
```

Out[77]:

Country	Total
AUSTRALIA	594.24
AUSTRIA	222.84
BELGIUM	173.76
CHANNEL ISLANDS	173.76
CYPRUS	12.38
DENMARK	434.40
EIRE	699.84
FINLAND	445.68
FRANCE	1547.50
GERMANY	1782.72
GREECE	86.88
ICELAND	318.56
ITALY	891.36
JAPAN	15646.80
MALTA	18.39
NETHERLANDS	8913.60
NORWAY	2488.80
PORTUGAL	868.80
SPAIN	181.00
SWEDEN	289.60
SWITZERLAND	260.64
UNITED KINGDOM	1002718.10
USA	441.36

In [78]:
```
#TABELA PIVOT TABLE DE QUANTIDADE de VENDAS POR PAÍS
dataset.pivot_table(['Quantity'], index=['Country'],
                    aggfunc=np.max)
```

Out[78]:

Country	Quantity
AUSTRALIA	96
AUSTRIA	36
BELGIUM	25
CHANNEL ISLANDS	24
CYPRUS	2
DENMARK	60
EIRE	96
FINLAND	72
FRANCE	250
GERMANY	288
GREECE	12
ICELAND	44
ITALY	144
JAPAN	2040
MALTA	3
NETHERLANDS	1440
NORWAY	240
PORTUGAL	120
SPAIN	25
SWEDEN	40
SWITZERLAND	36
UNITED KINGDOM	80995
USA	72

In [79]:
```
#COMBINAÇÃO METODO APPLY: SEPARAR- APLICAR E COMBINAR OS DADOS
```

In [80]:
```
def top(datasetLimpo, n=10, column='Total'):
    return datasetLimpo.sort_values(by=column)[-n:]
```

In [81]:
```
dataset.groupby([dataset['Country'], dataset['Total']]).apply(top)
```

Out[81]:

Country	Total	TransactionNo	Data	ProductNo	ProductName	CustomerNo	Count
AUSTRALIA	**74.28**	**43778**	578459	11/24/2019	22580	ADVENT CALENDAR GINGHAM SACK	12388.0	AUSTRAL
		43779	578459	11/24/2019	22669	RED BABY BUNTING	12388.0	AUSTRAL
	123.80	**43773**	578459	11/24/2019	22600	CHRISTMAS RETROSPOT STAR WOOD	12388.0	AUSTRAL
		43774	578459	11/24/2019	22910	PAPER CHAIN KIT VINTAGE CHRISTMAS	12388.0	AUSTRAL
		43775	578459	11/24/2019	22086	PAPER CHAIN KIT 50'S CHRISTMAS	12388.0	AUSTRAL
...	**...**	**...**	...	...	...	...	...	
USA	**148.56**	**19969**	580158	2019-02-12 00:00:00	84828	JUNGLE POPSICLES ICE LOLLY MOULDS	12558.0	US
	222.84	**6484**	C581229	2019-08-12 00:00:00	23158	SET OF 5 LUCKY CAT MAGNETS	12558.0	US
		19970	580158	2019-02-12 00:00:00	23158	SET OF 5 LUCKY CAT MAGNETS	12558.0	US
	441.36	**12541**	580553	2019-05-12 00:00:00	23366	SET 12 COLOURING PENCILS DOILY	12646.0	US
		12549	580553	2019-05-12 00:00:00	20975	12 PENCILS SMALL TUBE RED RETROSPOT	12646.0	US

5145 rows × 16 columns

In [82]:
```
dataset.groupby([dataset['Country'], dataset['Quantity']],
                group_keys=False).apply(top, n = 1)
```

Out[82]:

	TransactionNo	Data	ProductNo	ProductName	CustomerNo	Country	Cost	Unity	Quar
43779	578459	11/24/2019	22669	RED BABY BUNTING	12388.0	AUSTRALIA	0.63	0.52	
43777	578459	11/24/2019	22576	SWALLOW WOODEN CHRISTMAS DECORATION	12388.0	AUSTRALIA	0.29	0.31	
43776	578459	11/24/2019	22340	NOEL GARLAND PAINTED ZINC	12388.0	AUSTRALIA	0.70	0.26	
43772	578459	11/24/2019	22338	STAR DECORATION PAINTED ZINC	12388.0	AUSTRALIA	0.03	0.06	
3227	581232	2019-08-12 00:00:00	15056P	EDWARDIAN PARASOL PINK	12358.0	AUSTRIA	0.71	0.52	
...	...	...	...	...	...	...	...	...	
12548	580553	2019-05-12 00:00:00	20676	RED RETROSPOT BOWL	12646.0	USA	0.39	0.39	
19976	580158	2019-02-12 00:00:00	23209	LUNCH BAG VINTAGE DOILY	12558.0	USA	0.99	0.31	
19969	580158	2019-02-12 00:00:00	84828	JUNGLE POPSICLES ICE LOLLY MOULDS	12558.0	USA	0.90	0.26	
19970	580158	2019-02-12 00:00:00	23158	SET OF 5 LUCKY CAT MAGNETS	12558.0	USA	0.19	0.17	
12549	580553	2019-05-12 00:00:00	20975	12 PENCILS SMALL TUBE RED RETROSPOT	12646.0	USA	0.66	0.09	

434 rows × 16 columns

ETAPA 3.2: Análise exploratória das Variáveis Quantitativas

Atividade (A1): Relatório e Análise dos parâmetros de Localização / Posição:

- **Tendência central**: Média / $Moda^*$ / Mediana
- **Separatriz:** Quartis (Q1 - Q2 - Q3)
- **Análise dos Outliers**: Limite inferior e superior

In [83]:
```
#TABELA DE TENDENCIA CENTRAL  e SEPARATRIZ
```

In [84]:
```
datasetLimpo.describe()
```

Out[84]:

	CustomerNo	Cost	Unity	Quantity	Price	Age	Tota
count	49996.000000	49996.000000	49996.000000	49996.000000	49996.000000	49996.000000	4.999600e+0
mean	15160.610169	0.500249	3.121135	12.173054	6.527987	44.004920	8.819518e+0
std	1700.482296	0.291292	2.694891	516.160394	1.323589	15.246976	5.029240e+0
min	12067.000000	0.000000	0.000000	1.000000	5.130000	18.000000	5.130000e+0
25%	13777.000000	0.250000	0.720000	1.000000	6.190000	31.000000	7.240000e+0
50%	15152.000000	0.500000	2.060000	3.000000	6.190000	44.000000	1.857000e+0
75%	16566.750000	0.750000	6.190000	9.000000	6.390000	57.000000	6.040000e+0
max	18283.000000	1.000000	44.370000	80995.000000	97.370000	70.000000	1.002718e+0

In [85]:
```
#ANÁLISE DOS OUTLIERS (Quartis Q1, Q2, Q3) da COLUNA "Price"
```

In [86]:
```
#OBS: Gráfico  interativo (passe o cursos sobre o gráfico para ver os Quartis, limite
s e outliers)
#OBS-2 : Por ter a tabela mais de 500 mil amostras o gráfico boxplot  ficou  achatado
```

In [87]:
```
import plotly.express as px
df = datasetLimpo
fig = px.box(df, x="Total" )
fig.show()
```

In [88]:
```
#ANÁLISE DOS OUTLIERS (Quartis Q1, Q2, Q3) da COLUNA "Quantity"
```

In [89]:
```
#OBS: Gráfico  interativo (passe o cursos sobre o gráfico para ver os Quartis, limite
s e outliers)
#OBS-2 : Por ter a tabela mais de 500 mil amostras o gráfico boxplot  ficou  achatado
```

In [90]:
```
import plotly.express as px
df = datasetLimpo
fig = px.box(df, x="Quantity" )
fig.show()
```

Atividade (A2): Relatório e Análise dos parâmetros de Variabilidade / Disperção:

- **Amplitude**
- **Desvio-padrão**
- **Coeficiente de Variação (CV)**: Pesquisar / Implementar a Função (DEF).
- **Erro Padrão**: Pesquisar / Implementar a Função (DEF).

In [91]:
```
#AMPLITUDE e DESVIO PADRÃO ( ver tabela abaixo)
#A Amplitude é calculada somente sobre as variáveis Quantitativas:
```

In [92]:
```
datasetLimpo.describe()
```

Out[92]:

	CustomerNo	Cost	Unity	Quantity	Price	Age	Tota
count	49996.000000	49996.000000	49996.000000	49996.000000	49996.000000	49996.000000	4.999600e+0
mean	15160.610169	0.500249	3.121135	12.173054	6.527987	44.004920	8.819518e+0
std	1700.482296	0.291292	2.694891	516.160394	1.323589	15.246976	5.029240e+0
min	12067.000000	0.000000	0.000000	1.000000	5.130000	18.000000	5.130000e+0
25%	13777.000000	0.250000	0.720000	1.000000	6.190000	31.000000	7.240000e+0
50%	15152.000000	0.500000	2.060000	3.000000	6.190000	44.000000	1.857000e+0
75%	16566.750000	0.750000	6.190000	9.000000	6.390000	57.000000	6.040000e+0
max	18283.000000	1.000000	44.370000	80995.000000	97.370000	70.000000	1.002718e+0

In [93]:
```
# COEFICIENTE DE VARIAÇÃO DA COLUNA PRICE
```

In [94]:
```
CV = grafico_serie.TOTAL.std() / grafico_serie.TOTAL.mean() * 100
print(f'Valor do Coeficiente de Variação da coluna: TOTAL = {CV: .2f} %')
```

```
Valor do Coeficiente de Variação da coluna: TOTAL =  5702.40 %
```

In [95]:
```
##COEFICIENTE DE VARIAÇÃO DA COLUNA QUANTITY
```

In [96]:
```
CV = grafico_serie.QUANTITY.std() / grafico_serie.QUANTITY.mean() * 100
print(f'Valor do Coeficiente de Variação da coluna: Quantity = {CV: .2f} %')
```

```
Valor do Coeficiente de Variação da coluna: Quantity =  4240.19 %
```

Atividade (A3): Relatório e Análise dos parâmetros de Forma:

- **Assimetria**: Pesquisar / Implementar a Função (DEF).
- **curtuse**: Pesquisar / Implementar a Função (DEF).

O coeficiente de assimetria, também chamado de índice de assimetria de pearson, é utilizado para calcular a assimetria das distribuições, ou seja, sintetiza até que ponto uma distribuição é enviesada, deformada ou assimétrica. Uma das fórmulas que pode ser utilizada para calcular o valor é (MAHALUÇA, 2016):

$$Ca = \frac{3*(\bar{x} - \tilde{x})}{s}$$

Onde, $\bar{x}$ é a media, $\tilde{x}$ mediana e S é o desvio padrão

In [97]:
```
# ASSIMETRIA da coluna PRICE

CA = 3 * (grafico_serie.TOTAL.mean() - grafico_serie.TOTAL.median()) / grafico_serie.
TOTAL.std()
print(f'Valor do Coeficiente de Assimetria da coluna: PRICE = {CA: .2f}')
```

```
Valor do Coeficiente de Assimetria da coluna: PRICE =  0.04
```

In [98]:
```
#ASSIMETRIA da coluna QUANTITY

CA = 3 * (grafico_serie.QUANTITY.mean() - grafico_serie.QUANTITY.median()) / grafico_
serie.QUANTITY.std()
print(f'Valor do Coeficiente de Assimetria da coluna: QUANTITY = {CA: .2f}')
```

```
Valor do Coeficiente de Assimetria da coluna: QUANTITY =  0.05
```

Estes valores são utilizados para ter conhecimento geral da distribuição, a maioria tem um índice entre -3 e 3.

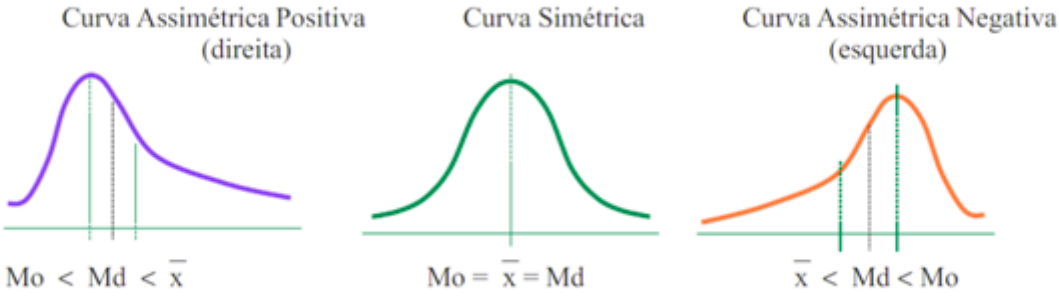
Portanto:

CA > 0 dizemos que os dados são assimétricos à direita onde (moda <= mediana <= media).

CA < 0 os dados são assimétricos à esquerda onde (media <= mediana <= moda) e

CA = 0 os dados são ditos simétricos onde (media = mediana = moda)

(MAHALUÇA, 2016).



Interpretação sobre o CA:

- 0 < Ca < 0.15, é dita assimetria fraca;
- |0.15| < Ca < |1|, é dita assimetria moderada;
- E, Ca >= |1|, assimetria é forte.

Curtose

A curtose mede o grau de achatamento ou afunilamento de uma curva simétrica, ou aproximadamente simétrica (MOREIRA, 2018):

$$K = \frac{1}{n}\sum\left[\frac{x_i - \bar{x}}{s}\right]^4 - 3$$

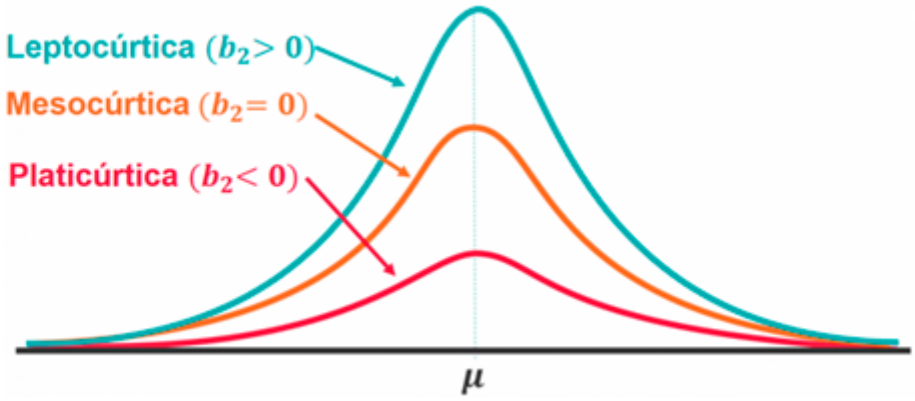
Onde, $\bar{x}$ é a média, **s** é o desvio padrão.

O resultado de K pode nos indicar:

K = 0, **mesocúrtica**, quando os dados estão razoavelmente concentrados em torno do seu centro, também é chamada de curva padrão (MOREIRA, 2018);

K > 0, **leptocúrtica**, quando a curva de frequência está bastante fechada, fortemente concentrada em torno da moda, também possui caudas grossas devido à grande amplitude (MOREIRA, 2018);

K < 0, **platicúrtica**, quando a curva de frequência está mais aberta, com topo achatado, em que várias classes apresentam frequências quase iguais (MOREIRA, 2018).



In [99]:
```
K = grafico_serie.TOTAL.kurtosis()
print(f'Valor do Coeficiente de Curtose da coluna: TOTAL = {K: .2f}')
```

```
Valor do Coeficiente de Curtose da coluna: TOTAL =  33571.80
```

In [100]:
```
#CURTOSE  da culuna QUANTITY
```

In [101]:
```
K = grafico_serie.QUANTITY.kurtosis()
print(f'Valor do Coeficiente de Curtose da coluna: QUANTITY = {K: .2f}')
```

```
Valor do Coeficiente de Curtose da coluna: QUANTITY =  24247.08
```

ETAPA 3.3: Aplicação e Análise por Métodos Quantitativos

DICAS: Como fazer ETAPAS 3.3 e 3.4: Link (https://colab.research.google.com/drive/1XA2hU9mioENk_Qnd06Sp0yMQsAFdBT-a?usp=sharing)

AMOSTRAGEM ALEATÓRIA: Método sample

Criar a Amostragem Aleatória (sem reposição: replace=False) de **3 conjuntos IGUAIS (LENGHT / 3)**:

- **Treinamento** (train):.
- **Teste** (test):.
- **Validação** (validation):.

E repetir esse processo **10 vezes (SPLIT)**, para testar a **VALIDAÇÃO CRUZADA** dos seus resultados.

A partir destas amostras, faça:

In [102]:
```
#Criando a amostragem aelatória 1
```

In [103]:
```
sample1_random = dataset.sample(n=1022, replace=False)
sample1_random.head()
```

Out[103]:

	TransactionNo	Data	ProductNo	ProductName	CustomerNo	Country	Cost	Unity	Quanti
26216	579571	11/30/2019	10120	DOGGY RUBBER	16657.0	UNITED KINGDOM	0.40	1.81	
16877	580729	2019-05-12 00:00:00	20728	LUNCH BAG CARS BLUE	16729.0	UNITED KINGDOM	0.64	5.97	
10484	580904	2019-06-12 00:00:00	23317	BLUE REFECTORY CLOCK	15311.0	UNITED KINGDOM	0.83	6.19	
29199	579291	11/29/2019	22737	RIBBON REEL CHRISTMAS PRESENT	14291.0	UNITED KINGDOM	0.76	0.77	
6375	581469	2019-08-12 00:00:00	22320	BIRDS MOBILE VINTAGE DESIGN	14606.0	UNITED KINGDOM	0.93	6.19	

In [104]:
```
#Criando a amostragem aleatóra 2
```

In [105]:
```
sample2_random = dataset.sample(n=1022, replace=False)
sample2_random.head()
```

Out[105]:

	TransactionNo	Data	ProductNo	ProductName	CustomerNo	Country	Cost	Unity	C
16796	580729	2019-05-12 00:00:00	85179C	PINK BITTY LIGHT CHAIN	16729.0	UNITED KINGDOM	0.75	7.29	
8536	581175	2019-07-12 00:00:00	23390	DOLLY GIRL MINI BACKPACK	14646.0	NETHERLANDS	0.01	0.14	
6761	581014	2019-07-12 00:00:00	23581	JUMBO BAG PAISLEY PARK	16458.0	UNITED KINGDOM	0.12	0.31	
12782	580610	2019-05-12 00:00:00	21935	SUKI SHOULDER BAG	12610.0	UNITED KINGDOM	0.44	1.02	
32496	579512	11/29/2019	22617	BAKING SET SPACEBOY DESIGN	14512.0	UNITED KINGDOM	0.97	7.24	

In [106]:
```
#Criando a amostragem aleatória 3
```

In [107]:
```
sample3_random = dataset.sample(n=1022, replace=False)
sample3_random.head()
```

Out[107]:

	TransactionNo	Data	ProductNo	ProductName	CustomerNo	Country	Cost	Unity	Quant
11506	580983	2019-06-12 00:00:00	21813	GARLAND WITH STARS AND BELLS	16983.0	UNITED KINGDOM	0.75	0.62	
41047	578829	11/25/2019	23568	EGG CUP HENRIETTA HEN CREAM	14710.0	UNITED KINGDOM	0.33	6.19	
218	581492	2019-09-12 00:00:00	23571	TRADITIONAL NAUGHTS & CROSSES	15492.0	UNITED KINGDOM	0.66	3.10	
26573	579677	11/30/2019	22839	3 TIER CAKE TIN GREEN AND CREAM	13255.0	UNITED KINGDOM	0.59	6.19	
27555	579777	11/30/2019	47591D	PINK FAIRY CAKE CHILDRENS APRON	13777.0	UNITED KINGDOM	0.43	6.19	

In [108]:
```
#TREINAMENTO DA AMOSTRAGEM ALEATÓRIA
```

In [109]:
```
train = dataset.sample(len(dataset)//3, replace=False)
```

In [110]:
```
dataset = dataset.drop(train.index)
```

In [111]:
```
test = dataset.sample(len(dataset)//2, replace=False)
validation = dataset.drop(test.index)
```

In [112]:
```
dataset.shape
```

Out[112]:
```
(33332, 16)
```

In [113]:
```
arrayP = np.zeros(10)
```

In [114]:
```
#VALIDAÇÃO CRUZADA (10X)
```

In [115]:

```
print("RESULTADOS POR VALIDAÇÃO CRUZADA (10 X):")
for N in range(10):
  train = dataset.sample(len(dataset)//3, replace=False)
  subdataset = dataset.drop(train.index)
  test = subdataset.sample(len(subdataset)//2, replace=False)
  validation = subdataset.drop(test.index)
  print(f'Teste de Normalidade {N + 1}')
  arrayP[N] = Normality_Test(train['Total'])
  print('\n\n')
```

RESULTADOS POR VALIDAÇÃO CRUZADA (10 X):
Teste de Normalidade 1
DISTRIBUIÇÃO AMOSTRAL: stat = 3576.246
Intervalo de Confiança (15.0 %): Probably Not Gaussian.
Intervalo de Confiança (10.0 %): Probably Not Gaussian.
Intervalo de Confiança (5.0 %): Probably Not Gaussian.
Intervalo de Confiança (2.5 %): Probably Not Gaussian.
Intervalo de Confiança (1.0 %): Probably Not Gaussian.

Teste de Normalidade 2
DISTRIBUIÇÃO AMOSTRAL: stat = 2441.019
Intervalo de Confiança (15.0 %): Probably Not Gaussian.
Intervalo de Confiança (10.0 %): Probably Not Gaussian.
Intervalo de Confiança (5.0 %): Probably Not Gaussian.
Intervalo de Confiança (2.5 %): Probably Not Gaussian.
Intervalo de Confiança (1.0 %): Probably Not Gaussian.

Teste de Normalidade 3
DISTRIBUIÇÃO AMOSTRAL: stat = 3604.021
Intervalo de Confiança (15.0 %): Probably Not Gaussian.
Intervalo de Confiança (10.0 %): Probably Not Gaussian.
Intervalo de Confiança (5.0 %): Probably Not Gaussian.
Intervalo de Confiança (2.5 %): Probably Not Gaussian.
Intervalo de Confiança (1.0 %): Probably Not Gaussian.

Teste de Normalidade 4
DISTRIBUIÇÃO AMOSTRAL: stat = 2377.474
Intervalo de Confiança (15.0 %): Probably Not Gaussian.
Intervalo de Confiança (10.0 %): Probably Not Gaussian.
Intervalo de Confiança (5.0 %): Probably Not Gaussian.
Intervalo de Confiança (2.5 %): Probably Not Gaussian.
Intervalo de Confiança (1.0 %): Probably Not Gaussian.

Teste de Normalidade 5
DISTRIBUIÇÃO AMOSTRAL: stat = 3602.828
Intervalo de Confiança (15.0 %): Probably Not Gaussian.
Intervalo de Confiança (10.0 %): Probably Not Gaussian.
Intervalo de Confiança (5.0 %): Probably Not Gaussian.
Intervalo de Confiança (2.5 %): Probably Not Gaussian.
Intervalo de Confiança (1.0 %): Probably Not Gaussian.

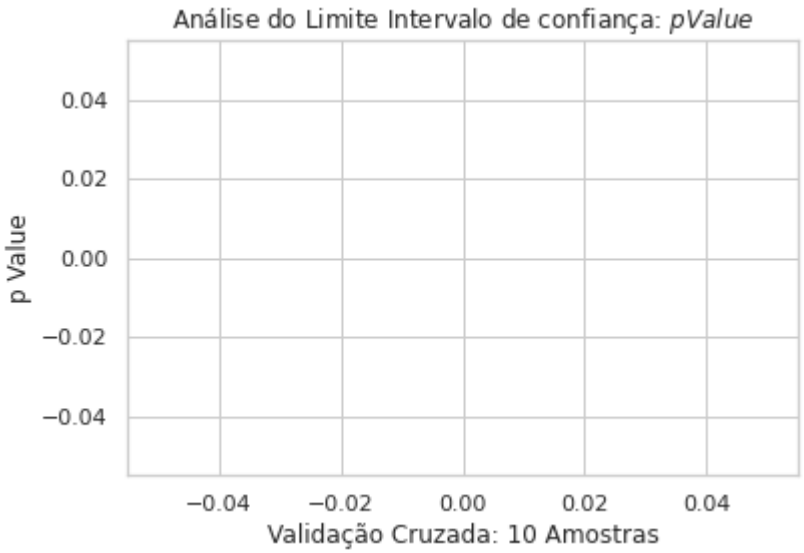
Teste de Normalidade 6
DISTRIBUIÇÃO AMOSTRAL: stat = 3616.892
Intervalo de Confiança (15.0 %): Probably Not Gaussian.
Intervalo de Confiança (10.0 %): Probably Not Gaussian.
Intervalo de Confiança (5.0 %): Probably Not Gaussian.
Intervalo de Confiança (2.5 %): Probably Not Gaussian.
Intervalo de Confiança (1.0 %): Probably Not Gaussian.

Teste de Normalidade 7
DISTRIBUIÇÃO AMOSTRAL: stat = 3605.636
Intervalo de Confiança (15.0 %): Probably Not Gaussian.
Intervalo de Confiança (10.0 %): Probably Not Gaussian.

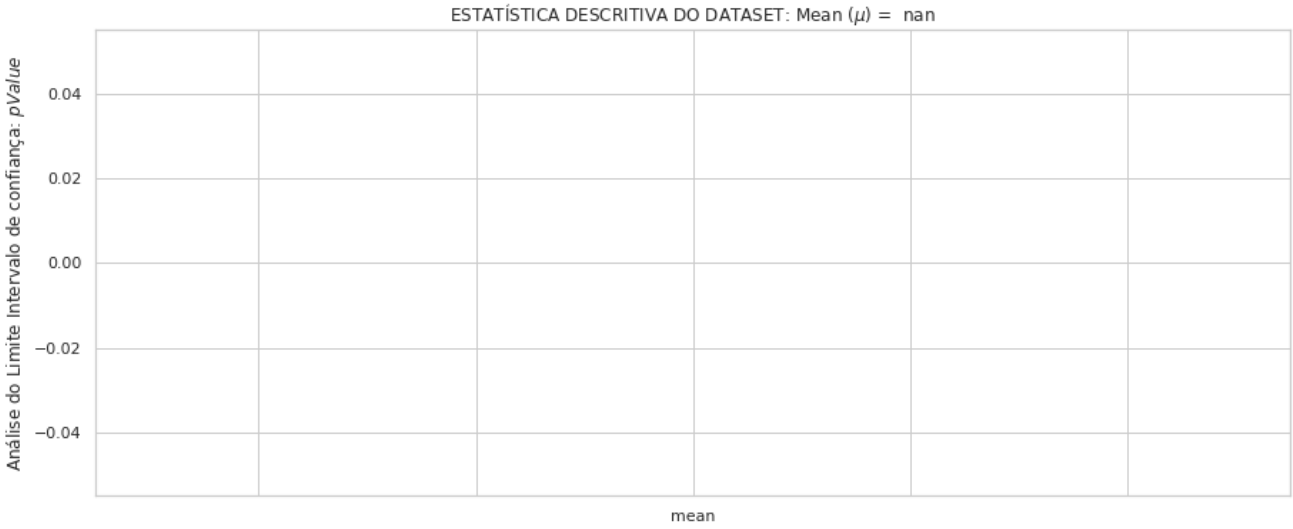
```
Intervalo de Confiança (5.0 %): Probably Not Gaussian.
Intervalo de Confiança (2.5 %): Probably Not Gaussian.
Intervalo de Confiança (1.0 %): Probably Not Gaussian.


Teste de Normalidade 8
DISTRIBUIÇÃO AMOSTRAL: stat = 4219.194
Intervalo de Confiança (15.0 %): Probably Not Gaussian.
Intervalo de Confiança (10.0 %): Probably Not Gaussian.
Intervalo de Confiança (5.0 %): Probably Not Gaussian.
Intervalo de Confiança (2.5 %): Probably Not Gaussian.
Intervalo de Confiança (1.0 %): Probably Not Gaussian.


Teste de Normalidade 9
DISTRIBUIÇÃO AMOSTRAL: stat = 2756.290
Intervalo de Confiança (15.0 %): Probably Not Gaussian.
Intervalo de Confiança (10.0 %): Probably Not Gaussian.
Intervalo de Confiança (5.0 %): Probably Not Gaussian.
Intervalo de Confiança (2.5 %): Probably Not Gaussian.
Intervalo de Confiança (1.0 %): Probably Not Gaussian.


Teste de Normalidade 10
DISTRIBUIÇÃO AMOSTRAL: stat = 2047.858
Intervalo de Confiança (15.0 %): Probably Not Gaussian.
Intervalo de Confiança (10.0 %): Probably Not Gaussian.
Intervalo de Confiança (5.0 %): Probably Not Gaussian.
Intervalo de Confiança (2.5 %): Probably Not Gaussian.
Intervalo de Confiança (1.0 %): Probably Not Gaussian.
```

In [116]:
```
# GRAFICO SEM AS LINHAS ( náo se se vou manter !)
```

In [117]:
```
plt.plot(arrayP)
plt.title('Análise do Limite Intervalo de confiança: $p Value$')
plt.ylabel('p Value')
plt.xlabel('Validação Cruzada: 10 Amostras')
plt.show()
```



In [118]:
```
PlotarStatistic(pd.Series(arrayP), yLabel = 'Análise do Limite Intervalo de confianç
a: $p Value$')
```



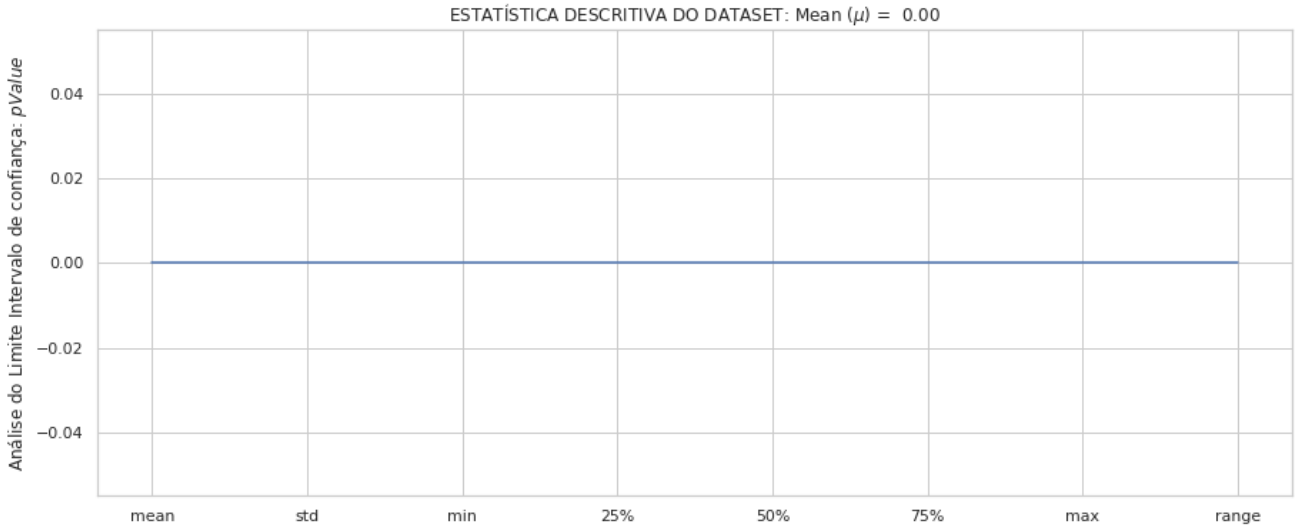

```
ESTATÍSTICA DESCRITIVA DO DATASET:
mean    NaN
std     NaN
min     NaN
25%     NaN
50%     NaN
75%     NaN
max     NaN
range   NaN
dtype: float64
```

In [118]:

In [119]:
```
print("RESULTADOS POR VALIDAÇÃO CRUZADA (10 X):")
for N in range(10):
  train = dataset.sample(len(dataset)//3, replace=False)
  subdataset = dataset.drop(train.index)
  test = subdataset.sample(len(subdataset)//2, replace=False)
  validation = subdataset.drop(test.index)
  stat, p = shapiro(train['Total'])
  arrayP[N] = p
  print(f'Teste de Normalidade {N + 1}: {p:.3f}')
  print('')
```

```
RESULTADOS POR VALIDAÇÃO CRUZADA (10 X):
Teste de Normalidade 1: 0.000

Teste de Normalidade 2: 0.000

Teste de Normalidade 3: 0.000

Teste de Normalidade 4: 0.000

Teste de Normalidade 5: 0.000

Teste de Normalidade 6: 0.000


/usr/local/lib/python3.7/dist-packages/scipy/stats/morestats.py:1760: UserWarning:

p-value may not be accurate for N > 5000.


Teste de Normalidade 7: 0.000

Teste de Normalidade 8: 0.000

Teste de Normalidade 9: 0.000

Teste de Normalidade 10: 0.000
```

In [120]:
```
PlotarStatistic(pd.Series(arrayP), yLabel = 'Análise do Limite Intervalo de confianç
a: $p Value$')
```




```
ESTATÍSTICA DESCRITIVA DO DATASET:
mean     0.0
std      0.0
min      0.0
25%      0.0
50%      0.0
75%      0.0
max      0.0
range    0.0
dtype: float64
```

Atividade (A1): Testes de Hipótese - Normalidade sobre a TARGET (y) (Total gasto pelo Cliente):

- **Normality_Test**: Anderson-Darling Normality Test (com intervalo de confiança)
- **normaltest**: D'Agostino's K^2 Normality Test
- **shapiro**: Shapiro Normality Test

In [121]:
```
# NORMALITY TEST
```

In [122]:
```
#Aplicando teste de normalidade sobre a coluna Price - amostra 1
```

In [123]:
```
Normality_Test(sample1_random['Total'])
```

```
DISTRIBUIÇÃO AMOSTRAL: stat = 218.857
Intervalo de Confiança (15.0 %): Probably Not Gaussian.
Intervalo de Confiança (10.0 %): Probably Not Gaussian.
Intervalo de Confiança (5.0 %): Probably Not Gaussian.
Intervalo de Confiança (2.5 %): Probably Not Gaussian.
Intervalo de Confiança (1.0 %): Probably Not Gaussian.
```

In [124]:
```
#Aplicando teste de normalidade sobre a coluna Price - amostra 2
```

In [125]:
```
Normality_Test(sample2_random['Total'])
```

```
DISTRIBUIÇÃO AMOSTRAL: stat = 185.241
Intervalo de Confiança (15.0 %): Probably Not Gaussian.
Intervalo de Confiança (10.0 %): Probably Not Gaussian.
Intervalo de Confiança (5.0 %): Probably Not Gaussian.
Intervalo de Confiança (2.5 %): Probably Not Gaussian.
Intervalo de Confiança (1.0 %): Probably Not Gaussian.
```

In [126]:
```
#Aplicando teste de normalidade sobre a coluna Price - amostra 3
```

In [127]:
```
Normality_Test(sample3_random['Total'])
```

```
DISTRIBUIÇÃO AMOSTRAL: stat = 216.029
Intervalo de Confiança (15.0 %): Probably Not Gaussian.
Intervalo de Confiança (10.0 %): Probably Not Gaussian.
Intervalo de Confiança (5.0 %): Probably Not Gaussian.
Intervalo de Confiança (2.5 %): Probably Not Gaussian.
Intervalo de Confiança (1.0 %): Probably Not Gaussian.
```

In [128]:
```
# NORMAL TEST
```

In [129]:
```
# Exemplo da técnica D`AGOSTINO`S K^2 Normality Test
```

In [130]:
```
stat, p = normaltest(dataset['Total'])
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
        print('Probably Gaussian')
else:
        print('Probably not Gaussian')
```

```
stat=158371.009, p=0.000
Probably not Gaussian
```

In [131]:
```
# EXEMPLO DA TECNICA DE SHAPIRO
```

In [132]:
```
stat, p = shapiro(dataset['Total'])
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
        print('Probably Gaussian')
else:
        print('Probably not Gaussian')
```

```
stat=0.002, p=0.000
Probably not Gaussian

/usr/local/lib/python3.7/dist-packages/scipy/stats/morestats.py:1760: UserWarning:

p-value may not be accurate for N > 5000.
```

Atividade (A2): Testes de Correlação:

- **Pearson's Correlation Coefficient**
- **Spearman's Rank Correlation**
- **Kendall's Rank Correlation**
- **Chi-Squared Test**

Dataset: TREINAMENTO x TESTE

In [133]:
```
# FAZENDO OS TESTES DE CORRELAÇÕES( AQUI ESCOLHI A COLUNA PRICE também!)
```

In [134]:
```
# Pearson's Correlation Coefficient

stat, p = pearsonr(sample1_random['Total'], sample2_random['Total'])
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
        print('Probably independent')
else:
        print('Probably dependent')
```

```
stat=-0.019, p=0.547
Probably independent
```

In [135]:
```
# Spearman's Rank Correlation

stat, p = spearmanr(sample1_random['Total'], sample2_random['Total'])
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
        print('Probably independent')
else:
        print('Probably dependent')
```

```
stat=-0.002, p=0.958
Probably independent
```

In [136]:
```
#Kendall's Rank Correlation

stat, p = kendalltau(sample1_random['Total'], sample2_random['Total'])
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
        print('Probably independent')
else:
        print('Probably dependent')
```

```
stat=-0.001, p=0.965
Probably independent
```

In [137]:
```
#Chi-Squared Test
stat, p, dof, expected = chi2_contingency(sample1_random['Total'], sample2_random['To
tal'])
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
        print('Probably independent')
else:
        print('Probably dependent')
```

```
stat=0.000, p=1.000
Probably independent
```

In [138]:
```
# DATASET DE TREINAMENTO
```

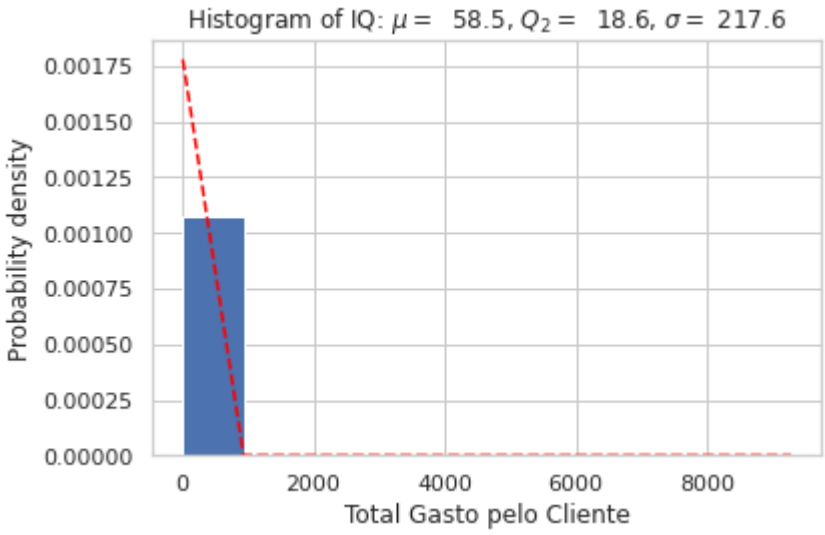
In [139]:
```
train.reset_index(drop=True, inplace=True)
train
```

Out[139]:

	TransactionNo	Data	ProductNo	ProductName	CustomerNo	Country	Cost	Unity	Qua
0	579533	11/30/2019	23284	DOORMAT KEEP CALM AND COME IN	12953.0	UNITED KINGDOM	0.23	3.10	
1	577807	11/22/2019	23393	HOME SWEET HOME CUSHION COVER	15699.0	UNITED KINGDOM	0.76	1.49	
2	580401	2019-04-12 00:00:00	22148	EASTER CRAFT 4 CHICKS	17618.0	UNITED KINGDOM	0.72	0.52	
3	578539	11/24/2019	22540	MINI JIGSAW CIRCUS PARADE	16539.0	UNITED KINGDOM	0.51	0.26	
4	578488	11/24/2019	23583	LUNCH BAG PAISLEY PARK	12784.0	PORTUGAL	0.05	0.72	
...	...	...	...	...	...	...	...	...	
11105	578926	11/27/2019	23357	HOT WATER BOTTLE SEX BOMB	18156.0	UNITED KINGDOM	0.54	2.41	
11106	579885	11/30/2019	22170	PICTURE FRAME WOOD TRIPLE PORTRAIT	15444.0	UNITED KINGDOM	0.59	7.24	
11107	577813	11/22/2019	23265	SET OF 3 WOODEN TREE DECORATIONS	14297.0	UNITED KINGDOM	0.09	0.26	
11108	579507	11/29/2019	22941	CHRISTMAS LIGHTS 10 REINDEER	15297.0	UNITED KINGDOM	0.86	3.62	
11109	579103	11/28/2019	22578	WOODEN STAR CHRISTMAS SCANDINAVIAN	18041.0	UNITED KINGDOM	0.97	1.00	

11110 rows × 16 columns

In [140]:
```
PlotarDistrit(train['Total'], bins = 10) # bins ajustado
```



In [141]:
```
# DATASET DE TESTE
```

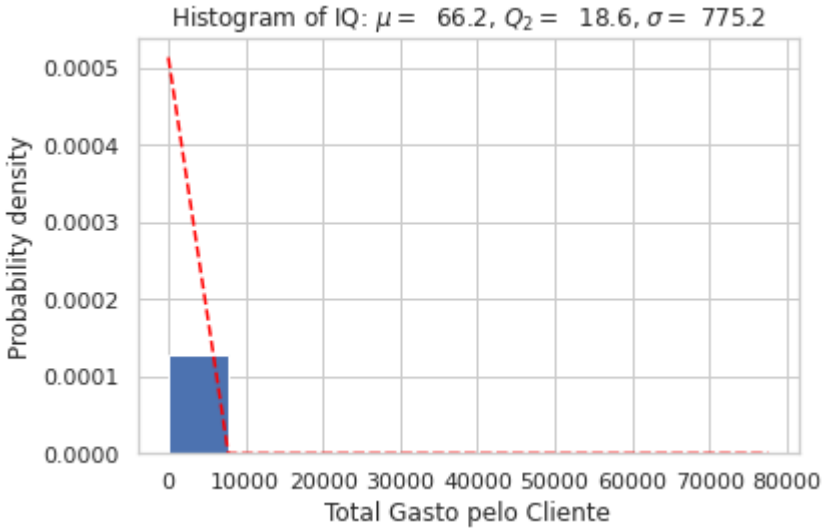
In [142]:
```
test.reset_index(drop=True, inplace=True)
test
```

Out[142]:

	TransactionNo	Data	ProductNo	ProductName	CustomerNo	Country	Cost	Unity	Quant
0	579196	11/28/2019	20728	LUNCH BAG CARS BLUE	14096.0	UNITED KINGDOM	0.68	6.19	
1	580504	2019-04-12 00:00:00	84879	ASSORTED COLOUR BIRD ORNAMENT	17936.0	UNITED KINGDOM	0.67	3.10	
2	579152	11/28/2019	20978	36 PENCILS TUBE SKULLS	12479.0	GERMANY	0.13	0.39	
3	578843	11/25/2019	22948	METAL DECORATION NAUGHTY CHILDREN	17813.0	UNITED KINGDOM	0.60	1.55	
4	579001	11/27/2019	23006	TRAVEL CARD WALLET FLOWER MEADOW	15555.0	UNITED KINGDOM	0.72	1.24	
...	...	...	...	...	...	...	...	...	
11106	581119	2019-07-12 00:00:00	23367	SET 12 COLOUR PENCILS SPACEBOY	15136.0	UNITED KINGDOM	0.49	0.39	
11107	580256	2019-02-12 00:00:00	22274	FELTCRAFT DOLL EMILY	14688.0	UNITED KINGDOM	0.00	6.19	
11108	579557	11/30/2019	23206	LUNCH BAG APPLE DESIGN	12557.0	UNITED KINGDOM	0.16	0.69	
11109	580504	2019-04-12 00:00:00	23281	FOLDING BUTTERFLY MIRROR RED	17936.0	UNITED KINGDOM	0.39	0.52	
11110	579473	11/29/2019	22959	WRAP CHRISTMAS VILLAGE	13198.0	UNITED KINGDOM	0.78	0.25	

11111 rows × 16 columns

In [143]:
```
PlotarDistrit(test['Total'], bins = 10) # bins ajustado
```



Dataset: TREINAMENTO x VALIDAÇÃO

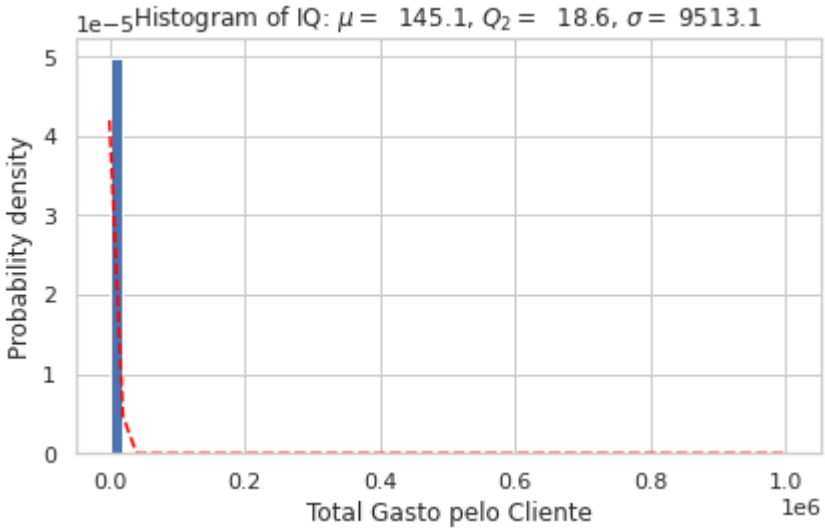
In [144]:
```
validation.reset_index(drop=True, inplace=True)
validation
```

Out[144]:

	TransactionNo	Data	ProductNo	ProductName	CustomerNo	Country	Cost	Unity	Q
0	581475	2019-09-12 00:00:00	22596	CHRISTMAS STAR WISH LIST CHALKBOARD	13069.0	UNITED KINGDOM	0.19	0.30	
1	581475	2019-09-12 00:00:00	21705	BAG 500G SWIRLY MARBLES	13069.0	UNITED KINGDOM	0.06	0.44	
2	581475	2019-09-12 00:00:00	22217	T-LIGHT HOLDER HANGING LACE	13069.0	UNITED KINGDOM	0.73	0.89	
3	581475	2019-09-12 00:00:00	22785	SQUARECUSHION COVER PINK UNION JACK	13069.0	UNITED KINGDOM	0.95	0.96	
4	581475	2019-09-12 00:00:00	22956	36 FOIL HEART CAKE CASES	13069.0	UNITED KINGDOM	0.34	0.46	
...	...	...	...	...	...	...	...	...	
11106	577855	11/22/2019	23118	PARISIENNE JEWELLERY DRAWER	15877.0	UNITED KINGDOM	0.35	6.13	
11107	577855	11/22/2019	22621	TRADITIONAL KNITTING NANCY	15877.0	UNITED KINGDOM	0.52	6.13	
11108	577855	11/22/2019	22593	CHRISTMAS GINGHAM STAR	15877.0	UNITED KINGDOM	0.17	6.13	
11109	577855	11/22/2019	21211	SET OF 72 SKULL PAPER DOILIES	15877.0	UNITED KINGDOM	0.40	6.13	
11110	577855	11/22/2019	22262	FELT EGG COSY CHICKEN	15877.0	UNITED KINGDOM	0.78	7.29	

11111 rows × 16 columns

In [145]:
```
PlotarDistrit(validation['Total'], bins = 50) #bins ajustado
```



Plotar os gráficos (Método: PlotarStatistic feito por mim) dos parâmetros dos 10 SPLIT's, para os valores de:

- **p-value**: p

Atividade (A3): Nonparametric Statistical Hypothesis Tests:

- **Mann-Whitney U Test**
- **Wilcoxon Signed-Rank Test**
- **Kruskal-Wallis H Test**

In [146]:
```
# Examplo com a técnica de  Mann-Whitney U Test
stat, p = mannwhitneyu(sample1_random['Total'], sample2_random['Total'])
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
        print('Probably the same distribution')
else:
        print('Probably different distributions')
```

```
stat=497162.000, p=0.059
Probably the same distribution
```

In [147]:
```
# Examplo coom a técnica Wilcoxon Signed-Rank Test
stat, p = wilcoxon(sample1_random['Total'], sample2_random['Total'])
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
        print('Probably the same distribution')
else:
        print('Probably different distributions')
```

```
stat=223289.000, p=0.198
Probably the same distribution
```

In [148]:
```
# Examplo  com a técnica  Kruskal-Wallis H Test
stat, p = kruskal(sample1_random['Total'], sample2_random['Total'])
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
        print('Probably the same distribution')
else:
        print('Probably different distributions')
```

```
stat=3.557, p=0.059
Probably the same distribution
```

ETAPA 3.4: Análise e Redução de Dimensionalidade

Atividade (A1): Análise das Características (FEATURES):

- **(A1.1): Removing features with low variance (σ^2)**

In [ ]:
```
# REMOVENDO AS COLUNAS CATEGÓRICAS QUALITIATIVAS
```

In [179]:
```
datasetLimpo=dataset.drop(['Data', 'TransactionNo', 'ProductNo','CustomerNo',"Genre",
'ProductName','Age', 'Payment', 'Country'],axis=1)
```

In [180]:
```
datasetLimpo
```

Out[180]:

	Cost	Unity	Quantity	Price	Total	Profit	Profile
0	0.43	1.79	12	21.47	257.64	257.21	6
1	0.19	0.30	36	10.65	383.40	383.21	7
2	0.50	0.96	12	11.53	138.36	137.86	4
3	0.08	0.89	12	10.65	127.80	127.72	4
5	0.06	0.44	24	10.65	255.60	255.54	6
...	...	...	...	...	...	...	...
49990	0.40	6.13	1	6.13	6.13	5.73	1
49991	0.34	6.13	1	6.13	6.13	5.79	1
49994	0.78	7.29	1	7.29	7.29	6.51	1
49995	0.36	7.29	1	7.29	7.29	6.93	1
49997	0.99	7.29	1	7.29	7.29	6.30	1

33332 rows × 7 columns

In [ ]:
```
# MOSTRANDO AS COLUNAS NUMÉRICAS COM MENOR VARIÂNCIA
```

In [181]:
```
from sklearn.feature_selection import VarianceThreshold
threshold_n=0.95
sel = VarianceThreshold(threshold=(threshold_n* (1 - threshold_n) ))
sel_var=sel.fit_transform(datasetLimpo)
datasetLimpo[datasetLimpo.columns[sel.get_support(indices=True)]]
```

Out[181]:

	Cost	Unity	Quantity	Price	Total	Profit	Profile
0	0.43	1.79	12	21.47	257.64	257.21	6
1	0.19	0.30	36	10.65	383.40	383.21	7
2	0.50	0.96	12	11.53	138.36	137.86	4
3	0.08	0.89	12	10.65	127.80	127.72	4
5	0.06	0.44	24	10.65	255.60	255.54	6
...	...	...	...	...	...	...	...
49990	0.40	6.13	1	6.13	6.13	5.73	1
49991	0.34	6.13	1	6.13	6.13	5.79	1
49994	0.78	7.29	1	7.29	7.29	6.51	1
49995	0.36	7.29	1	7.29	7.29	6.93	1
49997	0.99	7.29	1	7.29	7.29	6.30	1

33332 rows × 7 columns

In [ ]:
```
# FUNÇÃO QUE RETORNA COLUNAS COM MENOR VARIÂNCIA
```

In [182]:
```
from sklearn.feature_selection import VarianceThreshold
from itertools import compress

def fs_variance(df, threshold:float=0.1):
    """
    Return a list of selected variables based on the threshold.
    """

    # The list of columns in the data frame
    features = list(df.columns)

    # Initialize and fit the method
    vt = VarianceThreshold(threshold = threshold)
    _ = vt.fit(df)

    # Get which column names which pass the threshold
    feat_select = list(compress(features, vt.get_support()))

    return feat_select
```

In [ ]:
```
# USANDO A FUNÇÃO ACIMA
```

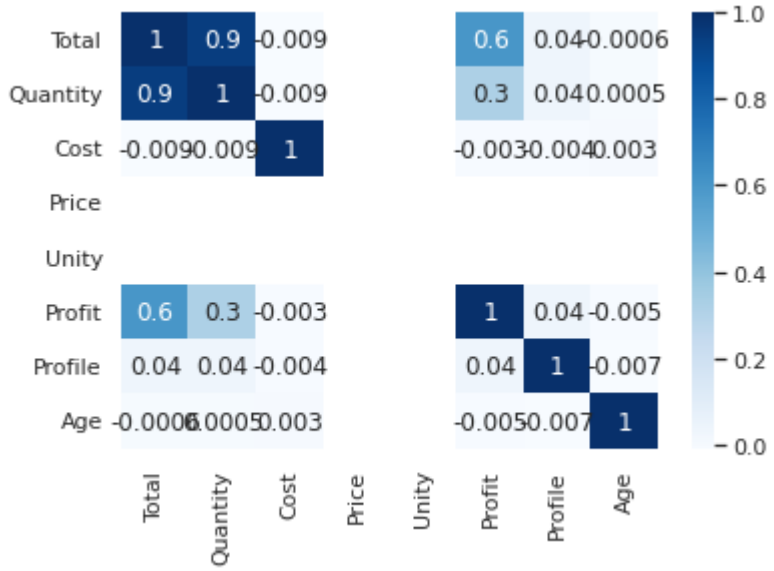
In [183]:
```
fs_variance(datasetLimpo)
```

Out[183]:
```
['Unity  ', 'Quantity', 'Price ', 'Total', 'Profit', 'Profile']
```

In [184]:
```
# HEATMAP- Utitlizei colunas criadas para melhor visualização do dataset original
```

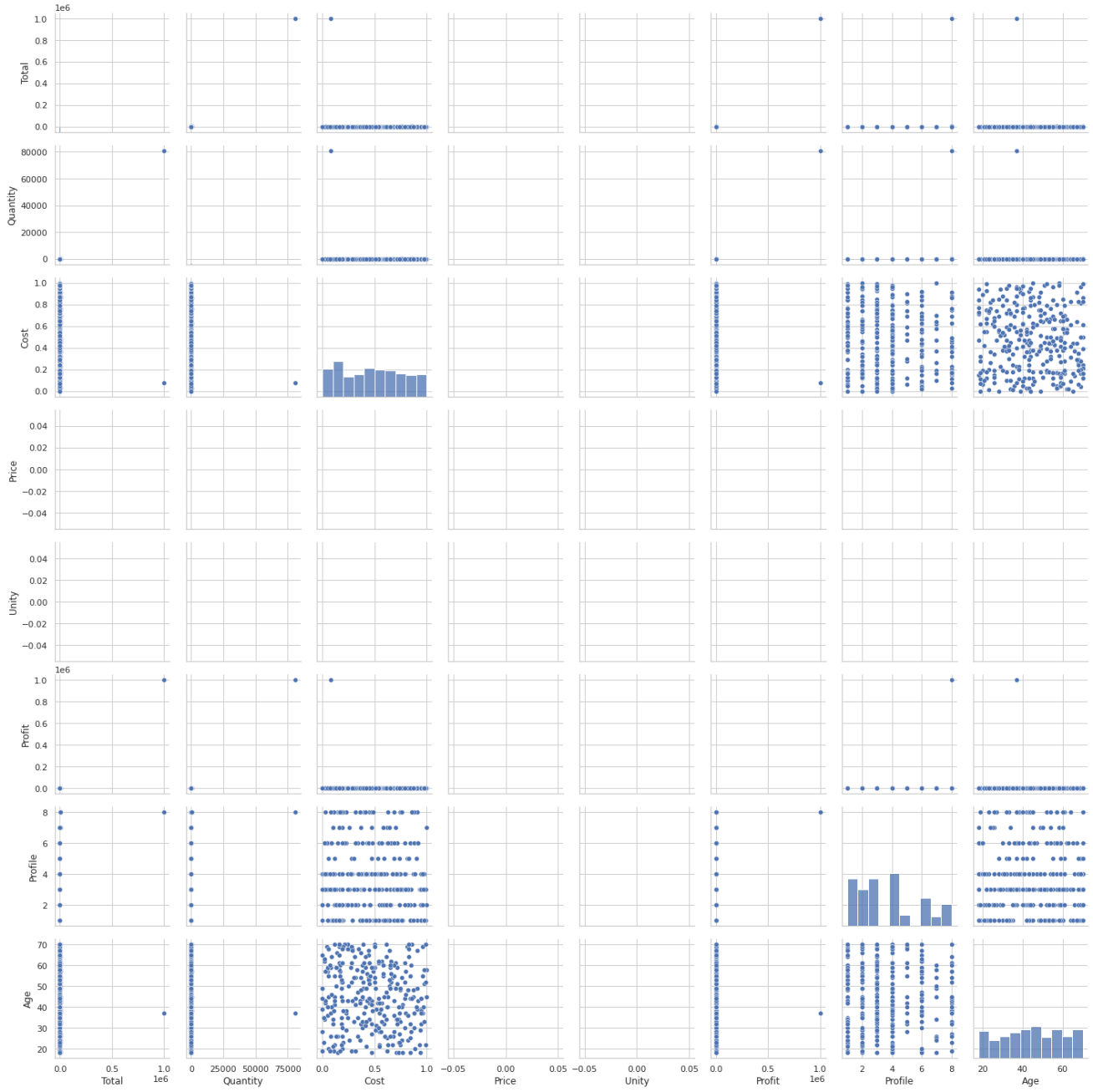
In [161]:
```
X=pd.DataFrame(datasetLimpo, columns=['Total', 'Quantity', 'Cost', 'Price', 'Unity',
'Profit', 'Profile', 'Age'])
```

In [169]:
```
heatmap = sns.heatmap(X.corr(), annot=True, cmap="Blues", fmt='.0g')
```



In [170]:
```
sns.pairplot(X[0:300]) # PEGANDO APENAS UMA PEQUENA AMOSTRA DE 300 LINHAS, pois o dataset original consome toda RAM do colab
```

Out[170]: <seaborn.axisgrid.PairGrid at 0x7f847f48e650>



ETAPA 4: Data Modeling

ETAPA 5: Data Presentation

CONVERTENDO IPYNB para PDF

PASSO A PASSO:

1. Fazer o download do seu IPYNB.
2. Fazer o upload do seu IPYNB para área temporária: content.
3. Rodar o código abaixo com seu NOME_DO_COLAB.ipynb: Gerar um HTML.
4. Fazer o download do seu HTML e abrir em um Browser qualquer.
5. Imprimir a página HTML em PDF. Pronto (UFA)!

In [172]:
```
%%shell
jupyter nbconvert --to html //content/Proj_IntegradorETAPA_1_Thiago_Rosemberg_Jager.ipynb
```

```
[NbConvertApp] Converting notebook //content/Proj_IntegradorETAPA_1_Thiago_Rosemberg_Jager.ipynb to html
[NbConvertApp] Writing 4801265 bytes to //content/Proj_IntegradorETAPA_1_Thiago_Rosemberg_Jager.html
```

Out[172]: