In [1]:
```
# Problem Statement 3: You are working in an e-commerce company,
# and your company has put forward a task to analyze the customer
# reviews for various products.
# You are supposed to create a report that classifies
# the products based on the customer reviews.
```

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [3]:
```
ecom = pd.read_csv("Reviews.csv")
```

In [4]:
```
ecom
```

Out[4]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summar
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	Goo Quality Do Foo
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1346976000	Not a Advertise
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	1219017600	"Delight says it a
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2	1307923200	Coug Medicin
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0	5	1350777600	Great taff
...	...	...	...	...	...	...	...	...	.
568449	568450	B001EO7N10	A28KG5XORO54AY	Lettie D. Carter	0	0	5	1299628800	Will not d withou
568450	568451	B003S1WTCU	A3I8AFVPEE8KI5	R. Sawyer	0	0	2	1331251200	disappointe
568451	568452	B004I613EE	A121AA1GQV751Z	pksd "pk_007"	2	2	5	1329782400	Perfect fo our maltipo
568452	568453	B004I613EE	A3IBEVCTXKNOH	Kathy A. Welch "katwel"	1	1	5	1331596800	Favorit Training an reward trea
568453	568454	B001LR2CU2	A3LGQPJCZVL9UC	srfell17	0	0	5	1338422400	Great Hone

568454 rows × 10 columns

In [5]:
```
ecom.head( 60145)
```

Out[5]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Sum
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	Good Q Dog

1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1346976000	N Adver
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	1219017600	"Delight"
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2	1307923200	C Mec
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0	5	1350777600	Great
...	...	...	...	...	...	...	...	...	
60140	60141	B000EDG598	AQHFWMW7AV9NG	Penny Lane	12	12	5	1294272000	MARVEL as a cleanse n
60141	60142	B000EDG598	A3QWPNDJ7ET7BL	Dagger	13	14	5	1150675200	So ver and
60142	60143	B000EDG598	A2D95KT35FSHE9	S. Leiker	9	9	5	1217116800	gluter
60143	60144	B000EDG598	A3C4DA2XI4X4JA	Lisa B.	7	7	5	1233792000	Bob's Mill Alr
60144	60145	B000EDG598	A3UKWQS8SRW6IO	TropicalMinnesota	6	6	5	1316217600	Great pr

60145 rows × 10 columns

In [6]:
```
ecom.shape
```

Out[6]: (568454, 10)

In [7]:
```
ecom.describe()
```

Out[7]:

	Id	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time
count	568454.000000	568454.000000	568454.00000	568454.000000	5.684540e+05
mean	284227.500000	1.743817	2.22881	4.183199	1.296257e+09
std	164098.679298	7.636513	8.28974	1.310436	4.804331e+07
min	1.000000	0.000000	0.00000	1.000000	9.393408e+08
25%	142114.250000	0.000000	0.00000	4.000000	1.271290e+09
50%	284227.500000	0.000000	1.00000	5.000000	1.311120e+09
75%	426340.750000	2.000000	2.00000	5.000000	1.332720e+09
max	568454.000000	866.000000	923.00000	5.000000	1.351210e+09

In [8]:
```
ecom.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 568454 entries, 0 to 568453
Data columns (total 10 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   Id                      568454 non-null  int64
 1   ProductId               568454 non-null  object
 2   UserId                  568454 non-null  object
 3   ProfileName             568438 non-null  object
 4   HelpfulnessNumerator    568454 non-null  int64
 5   HelpfulnessDenominator  568454 non-null  int64
 6   Score                   568454 non-null  int64
 7   Time                    568454 non-null  int64
 8   Summary                 568427 non-null  object
 9   Text                    568454 non-null  object
dtypes: int64(5), object(5)
memory usage: 43.4+ MB
```

In [9]:
```
ecom.isna().sum()
```

Out[9]:
```
Id                         0
ProductId                  0
UserId                     0
ProfileName               16
HelpfulnessNumerator       0
HelpfulnessDenominator     0
Score                      0
Time                       0
Summary                   27
Text                       0
dtype: int64
```

In [10]:
```
ecom.dtypes
```

Out[10]:
```
Id                         int64
ProductId                 object
UserId                    object
ProfileName               object
HelpfulnessNumerator       int64
HelpfulnessDenominator     int64
Score                      int64
Time                       int64
Summary                   object
Text                      object
dtype: object
```

In [11]:
```
import numpy as np
ecom['Id'] = np.where(ecom['ProductId'] == 'yes',1,0)
ecom['HelpfulnessDenominator'] = ecom['Time']
type(ecom["Text"])
```

Out[11]:
```
pandas.core.series.Series
```

In [12]:
```
ecom['Id'] = np.where(ecom['ProductId'] == 'yes',1,0)
ecom['UserId'] = ecom['Score'].astype(str)
```

In [13]:
```
ecom.head(100)
```

Out[13]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
0	0	B001E4KFG0	5	delmartian	1	1303862400	5	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned d...
1	0	B00813GRG4	1	dll pa	0	1346976000	1	1346976000	Not as Advertised	Product arrived labeled as Jumbo Salted Peanut...

This is a

2	0	B000LQOCH0	4	Natalia Corres "Natalia Corres"	1	1219017600	4	1219017600	"Delight" says it all	confection that has been around a fe...
3	0	B000UA0QIQ	2	Karl	3	1307923200	2	1307923200	Cough Medicine	If you are looking for the secret ingredient i...
4	0	B006K2ZZ7K	5	Michael D. Bigham "M. Wassir"	0	1350777600	5	1350777600	Great taffy	Great taffy at a great price. There was a wid...
...	...	...	...	...	...	...	...	...	...	...
95	0	B0019CW0HE	5	E. Triebe	0	1320105600	5	1320105600	Good healthy dog food	I've been very pleased with the Natural Balanc...
96	0	B0019CW0HE	5	Rhiever	0	1303776000	5	1303776000	Great dog food	My 1-1/2 year old basenji/jack russell mix lov...
97	0	B0019CW0HE	5	FuNky Faja "SiLkk"	0	1297296000	5	1297296000	Great allergy sensitive dog food, dogs love it	Our pup has experienced allergies in forms of ...
98	0	B0019CW0HE	5	Amazon-tron 3000	0	1295308800	5	1295308800	Perfect for our English Bulldog with Allergies	My English Bulldog had skin allergies the summ...
99	0	B0019CW0HE	1	Melissa Benjamin	0	1331164800	1	1331164800	Bad	I fed this to my Golden Retriever and he hated...

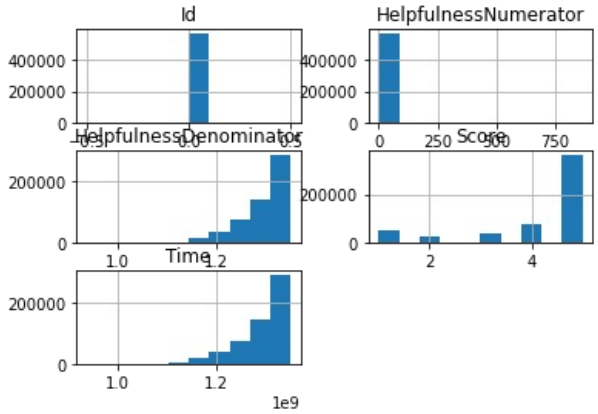
100 rows × 10 columns

In [14]:
```
print(ecom['HelpfulnessDenominator'].mode())
```

```
0    1350345600
dtype: int64
```

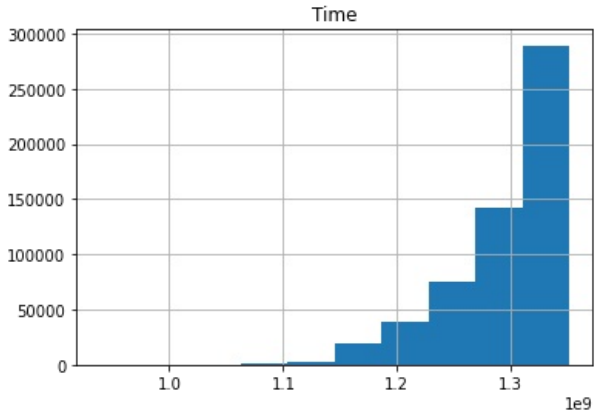
In [15]:
```
ecom.hist()
```

Out[15]:
```
array([[<AxesSubplot:title={'center':'Id'}>,
        <AxesSubplot:title={'center':'HelpfulnessNumerator'}>],
       [<AxesSubplot:title={'center':'HelpfulnessDenominator'}>,
        <AxesSubplot:title={'center':'Score'}>],
       [<AxesSubplot:title={'center':'Time'}>, <AxesSubplot:>]],
      dtype=object)
```



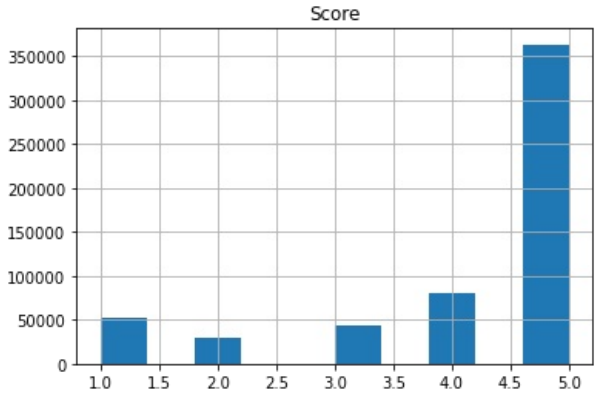
In [16]:
```
ecom.hist('Time')
```

Out[16]:
```
array([[<AxesSubplot:title={'center':'Time'}>]], dtype=object)
```



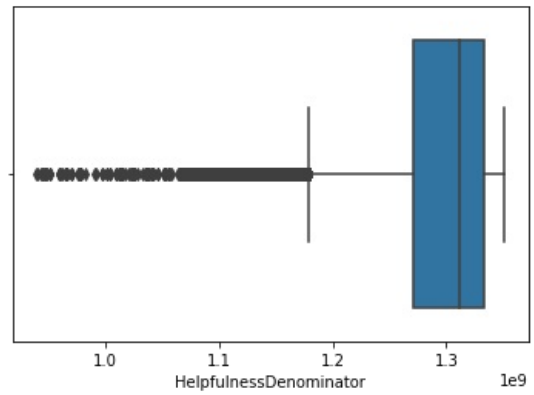
In [17]:
```
ecom.hist('Score')
```

Out[17]:
```
array([[<AxesSubplot:title={'center':'Score'}>]], dtype=object)
```



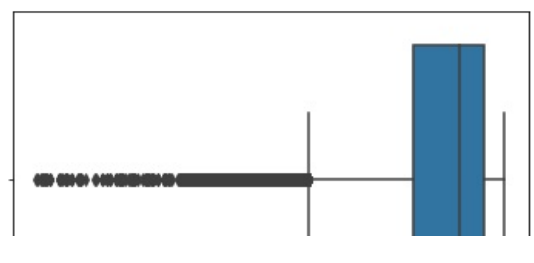
In [18]:
```
import seaborn as sns
sns.boxplot(x=ecom['HelpfulnessDenominator'])
```

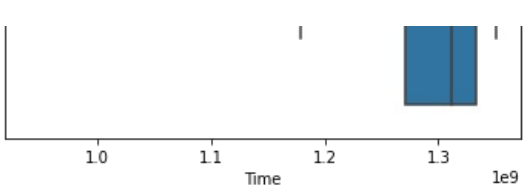
Out[18]:
```
<AxesSubplot:xlabel='HelpfulnessDenominator'>
```



In [19]:
```
import seaborn as sns
sns.boxplot(x=ecom['Time'])
```

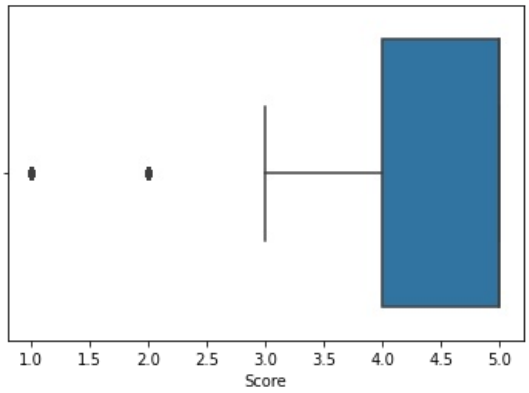
Out[19]:
```
<AxesSubplot:xlabel='Time'>
```





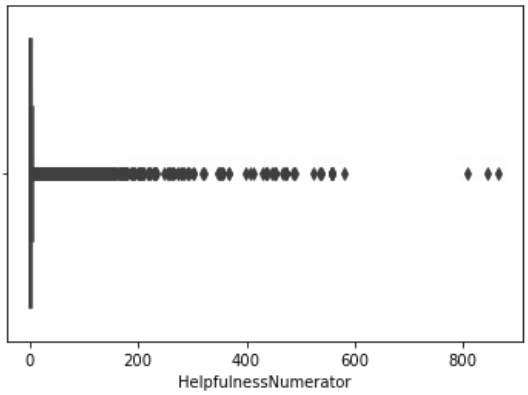
In [20]:
```
import seaborn as sns
sns.boxplot(x=ecom['Score'])
```

Out[20]: <AxesSubplot:xlabel='Score'>



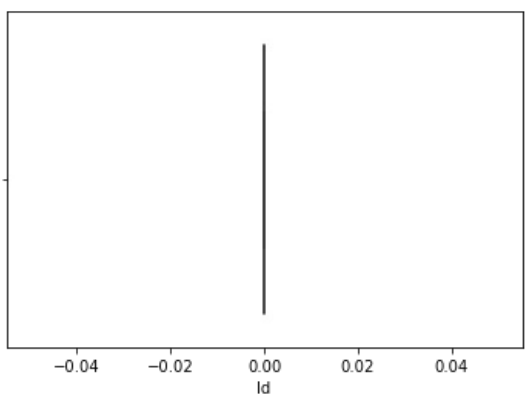
In [21]:
```
import seaborn as sns
sns.boxplot(x=ecom['HelpfulnessNumerator'])
```

Out[21]: <AxesSubplot:xlabel='HelpfulnessNumerator'>



In [22]:
```
import seaborn as sns
sns.boxplot(x=ecom['Id'])
```

Out[22]: <AxesSubplot:xlabel='Id'>



In [23]:
```
ecom.describe()
```

Out[23]:

	Id	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time
count	568454.0	568454.000000	5.684540e+05	568454.000000	5.684540e+05
mean	0.0	1.743817	1.296257e+09	4.183199	1.296257e+09
std	0.0	7.636513	4.804331e+07	1.310436	4.804331e+07
min	0.0	0.000000	9.393408e+08	1.000000	9.393408e+08
25%	0.0	0.000000	1.271290e+09	4.000000	1.271290e+09
50%	0.0	0.000000	1.311120e+09	5.000000	1.311120e+09
75%	0.0	2.000000	1.332720e+09	5.000000	1.332720e+09
max	0.0	866.000000	1.351210e+09	5.000000	1.351210e+09

In [24]:
```
Q1A = ecom.Time.quantile(0.25)
Q3A = ecom.Score.quantile(0.75)
IQRA= Q3A-Q1A
print(IQRA)
```

```
-1271289595.0
```

In [25]:
```
print(Q1A-1.5*IQRA)
print(Q3A+1.5*IQRA)
```

```
3178223992.5
-1906934387.5
```

In [26]:
```
Q1 = ecom.quantile(0.25)
Q3 = ecom.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

```
Id                              0.0
HelpfulnessNumerator            2.0
HelpfulnessDenominator    61430400.0
Score                           1.0
Time                      61430400.0
dtype: float64
```

In [27]:
```
print(ecom < (Q1 - 1.5 * IQR)) |(ecom > (Q3 + 1.5 * IQR))
```

```
<ipython-input-27-149d5f3b7223>:1: FutureWarning: Automatic reindexing on DataFrame vs Series comparisons is deprecated and will raise ValueError in a future version.  Do `left, right = left.align(right, axis=1, copy=False)` before e.g. `left == right`
  print(ecom < (Q1 - 1.5 * IQR)) |(ecom > (Q3 + 1.5 * IQR))
        HelpfulnessDenominator  HelpfulnessNumerator     Id  ProductId  \
0                        False                 False  False      False
1                        False                 False  False      False
2                        False                 False  False      False
3                        False                 False  False      False
4                        False                 False  False      False
...                        ...                   ...    ...        ...
568449                   False                 False  False      False
568450                   False                 False  False      False
568451                   False                 False  False      False
568452                   False                 False  False      False
568453                   False                 False  False      False

        ProfileName  Score  Summary   Text   Time  UserId
0             False  False    False  False  False   False
1             False   True    False  False  False   False
2             False  False    False  False  False   False
3             False   True    False  False  False   False
4             False  False    False  False  False   False
...             ...    ...      ...    ...    ...     ...
568449        False  False    False  False  False   False
568450        False   True    False  False  False   False
568451        False  False    False  False  False   False
568452        False  False    False  False  False   False
```

```
568453        False  False    False  False  False   False

[568454 rows x 10 columns]
<ipython-input-27-149d5f3b7223>:1: FutureWarning: Automatic reindexing on DataFrame vs Series comparisons is depr
ecated and will raise ValueError in a future version.  Do `left, right = left.align(right, axis=1, copy=False)` b
efore e.g. `left == right`
  print(ecom < (Q1 - 1.5 * IQR)) |(ecom > (Q3 + 1.5 * IQR))
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/opt/anaconda3/lib/python3.8/site-packages/pandas/core/ops/array_ops.py in na_logical_op(x, y, op)
    264         #  (xint or xbool) and (yint or bool)
--> 265         result = op(x, y)
    266     except TypeError:

/opt/anaconda3/lib/python3.8/site-packages/pandas/core/ops/roperator.py in ror_(left, right)
     55 def ror_(left, right):
---> 56     return operator.or_(right, left)
     57

TypeError: unsupported operand type(s) for |: 'NoneType' and 'bool'

During handling of the above exception, another exception occurred:

ValueError                                Traceback (most recent call last)
/opt/anaconda3/lib/python3.8/site-packages/pandas/core/ops/array_ops.py in na_logical_op(x, y, op)
    278             try:
--> 279                 result = libops.scalar_binop(x, y, op)
    280             except (

pandas/_libs/ops.pyx in pandas._libs.ops.scalar_binop()

ValueError: Buffer has wrong number of dimensions (expected 1, got 2)

The above exception was the direct cause of the following exception:

TypeError                                 Traceback (most recent call last)
<ipython-input-27-149d5f3b7223> in <module>
----> 1 print(ecom < (Q1 - 1.5 * IQR)) |(ecom > (Q3 + 1.5 * IQR))

/opt/anaconda3/lib/python3.8/site-packages/pandas/core/ops/common.py in new_method(self, other)
     63         other = item_from_zerodim(other)
     64
---> 65         return method(self, other)
     66
     67     return new_method

/opt/anaconda3/lib/python3.8/site-packages/pandas/core/arraylike.py in __ror__(self, other)
     69     @unpack_zerodim_and_defer("__ror__")
     70     def __ror__(self, other):
---> 71         return self._logical_method(other, roperator.ror_)
     72
     73     @unpack_zerodim_and_defer("__xor__")

/opt/anaconda3/lib/python3.8/site-packages/pandas/core/frame.py in _arith_method(self, other, op)
   5980         self, other = ops.align_method_FRAME(self, other, axis, flex=True, level=None)
   5981
-> 5982         new_data = self._dispatch_frame_op(other, op, axis=axis)
   5983         return self._construct_result(new_data)
   5984

/opt/anaconda3/lib/python3.8/site-packages/pandas/core/frame.py in _dispatch_frame_op(self, right, func, axis)
   6006         if not is_list_like(right):
   6007             # i.e. scalar, faster than checking np.ndim(right) == 0
-> 6008             bm = self._mgr.apply(array_op, right=right)
   6009             return type(self)(bm)
   6010

/opt/anaconda3/lib/python3.8/site-packages/pandas/core/internals/managers.py in apply(self, f, align_keys, ignore
_failures, **kwargs)
    423             try:
    424                 if callable(f):
--> 425                     applied = b.apply(f, **kwargs)
    426                 else:
    427                     applied = getattr(b, f)(**kwargs)

/opt/anaconda3/lib/python3.8/site-packages/pandas/core/internals/blocks.py in apply(self, func, **kwargs)
    376         """
    377         with np.errstate(all="ignore"):
--> 378             result = func(self.values, **kwargs)
    379
    380         return self._split_op_result(result)

/opt/anaconda3/lib/python3.8/site-packages/pandas/core/ops/array_ops.py in logical_op(left, right, op)
```

```
    353         filler = fill_int if is_self_int_dtype and is_other_int_dtype else fill_bool
    354
--> 355         res_values = na_logical_op(lvalues, rvalues, op)
    356         # error: Cannot call function of unknown type
    357         res_values = filler(res_values)  # type: ignore[operator]

/opt/anaconda3/lib/python3.8/site-packages/pandas/core/ops/array_ops.py in na_logical_op(x, y, op)
    286             ) as err:
    287                 typ = type(y).__name__
--> 288                 raise TypeError(
    289                     f"Cannot perform '{op.__name__}' with a dtyped [{x.dtype}] array "
    290                     f"and scalar of type [{typ}]"

TypeError: Cannot perform 'ror_' with a dtyped [bool] array and scalar of type [NoneType]
```

In [ ]:
```
data = ecom[~((ecom < (Q1 - 1.5 * IQR)) |(ecom > (Q3 + 1.5 * IQR))).any(axis=1)]
```

In [ ]:
```
ecom.head(10)
```

In [ ]:
```
ecom = list(ecom.columns)
```

In [ ]:
```
ecom
```

In [ ]:
```
ecom.remove('Summary')
```

In [ ]:
```
print(ecom)
```

In [ ]:
```
col_list = []
for col in ecom.columns:
    if ((ecom[col].dtype != 'object') & (col != 'y') ):
        col_list.append(col)
for i in range(len(X.columns))
X = ecom[col_list]
vif_ecom = pd.DataFrame()
vif_ecom["StockCode"] = X.columns
vif_ecom["VIF"] = [variance_inflation_factor(X.values, i)
                   print(vif_ecom)
```

In [ ]:
```
from sklearn.preprocessing import labelEncoder
labelencoder = labelEncoder()
for i in ecom:
    ecom[i]=labelencoder.fit_transform(ecom[i])
```

In [ ]:
```
import pandas as pd
```

In [ ]:
```
ecom
```

In [31]:
```
y = pd.DataFrame(ecom['HelpfulnessDenominator'])
```

In [32]:
```
y
```

Out[32]:

	HelpfulnessDenominator
0	1303862400
1	1346976000
2	1219017600
3	1307923200
4	1350777600
...	...

568449	1299628800
568450	1331251200
568451	1329782400
568452	1331596800
568453	1338422400

568454 rows × 1 columns

In [33]:
```
x = pd.DataFrame(ecom['HelpfulnessNumerator'])
```

In [35]:
```
x
```

Out[35]:

	HelpfulnessNumerator
0	1
1	0
2	1
3	3
4	0
...	...
568449	0
568450	0
568451	2
568452	1
568453	0

568454 rows × 1 columns

In [36]:
```
from sklearn.linear_model import LinearRegression
```

In [38]:
```
lin_fit=LinearRegression()
```

In [39]:
```
from sklearn.model_selection import train_test_split
```

In [46]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.39,random_state=45)
```

In [47]:
```
lin_reg=LinearRegression().fit(x_train,y_train)
lin_reg
```

Out[47]:
```
LinearRegression()
```

In [48]:
```
x_pred=lin_reg.predict(y_test)
```

In [49]:
```
x_pred
```

Out[49]:
```
array([[-1.24708258e+15],
       [-1.23273303e+15],
       [-1.11553120e+15],
       ...,
       [-1.26616002e+15],
       [-1.18794252e+15],
       [-1.28706226e+15]])
```

In [44]:
```
y_pred=lin_reg.predict(x_test)
```

In [50]:
```
y_pred
```

Out[50]:
```
array([[1.29806762e+09],
       [1.29806762e+09],
       [1.29806762e+09],
       ...,
       [1.29396519e+09],
       [1.28165789e+09],
       [1.29704201e+09]])
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js