```
import kagglehub
aliiihussain_amazon_sales_dataset_path = kagglehub.dataset_download('aliiihussain/amazon-sales-dataset')

print('Data source import complete.')
```

```



import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)



import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version u
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/amazon-sales-dataset/amazon_sales_dataset.csv
```

UNDERSTANDING THE DATA

```
import pandas as pd
import numpy as np
```

```
df=pd.read_csv("/kaggle/input/amazon-sales-dataset/amazon_sales_dataset.csv")
```

```
df
```

	order_id	order_date	product_id	product_category	price	discount_percent	quantity_sold	customer_region	payment_meth
0	1	2022-04-13	2637	Books	128.75	10	4	North America	U
1	2	2023-03-12	2300	Fashion	302.60	20	5	Asia	Credit Ca
2	3	2022-09-28	3670	Sports	495.80	20	2	Europe	U
3	4	2022-04-17	2522	Books	371.95	15	4	Middle East	U
4	5	2022-03-13	1717	Beauty	201.68	0	4	Middle East	U
...	...	...	...	...	...	...	...	...	
49995	49996	2022-09-03	1433	Beauty	26.99	0	5	Middle East	Credit Ca
49996	49997	2022-07-03	1428	Beauty	294.23	10	5	Asia	Credit Ca
49997	49998	2023-02-17	4651	Electronics	352.11	30	4	Asia	Debit Ca
49998	49999	2022-09-30	4371	Beauty	307.54	5	1	Middle East	U
49999	50000	2023-06-29	2944	Home & Kitchen	253.44	30	1	Europe	Debit Ca

50000 rows × 13 columns

```
df.head()
```

	order_id	order_date	product_id	product_category	price	discount_percent	quantity_sold	customer_region	payment_method
0	1	2022-04-13	2637	Books	128.75	10	4	North America	UPI
1	2	2023-03-12	2300	Fashion	302.60	20	5	Asia	Credit Card
2	3	2022-09-28	3670	Sports	495.80	20	2	Europe	UPI
3	4	2022-04-17	2522	Books	371.95	15	4	Middle East	UPI
4	5	2022-03-13	1717	Beauty	201.68	0	4	Middle East	UPI

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   order_id          50000 non-null  int64  
 1   order_date        50000 non-null  object 
 2   product_id        50000 non-null  int64  
 3   product_category  50000 non-null  object 
 4   price             50000 non-null  float64
 5   discount_percent  50000 non-null  int64  
 6   quantity_sold     50000 non-null  int64  
 7   customer_region   50000 non-null  object 
 8   payment_method    50000 non-null  object 
 9   rating            50000 non-null  float64
 10  review_count      50000 non-null  int64  
 11  discounted_price  50000 non-null  float64
 12  total_revenue     50000 non-null  float64
dtypes: float64(4), int64(5), object(4)
memory usage: 5.0+ MB
```

```
import pandas as pd
df['order_date'] = pd.to_datetime(df['order_date'])
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   order_id          50000 non-null  int64  
 1   order_date        50000 non-null  object 
 2   product_id        50000 non-null  int64  
 3   product_category  50000 non-null  object 
 4   price             50000 non-null  float64
 5   discount_percent  50000 non-null  int64  
 6   quantity_sold     50000 non-null  int64  
 7   customer_region   50000 non-null  object 
 8   payment_method    50000 non-null  object 
 9   rating            50000 non-null  float64
 10  review_count      50000 non-null  int64  
 11  discounted_price  50000 non-null  float64
 12  total_revenue     50000 non-null  float64
dtypes: float64(4), int64(5), object(4)
memory usage: 5.0+ MB
```

```
df.isnull().sum()
```

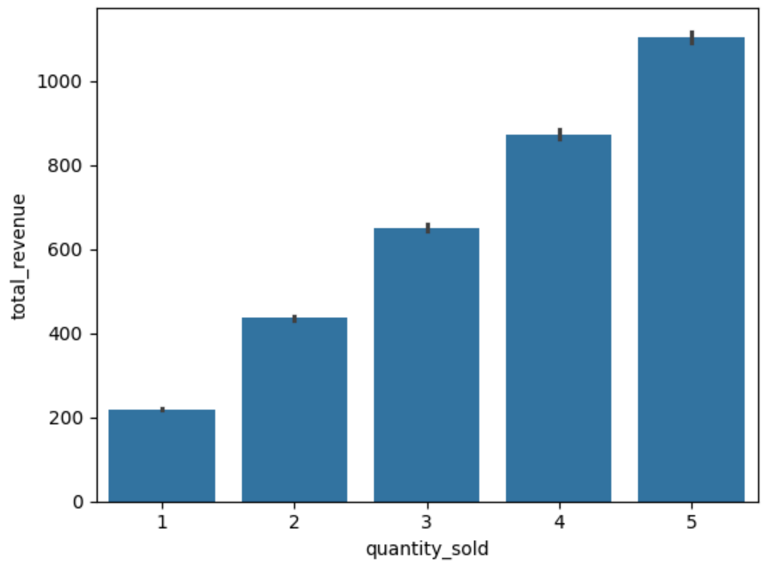
```
order_id            0
order_date          0
product_id          0
product_category    0
price               0
discount_percent    0
quantity_sold       0
customer_region     0
payment_method      0
rating              0
review_count        0
discounted_price    0
total_revenue       0
dtype: int64
```

```
df['product_category'].value_counts()
```

```
product_category
Beauty            8465
Fashion           8365
Books             8327
Electronics       8320
Sports            8265
Home & Kitchen    8258
Name: count, dtype: int64
```

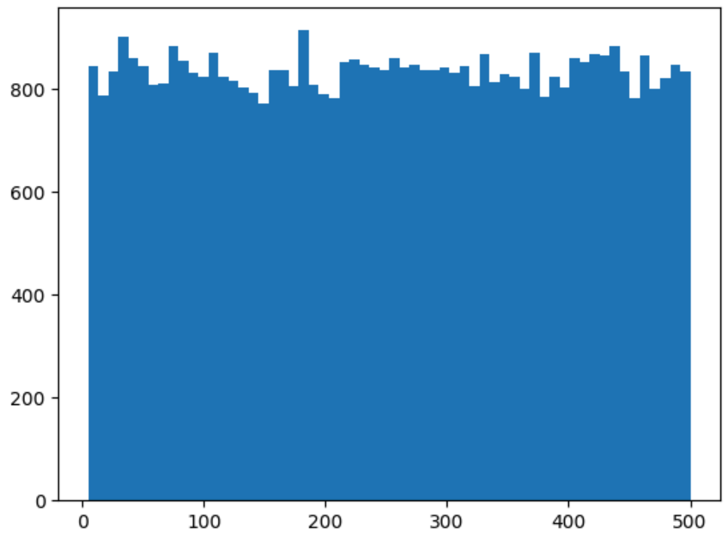
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.barplot(x=df['quantity_sold'],y=df['total_revenue'] )
```

```
<Axes: xlabel='quantity_sold', ylabel='total_revenue'>
```



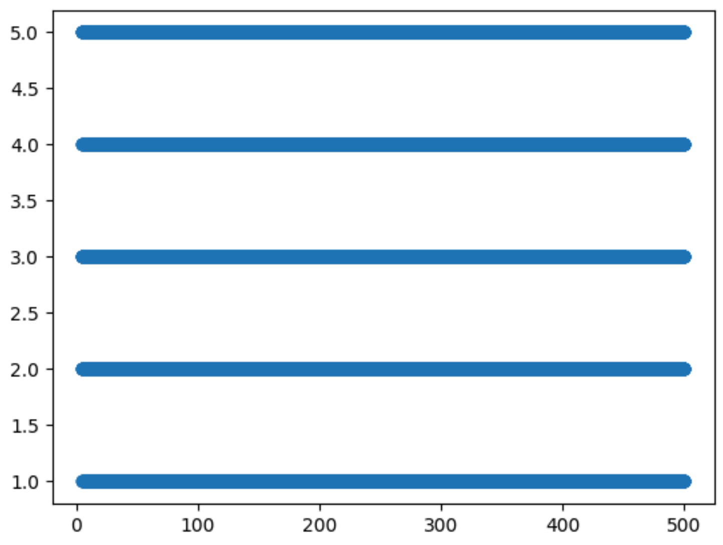
```
plt.hist(df['price'],bins=60)
```

```
(array([843., 786., 834., 902., 858., 844., 808., 810., 883., 854., 832.,
        824., 870., 824., 815., 802., 792., 772., 837., 837., 806., 913.,
        807., 789., 781., 851., 856., 847., 840., 836., 859., 841., 847.,
        835., 837., 840., 832., 844., 805., 866., 812., 828., 823., 799.,
        869., 785., 823., 802., 858., 851., 866., 864., 882., 833., 782.,
        865., 799., 821., 846., 833.]),
 array([  5.01      ,  13.25966667,  21.50933333,  29.759     ,
         38.00866667,  46.25833333,  54.508     ,  62.75766667,
         71.00733333,  79.257     ,  87.50666667,  95.75633333,
        104.006     , 112.25566667, 120.50533333, 128.755     ,
        137.00466667, 145.25433333, 153.504     , 161.75366667,
        170.00333333, 178.253     , 186.50266667, 194.75233333,
        203.002     , 211.25166667, 219.50133333, 227.751     ,
        236.00066667, 244.25033333, 252.5       , 260.74966667,
        268.99933333, 277.249     , 285.49866667, 293.74833333,
        301.998     , 310.24766667, 318.49733333, 326.747     ,
        334.99666667, 343.24633333, 351.496     , 359.74566667,
        367.99533333, 376.245     , 384.49466667, 392.74433333,
        400.994     , 409.24366667, 417.49333333, 425.743     ,
        433.99266667, 442.24233333, 450.492     , 458.74166667,
        466.99133333, 475.241     , 483.49066667, 491.74033333,
        499.99      ]),
 <BarContainer object of 60 artists>)
```



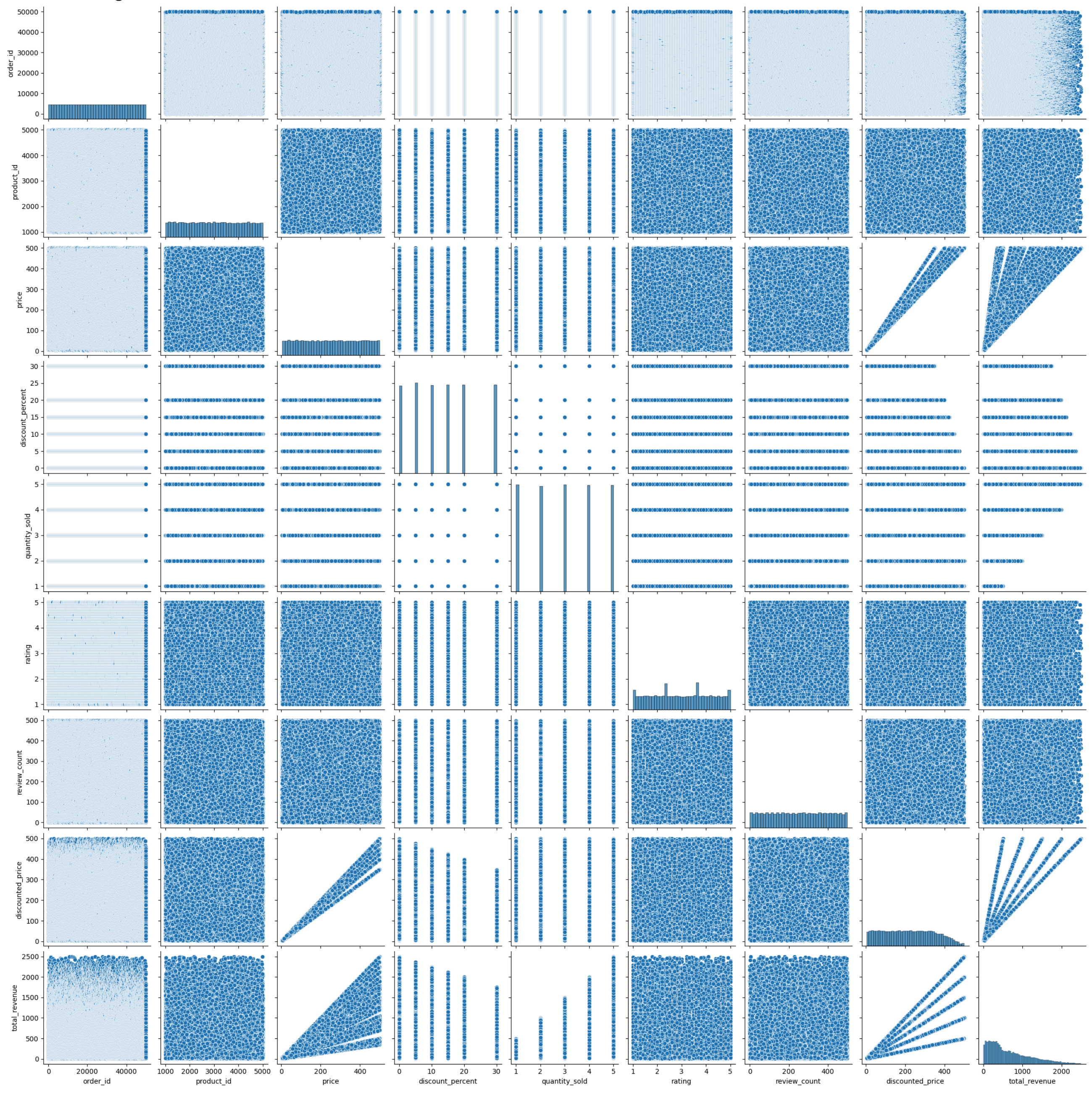
```
plt.scatter(df['price'],df['quantity_sold'])
```

```
<matplotlib.collections.PathCollection at 0x788734688110>
```



```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x788734a60ec0>
```

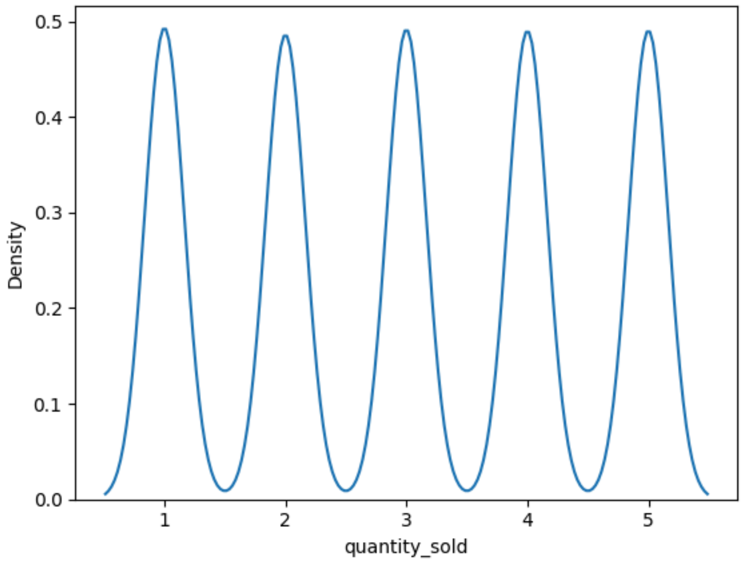

order_id
product_id
price
discount_percent
quantity_sold
rating
review_count
discounted_price
total_revenue

```
df.describe()
```

	order_id	product_id	price	discount_percent	quantity_sold	rating	review_count	discounted_price	to
count	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	5
mean	25000.500000	2986.848740	252.507260	13.340700	2.999400	2.996316	249.329280	218.886566	
std	14433.901067	1156.374535	143.025544	9.850694	1.415401	1.154295	144.251981	127.317681	
min	1.000000	1000.000000	5.010000	0.000000	1.000000	1.000000	0.000000	3.530000	
25%	12500.750000	1983.000000	127.840000	5.000000	2.000000	2.000000	125.000000	109.680000	
50%	25000.500000	2983.000000	252.970000	10.000000	3.000000	3.000000	250.000000	215.805000	
75%	37500.250000	3989.000000	376.335000	20.000000	4.000000	4.000000	374.000000	322.702500	
max	50000.000000	4999.000000	499.990000	30.000000	5.000000	5.000000	499.000000	499.910000	

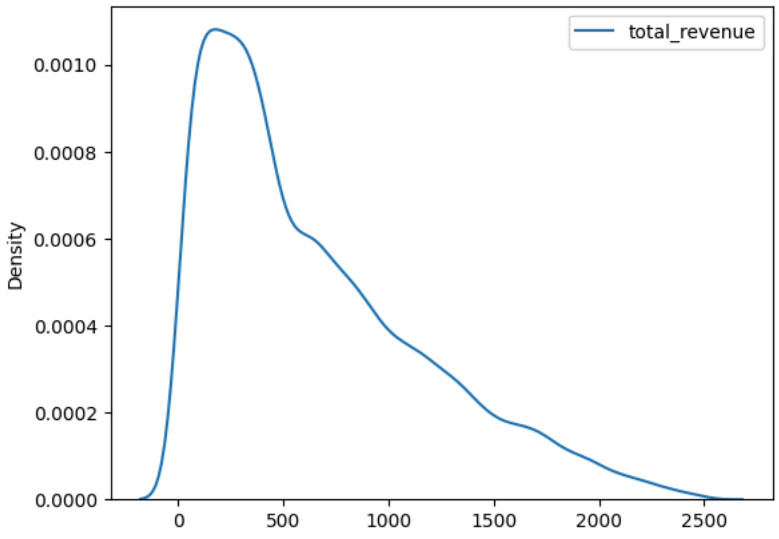
```
sns.kdeplot(df['quantity_sold'])
```

```
<Axes: xlabel='quantity_sold', ylabel='Density'>
```



```
sns.kdeplot([df['total_revenue']])
```

```
<Axes: ylabel='Density'>
```



FEATURE TRANSFORMATION

[A] FEATURE SCALING

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split( df[['quantity_sold', 'price']], df['rating'],test_size=0.2,random_state=10)
```

```
x_train
```

	quantity_sold	price
20433	2	56.68
28866	2	44.59
42817	2	270.33
9543	2	416.77
20619	5	366.65
...	...	...
40059	3	66.31
28017	1	282.77
29199	4	157.15
40061	5	105.35
17673	1	249.09

40000 rows × 2 columns

1)STANDARDIZATION

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(x_train)
x_train_scaled=scaler.transform(x_train)
x_test_scaled=scaler.transform(x_test)
```

```
df.describe()
```

	order_id	product_id	price	discount_percent	quantity_sold	rating	review_count	discounted_price	to
count	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	5
mean	25000.500000	2986.848740	252.507260	13.340700	2.999400	2.996316	249.329280	218.886566	
std	14433.901067	1156.374535	143.025544	9.850694	1.415401	1.154295	144.251981	127.317681	
min	1.000000	1000.000000	5.010000	0.000000	1.000000	1.000000	0.000000	3.530000	
25%	12500.750000	1983.000000	127.840000	5.000000	2.000000	2.000000	125.000000	109.680000	
50%	25000.500000	2983.000000	252.970000	10.000000	3.000000	3.000000	250.000000	215.805000	
75%	37500.250000	3989.000000	376.335000	20.000000	4.000000	4.000000	374.000000	322.702500	
max	50000.000000	4999.000000	499.990000	30.000000	5.000000	5.000000	499.000000	499.910000	

2)Normalization

```
# min max scaling most useful
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
scaler.fit(x_train)
x_train_scaler=scaler.transform(x_train)
x_test_scaler=scaler.transform(x_test)
```

```
df.describe()
```

	order_id	order_date	product_id	price	discount_percent	quantity_sold	rating	review_count	d:
count	50000.000000	50000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	
mean	25000.500000	2022-12-31 10:47:16.800000256	2986.848740	252.507260	13.340700	2.999400	2.996316	249.329280	
min	1.000000	2022-01-01 00:00:00	1000.000000	5.010000	0.000000	1.000000	1.000000	0.000000	
25%	12500.750000	2022-07-02 00:00:00	1983.000000	127.840000	5.000000	2.000000	2.000000	125.000000	
50%	25000.500000	2023-01-02 00:00:00	2983.000000	252.970000	10.000000	3.000000	3.000000	250.000000	
75%	37500.250000	2023-07-02 00:00:00	3989.000000	376.335000	20.000000	4.000000	4.000000	374.000000	
max	50000.000000	2023-12-31 00:00:00	4999.000000	499.990000	30.000000	5.000000	5.000000	499.000000	
std	14433.901067	NaN	1156.374535	143.025544	9.850694	1.415401	1.154295	144.251981	

[B] Encoding categorical

```
df
```

```
The history saving thread hit an unexpected error (OperationalError('attempt to write a readonly database')).History will not be
```

	order_id	order_date	product_id	product_category	price	discount_percent	quantity_sold	customer_region	payment_meth
0	1	2022-04-13	2637	Books	128.75	10	4	North America	U
1	2	2023-03-12	2300	Fashion	302.60	20	5	Asia	Credit Ca
2	3	2022-09-28	3670	Sports	495.80	20	2	Europe	U
3	4	2022-04-17	2522	Books	371.95	15	4	Middle East	U
4	5	2022-03-13	1717	Beauty	201.68	0	4	Middle East	U
...	...	...	...	...	...	...	...	...	
49995	49996	2022-09-03	1433	Beauty	26.99	0	5	Middle East	Credit Ca
49996	49997	2022-07-03	1428	Beauty	294.23	10	5	Asia	Credit Ca
49997	49998	2023-02-17	4651	Electronics	352.11	30	4	Asia	Debit Ca
49998	49999	2022-09-30	4371	Beauty	307.54	5	1	Middle East	U
49999	50000	2023-06-29	2944	Home & Kitchen	253.44	30	1	Europe	Debit Ca

50000 rows × 13 columns

```
df.isnull().sum()
```

```
order_id            0
order_date          0
product_id          0
product_category    0
price               0
discount_percent    0
quantity_sold       0
customer_region     0
payment_method      0
rating              0
review_count        0
discounted_price    0
total_revenue       0
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 13 columns):
```