Double-click (or enter) to edit

Import the libraries required for the Project

Double-click (or enter) to edit

```
import pandas as pd
import numpy as np
import matplotlib.pyplot  as plt
import seaborn as sns
```

Read CSV file into dataframe

```
df=pd.read_csv("/content/E-commerce Dataset.csv")
```

Get the information of the Data

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51290 entries, 0 to 51289
Data columns (total 16 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Order_Date           51290 non-null  object 
 1   Time                 51290 non-null  object 
 2   Aging                51289 non-null  float64
 3   Customer_Id          51290 non-null  int64  
 4   Gender               51290 non-null  object 
 5   Device_Type          51290 non-null  object 
 6   Customer_Login_type  51290 non-null  object 
 7   Product_Category     51290 non-null  object 
 8   Product              51290 non-null  object 
 9   Sales                51289 non-null  float64
 10  Quantity             51288 non-null  float64
 11  Discount             51289 non-null  float64
 12  Profit               51290 non-null  float64
 13  Shipping_Cost        51289 non-null  float64
 14  Order_Priority       51288 non-null  object 
 15  Payment_method       51290 non-null  object 
dtypes: float64(6), int64(1), object(9)
memory usage: 6.3+ MB
```

Check how many rows and columns

```
df.shape
```

```
(51290, 18)
```

Quick glimpse of data of five rows

```
df.head()
```

	Order_Date	Time	Aging	Customer_Id	Gender	Device_Type	Customer_Login_type	Product_Category	Product	Sales	Quantity	Discount	Profit	Shipping_Cost	Order_Priority	Payment_me
0	2018-01-02	10:56:33	8.0	37077	Female	Web	Member	Auto & Accessories	Car Media Players	140.0	1.0	0.3	46.0	4.6	Medium	credit_
1	2018-07-24	20:41:37	2.0	59173	Female	Web	Member	Auto & Accessories	Car Speakers	211.0	1.0	0.3	112.0	11.2	Medium	credit_
2	2018-11-08	08:38:49	8.0	41066	Female	Web	Member	Auto & Accessories	Car Body Covers	117.0	5.0	0.1	31.2	3.1	Critical	credit_
3	2018-04-18	19:28:06	7.0	50741	Female	Web	Member	Auto & Accessories	Car & Bike Care	118.0	1.0	0.3	26.2	2.6	High	credit_
4	2018-08-13	21:18:39	9.0	53639	Female	Web	Member	Auto & Accessories	Tyre	250.0	1.0	0.3	160.0	16.0	Critical	credit_

#Summary statistics for the numerical columns of a DataFrame

```
df.describe()
```

	Aging	Customer_Id	Sales	Quantity	Discount	Profit	Shipping_Cost
count	51289.000000	51290.000000	51289.000000	51288.000000	51289.000000	51290.000000	51289.000000
mean	5.255035	58155.758764	152.340872	2.502983	0.303821	70.407226	7.041557
std	2.959948	26032.215826	66.495419	1.511859	0.131027	48.729488	4.871745
min	1.000000	10000.000000	33.000000	1.000000	0.100000	0.500000	0.100000
25%	3.000000	35831.250000	85.000000	1.000000	0.200000	24.900000	2.500000
50%	5.000000	61018.000000	133.000000	2.000000	0.300000	59.900000	6.000000
75%	8.000000	80736.250000	218.000000	4.000000	0.400000	118.400000	11.800000
max	10.500000	99999.000000	250.000000	5.000000	0.500000	167.500000	16.800000

Check The Columns in a Data Set

```
df.columns
```

```
Index(['Order_Date', 'Time', 'Aging', 'Customer_Id', 'Gender', 'Device_Type',
       'Customer_Login_type', 'Product_Category', 'Product', 'Sales',
       'Quantity', 'Discount', 'Profit', 'Shipping_Cost', 'Order_Priority',
       'Payment_method'],
      dtype='object')
```

Double-click (or enter) to edit

```
df['Payment_method']
```

```
0        credit_card
1        credit_card
2        credit_card
3        credit_card
4        credit_card
            ...
51285    money_order
51286    credit_card
51287    credit_card
51288    credit_card
51289    credit_card
Name: Payment_method, Length: 51290, dtype: object
```

check the missing values in a data set

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

```
Order_Date            0
Time                  0
Aging                 1
Customer_Id           0
Gender                0
Device_Type           0
Customer_Login_type   0
Product_Category      0
Product               0
Sales                 1
Quantity              2
Discount              1
Profit                0
Shipping_Cost         1
Order_Priority        2
Payment_method        0
dtype: int64
```

Import the Simple Imputator

```
from sklearn.impute import SimpleImputer
```

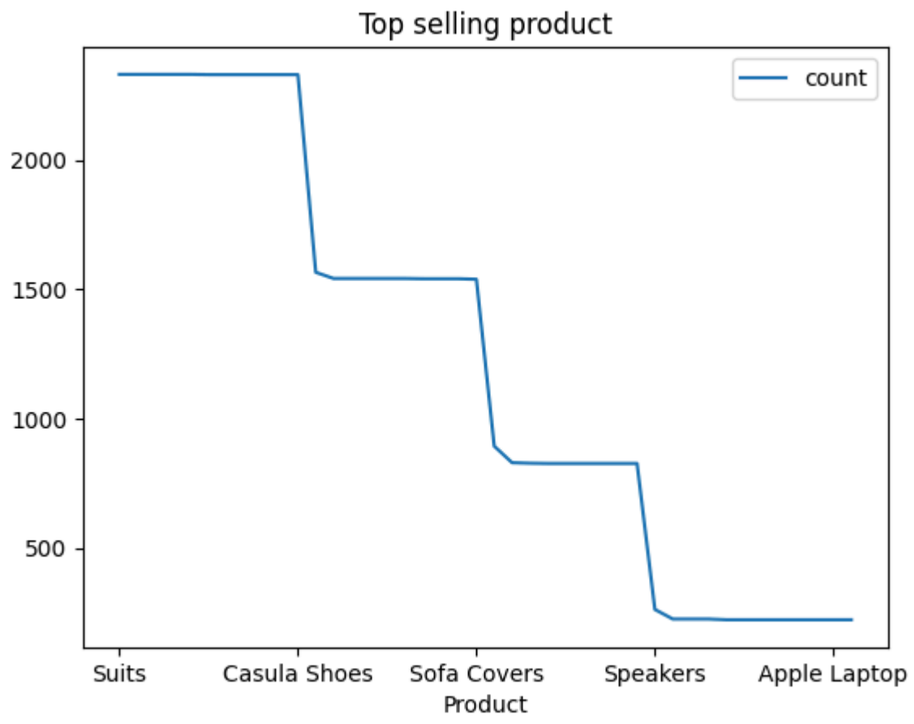
Replace missing values with mean,median or with NAN

```
def parameter(df):
  for col in df.columns[df.isnull().any()]:
    if df[col].dtype=='int64' or df[col].dtype=='float32' or df[col].dtype=='int32':
      strategy='mean'
    else:
      strategy='most_frequent'
    df[col] = df[col].astype(str) if strategy == 'most_frequent' else df[col]
    imputer=SimpleImputer(strategy=strategy)
    # The following line has been updated to correctly select the column as a 2D array
    df[col]=imputer.fit_transform(df[[col]]).ravel()
  return df
df_cleaned=parameter(df)
df_cleaned.isnull().sum()
```

```
Order_Date            0
Time                  0
Aging                 0
Customer_Id           0
Gender                0
Device_Type           0
Customer_Login_type   0
Product_Category      0
Product               0
Sales                 0
Quantity              0
Discount              0
Profit                0
Shipping_Cost         0
Order_Priority        0
Payment_method        0
dtype: int64
```

Objective: Identify which products are sold the most.

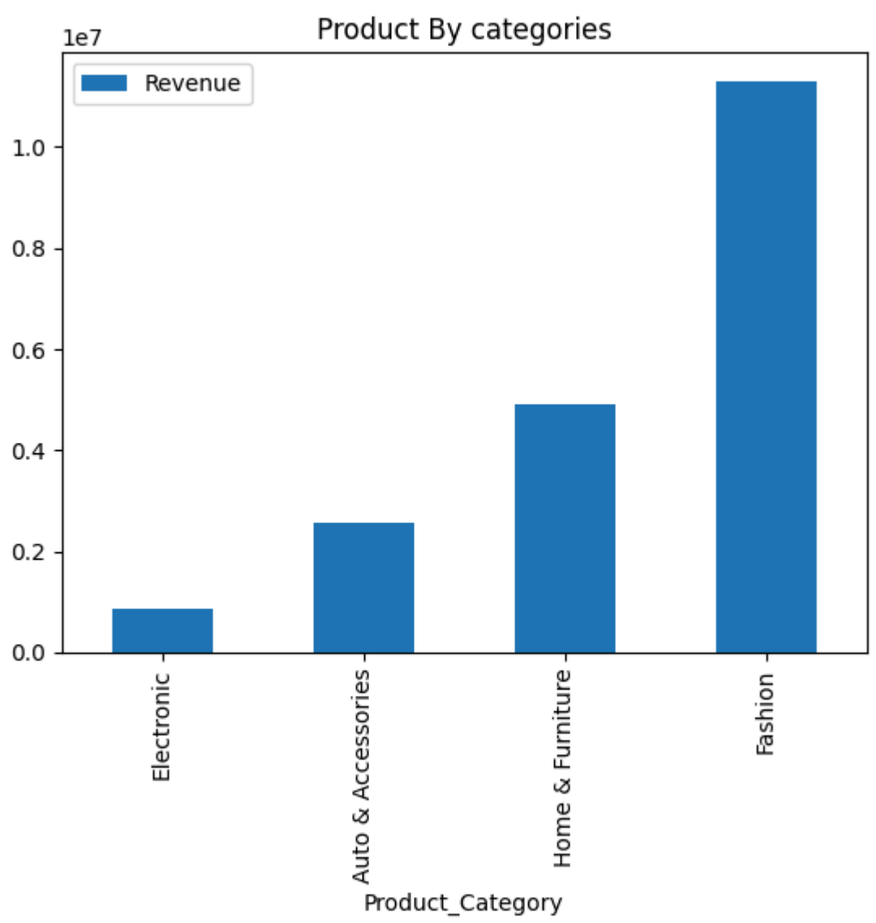
```
top_selling_products=df['Product'].value_counts()
top_selling_products
top_selling_products.plot(kind='line',title="Top selling product")
top_selling_products.ylabel=('Product')
plt.legend()
plt.show()
```



Which product categories generate the most revenue?

```
# Convert 'Quantity' and 'Sales' to numeric values
df['Quantity'] = pd.to_numeric(df['Quantity'], errors='coerce')
df['Sales'] = pd.to_numeric(df['Sales'], errors='coerce')

# Now calculate the revenue
df['Revenue'] = df['Quantity'] * df['Sales']

# Group by 'Product_Category' and sum the revenue
revenue_by_category = df.groupby('Product_Category')['Revenue'].sum().sort_values(ascending=True)

print(revenue_by_category)
revenue_by_category.plot(kind='bar',title="Product By categories")
revenue_by_category.ylabel=('Revenue')
plt.legend()

plt.show()
```

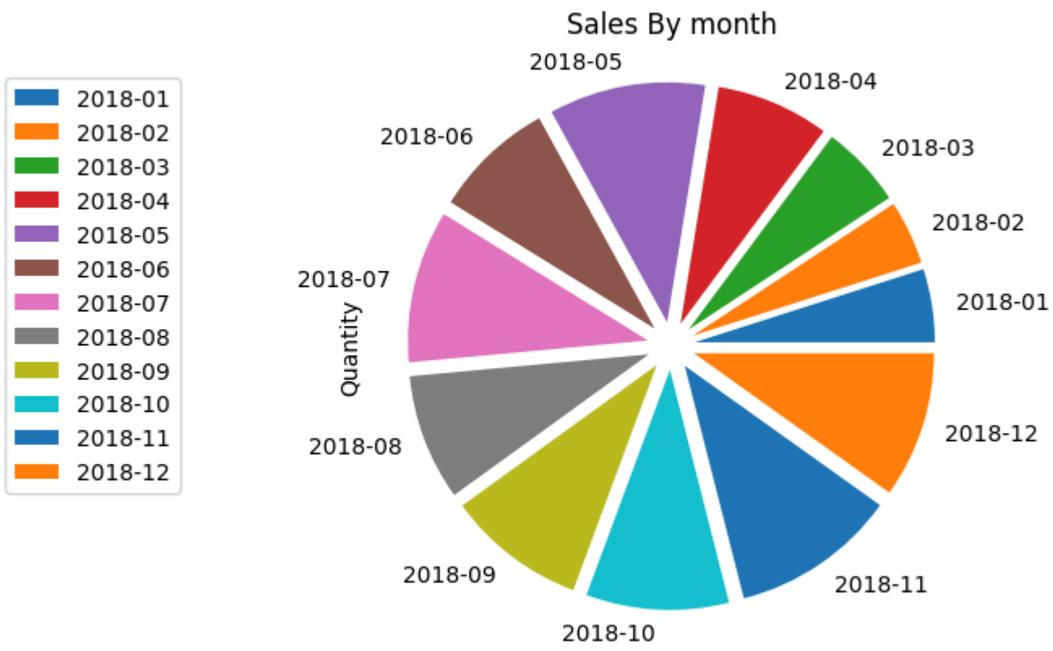
```
Product_Category
Electronic              868178.0
Auto & Accessories     2576442.0
Home & Furniture       4892455.0
Fashion               11298885.0
Name: Revenue, dtype: float64
```



What are the peak sales times or seasons?

```
df['Order_Date']=pd.to_datetime(df['Order_Date'])
sales_by_month = df.groupby(df['Order_Date'].dt.to_period('M'))['Quantity'].sum()
print(sales_by_month)
sales_by_month.plot(kind='pie',title="Sales By month",explode=[0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1])
plt.legend(loc='center right',bbox_to_anchor=(-0.3, 0.6))
plt.show()
```

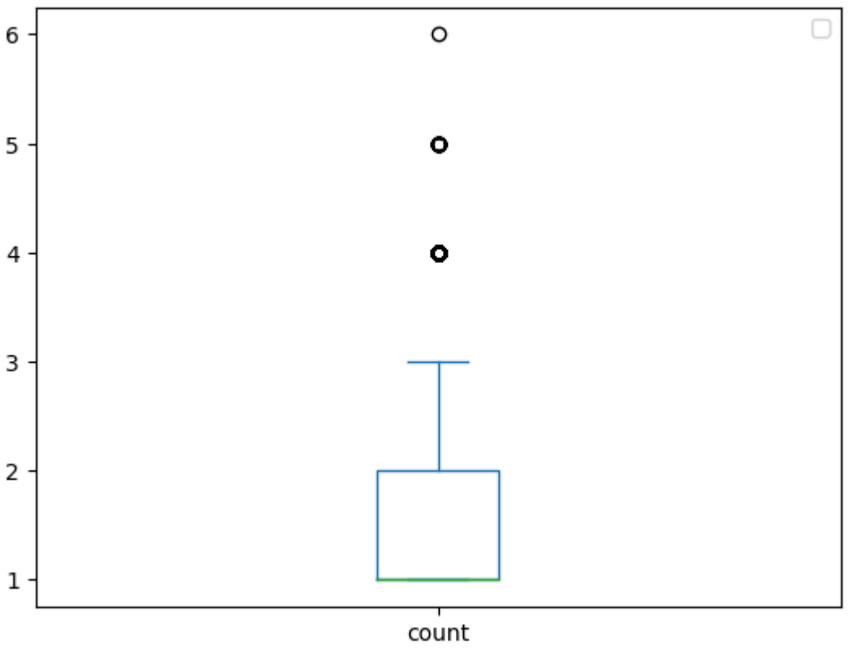
```
Order_Date
2018-01     6324.0
2018-02     5539.0
2018-03     7168.0
2018-04     9795.0
2018-05    13536.0
2018-06    10498.0
2018-07    13197.0
2018-08    10916.0
2018-09    12049.0
2018-10    12346.0
2018-11    14344.0
2018-12    12661.0
Freq: M, Name: Quantity, dtype: float64
```



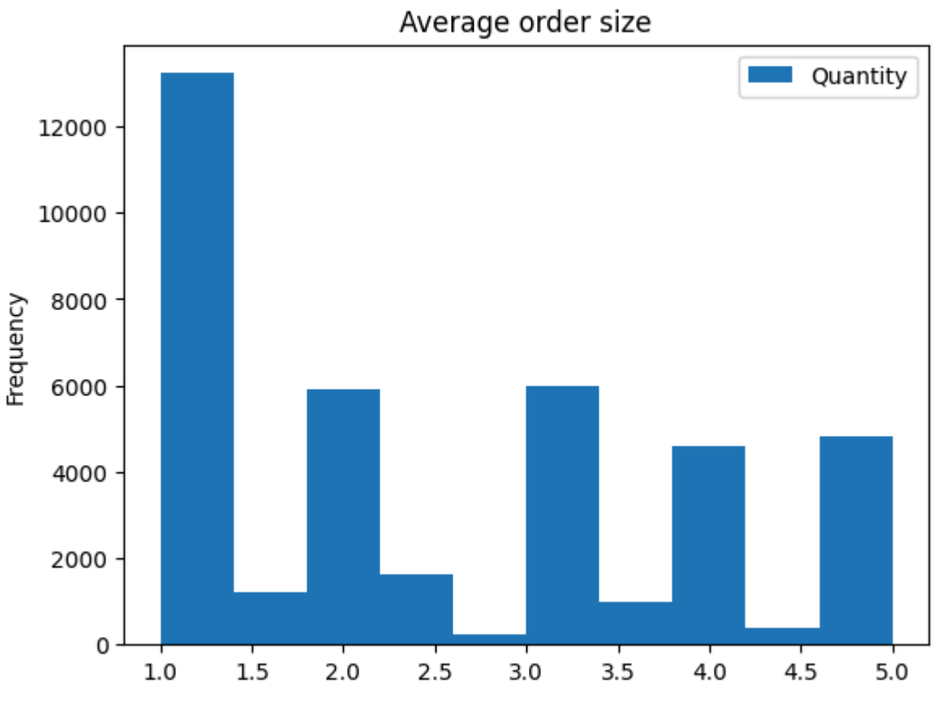
What are the customer purchasing patterns (e.g., repeat purchases, average order size)

```
 #Repeated customers
repeated_customers=df['Customer_Id'].value_counts()
repeated_customers
repeated_customers.plot(kind='box')
plt.legend()
plt.show()
```

```
WARNING:matplotlib.legend:No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignored when legend() is called with no argument.
```




```
average_order_size=df.groupby('Customer_Id')['Quantity'].mean()
average_order_size
average_order_size.plot(kind='hist',title="Average order size")
plt.legend()
plt.show()
```



Double-click (or enter) to edit

Most frequent payment methods

```
payment_methods=df.groupby(df['Payment_method'])['Payment_method'].count()
payment_methods
```

```
Payment_method
credit_card    38137
debit_card       734
e_wallet        2789
money_order     9629
not_defined        1
Name: Payment_method, dtype: int64
```

Data Encoding:From sklearnpreprocessing import OneHotEncoding

```
from sklearn.preprocessing import OneHotEncoder
```

Convert The categorical clomuns into Numerical values using Data encoding

```
from sklearn.preprocessing import OneHotEncoder
import pandas as pd

# Select categorical columns
categorical_columns = df.select_dtypes(include='object').columns

# Initialize the OneHotEncoder with sparse output to save memory
encoder = OneHotEncoder(sparse_output=True)

# Apply OneHotEncoder to categorical columns
encoded = encoder.fit_transform(df[categorical_columns])

# Convert sparse matrix to DataFrame
encoded_df = pd.DataFrame.sparse.from_spmatrix(encoded, columns=encoder.get_feature_names_out(categorical_columns))

# Drop the original categorical columns
df_dropped = df.drop(columns=categorical_columns)

# Concatenate the encoded columns back to the DataFrame
df_encoded = pd.concat([df_dropped, encoded_df], axis=1)

# Display the first few rows of the cleaned and encoded DataFrame
df_encoded.head()
```

	Aging	Customer_Id	Sales	Quantity	Discount	Profit	Shipping_Cost	Order_Date_2018-01-01	Order_Date_2018-01-02	Order_Date_2018-01-03	...	Order_Priority_Critical	Order_Priority_High	Order_Prior
0	8.0	37077	140.0	1.0	0.3	46.0	4.6	0.0	1.0	0.0	...	0.0	0.0	
1	2.0	59173	211.0	1.0	0.3	112.0	11.2	0.0	0.0	0.0	...	0.0	0.0	
2	8.0	41066	117.0	5.0	0.1	31.2	3.1	0.0	0.0	0.0	...	1.0	0.0	
3	7.0	50741	118.0	1.0	0.3	26.2	2.6	0.0	0.0	0.0	...	0.0	1.0	