```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
```

```
dataset = pd.read_csv('Daily Household Transactions.csv')
```

```
dataset.head()
```

	Date	Mode	Category	Subcategory	Note	Amount	Income/Expense	Currency
0	20/09/2018 12:04:08	Cash	Transportation	Train	2 Place 5 to Place 0	30.0	Expense	INR
1	20/09/2018 12:03:15	Cash	Food	snacks	Idli medu Vada mix 2 plates	60.0	Expense	INR
2	19/09/2018	Saving Bank account 1	subscription	Netflix	1 month subscription	199.0	Expense	INR
3	17/09/2018 23:41:17	Saving Bank account 1	subscription	Mobile Service Provider	Data booster pack	19.0	Expense	INR
4	16/09/2018 17:15:08	Cash	Festivals	Ganesh Pujan	Ganesh idol	251.0	Expense	INR

```
dataset.tail()
```

	Date	Mode	Category	Subcategory	Note	Amount	Income/Expense	Currency
594	23/01/2018 21:28:59	Credit Card	Household	Kirana	Smart point	273.0	Expense	INR
595	22/01/2018 19:05:26	Saving Bank account 1	Food	flour mill	2 kg Bajari	14.0	Expense	INR
596	22/01/2018 12:08:08	Cash	Food	flour mill	M D sure 3kg atta	162.0	Expense	INR
597	22/01/2018 10:25:23	Saving Bank account 1	Health	Medicine	Cough-sills 4 pcs + sinarest 3 pcs	30.0	Expense	INR
598	22/01/2018 10:25:11	Saving Bank account 1	Food	Milk	1 lit	69.0	Expense	INR

```
dataset.describe()
```

	Amount
count	599.000000
mean	2026.219983
std	8478.642674
min	2.000000
25%	36.000000
50%	77.000000
75%	489.680000
max	70255.000000

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 599 entries, 0 to 598
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Date            599 non-null    object
 1   Mode            599 non-null    object
 2   Category        599 non-null    object
 3   Subcategory     489 non-null    object
 4   Note            495 non-null    object
 5   Amount          599 non-null    float64
 6   Income/Expense  599 non-null    object
 7   Currency        599 non-null    object
dtypes: float64(1), object(7)
memory usage: 37.6+ KB
```

```
dataset.isnull().sum()
```

	0
Date	0
Mode	0
Category	0
Subcategory	110
Note	104
Amount	0
Income/Expense	0
Currency	0

```
dataset.count()
```

	0
Date	599
Mode	599
Category	599
Subcategory	489
Note	495
Amount	599
Income/Expense	599
Currency	599

```
dataset.shape
```

```
(599, 8)
```

```
dataset.dropna(inplace=True)
```

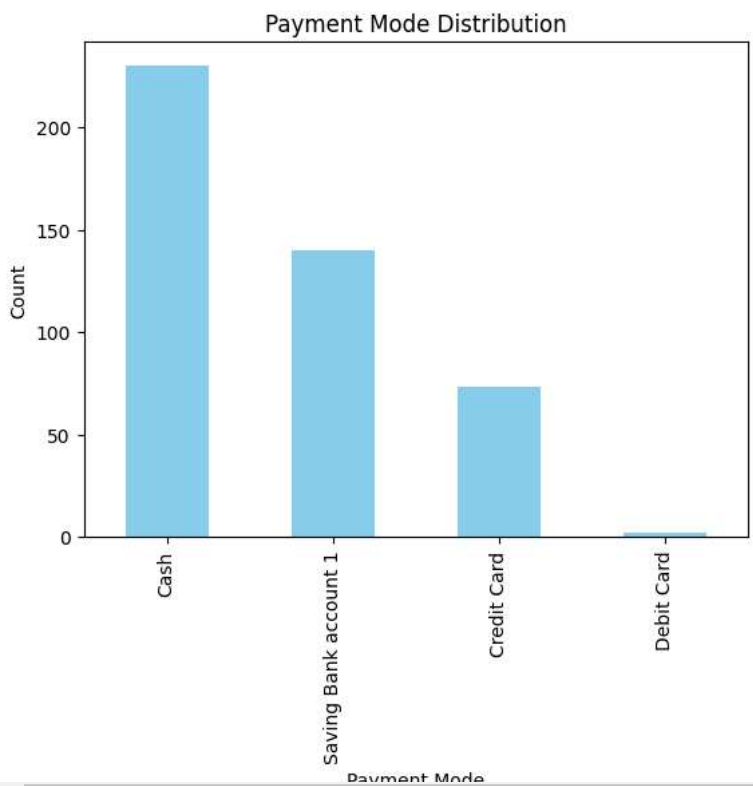
```
dataset.count()
```

	0
Date	445
Mode	445
Category	445
Subcategory	445
Note	445
Amount	445
Income/Expense	445
Currency	445

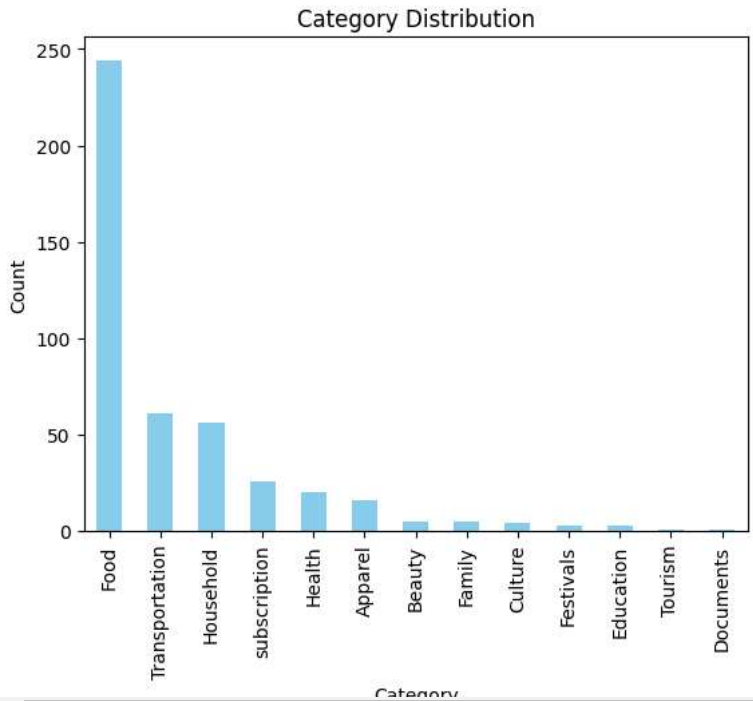
```
dataset.isnull().sum()
```

	0
Date	0
Mode	0
Category	0
Subcategory	0
Note	0
Amount	0
Income/Expense	0
Currency	0

```
payment_mode_counts = dataset['Mode'].value_counts()
payment_mode_counts.plot(kind='bar', color='skyblue')
plt.title('Payment Mode Distribution')
plt.xlabel('Payment Mode')
plt.ylabel('Count')
plt.show()
```

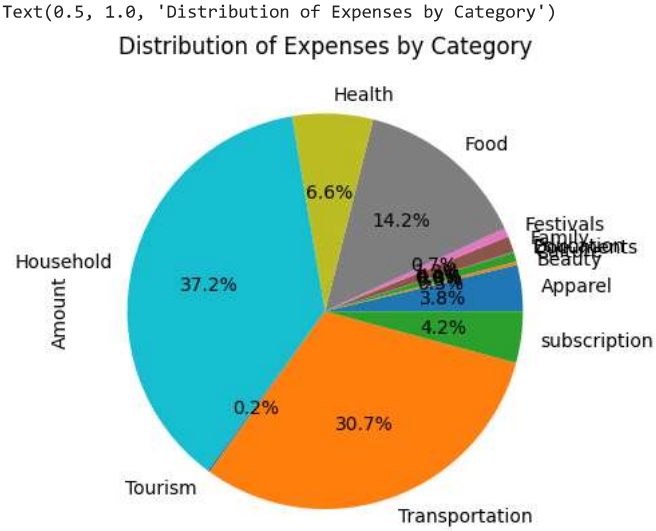



```
category_counts = dataset['Category'].value_counts()
category_counts.plot(kind='bar', color='skyblue')
plt.title('Category Distribution')
plt.xlabel('Category')
plt.ylabel('Count')
plt.show()
```



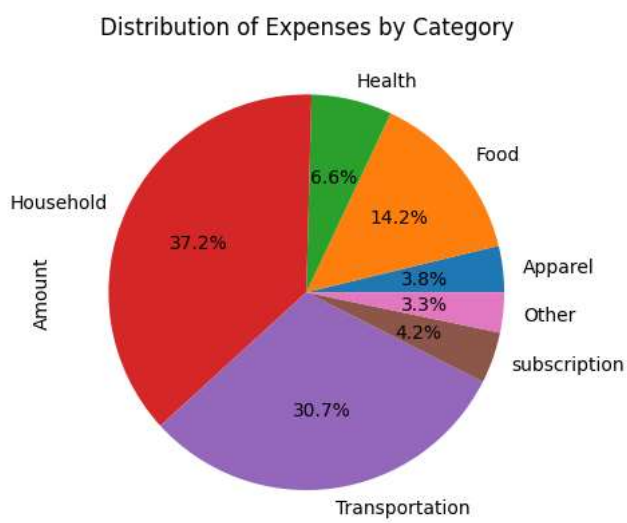

```
dataset.groupby('Category')['Amount'].sum().plot(kind='pie', autopct= '%1.1f%%')
plt.title('Distribution of Expenses by Category')
```

```
Text(0.5, 1.0, 'Distribution of Expenses by Category')
```

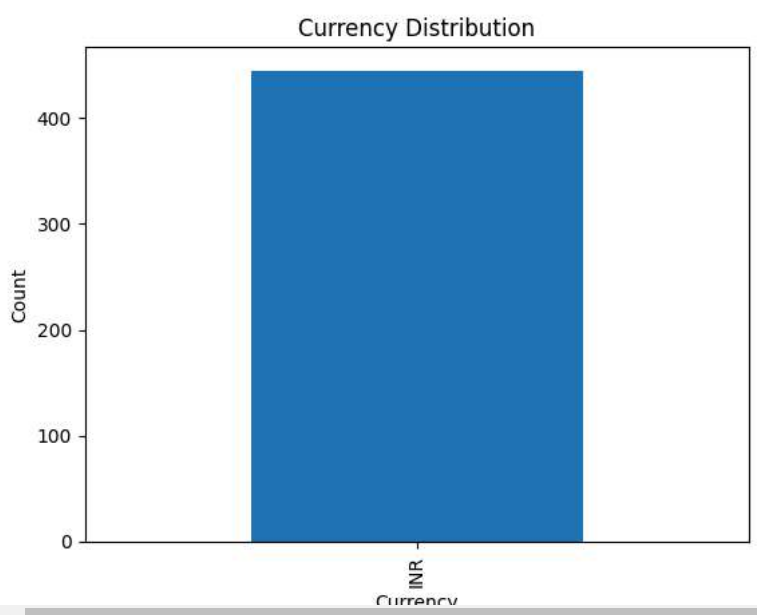



```
category_sums = dataset.groupby('Category')['Amount'].sum()

# Calculate the percentage of each category
category_percentages = category_sums / category_sums.sum() * 100

# Create a new category for values below 0.5%
other_categories = category_percentages[category_percentages < 2].sum()

# Filter categories above 0.5%
main_categories = category_percentages[category_percentages >= 2]

# Add the "Other" category
main_categories['Other'] = other_categories

# Plot the pie chart
main_categories.plot(kind='pie', autopct='%1.1f%%')
plt.title('Distribution of Expenses by Category')
```

```
plt.show()
```




```
curr_count = dataset['Currency'].value_counts()
curr_count.plot(kind='bar')
plt.title('Currency Distribution')
plt.xlabel('Currency')
plt.ylabel('Count')
plt.show()
```




```
selected_columns = ['Amount', 'Category', 'Subcategory']
selected_data = dataset[selected_columns]
```

```
data_encoded = pd.get_dummies(selected_data, columns=['Category', 'Subcategory'])
```

```
from sklearn.preprocessing import StandardScaler
scalar = StandardScaler()
data_Scaled = scalar.fit_transform(data_encoded)
```

```
data_Scaled_df = pd.DataFrame(data_Scaled, columns=data_encoded.columns)

data_Scaled_df.head()
```

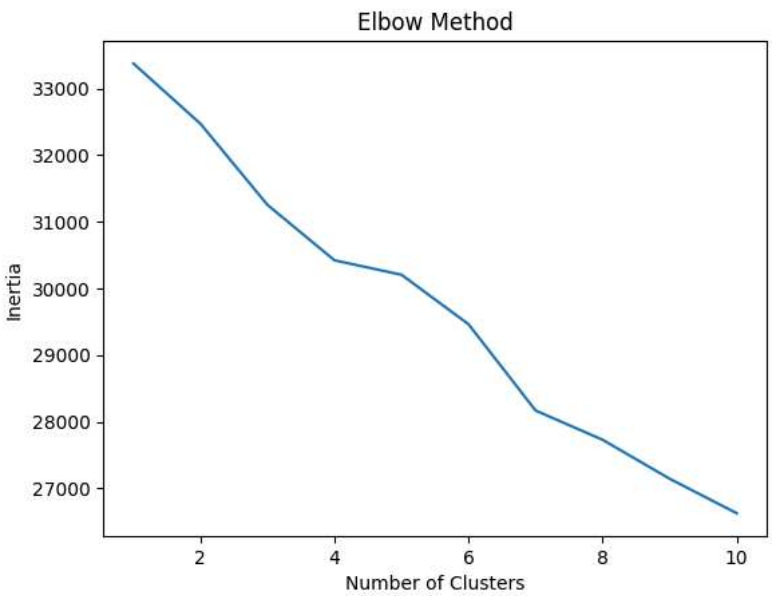
	Amount	Category_Apparel	Category_Beauty	Category_Culture	Category_Documents	Category_Education	Category_Family	Category_Fest
0	-0.154891	-0.193122	-0.1066	-0.095238	-0.047458	-0.082385	-0.1066	-0.0
1	-0.142333	-0.193122	-0.1066	-0.095238	-0.047458	-0.082385	-0.1066	-0.0
2	-0.084149	-0.193122	-0.1066	-0.095238	-0.047458	-0.082385	-0.1066	-0.0
3	-0.159495	-0.193122	-0.1066	-0.095238	-0.047458	-0.082385	-0.1066	-0.0
4	-0.062383	-0.193122	-0.1066	-0.095238	-0.047458	-0.082385	-0.1066	12.1

5 rows × 75 columns

```
inertia = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i,random_state= 0)
    kmeans.fit(data_Scaled_df)
    inertia.append(kmeans.inertia_)

plt.plot(range(1, 11), inertia)
plt.title('Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.show
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 1
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 1
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 1
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 1
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 1
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 1
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 1
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 1
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 1
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 1
  super()._check_params_vs_input(X, default_n_init=10)
```

```
matplotlib.pyplot.show
def show(*args, **kwargs)

Display all open figures.

Parameters
----------
block : bool, optional
    Whether to wait for all figures to be closed before returning.
```



```
optimal_clusters = 8
kmeans = KMeans(n_clusters=optimal_clusters, random_state=0)
kmeans.fit(data_Scaled_df)

dataset['Cluster'] = kmeans.labels_
print(dataset.head())
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 1
  super()._check_params_vs_input(X, default_n_init=10)
                  Date                  Mode        Category  \
0  20/09/2018 12:04:08                  Cash  Transportation   
1  20/09/2018 12:03:15                  Cash            Food   
2           19/09/2018  Saving Bank account 1    subscription   
3  17/09/2018 23:41:17  Saving Bank account 1    subscription   
4  16/09/2018 17:15:08                  Cash       Festivals   

               Subcategory                        Note  Amount  \
0                    Train         2 Place 5 to Place 0    30.0   
1                   snacks  Idli medu Vada mix 2 plates    60.0   
2                  Netflix         1 month subscription   199.0   
3  Mobile Service Provider            Data booster pack    19.0   
4             Ganesh Pujan                  Ganesh idol   251.0   
```

```
  Income/Expense Currency  Cluster
0        Expense      INR        1
1        Expense      INR        1
2        Expense      INR        1
3        Expense      INR        6
4        Expense      INR        1
```

```
for cluster in range(optimal_clusters):
    print(f"Cluster {cluster}")
    print(dataset[dataset['Cluster'] == cluster].head())
```