importing libraries

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

In [3]:
```
df = pd.read_csv("expense_data_1.csv")
```

start

In [5]:
```
df.head()
```

Out[5]:

	Date	Account	Category	Subcategory	Note	INR	Income/Expense	Note.1	Amount	Currency	Account.1
0	3/2/2022 10:11	CUB - online payment	Food	NaN	Brownie	50.0	Expense	NaN	50.0	INR	50.0
1	3/2/2022 10:11	CUB - online payment	Other	NaN	To lended people	300.0	Expense	NaN	300.0	INR	300.0
2	3/1/2022 19:50	CUB - online payment	Food	NaN	Dinner	78.0	Expense	NaN	78.0	INR	78.0
3	3/1/2022 18:56	CUB - online payment	Transportation	NaN	Metro	30.0	Expense	NaN	30.0	INR	30.0
4	3/1/2022 18:22	CUB - online payment	Food	NaN	Snacks	67.0	Expense	NaN	67.0	INR	67.0

In [6]:
```
df.drop_duplicates()
```

Out[6]:

	Date	Account	Category	Subcategory	Note	INR	Income/Expense	Note.1	Amount	Currency	Account.
0	3/2/2022 10:11	CUB - online payment	Food	NaN	Brownie	50.0	Expense	NaN	50.0	INR	50.
1	3/2/2022 10:11	CUB - online payment	Other	NaN	To lended people	300.0	Expense	NaN	300.0	INR	300.
2	3/1/2022 19:50	CUB - online payment	Food	NaN	Dinner	78.0	Expense	NaN	78.0	INR	78.
3	3/1/2022 18:56	CUB - online payment	Transportation	NaN	Metro	30.0	Expense	NaN	30.0	INR	30.
4	3/1/2022 18:22	CUB - online payment	Food	NaN	Snacks	67.0	Expense	NaN	67.0	INR	67.
...	...	...	...	...	...	...	...	...	...	...	
272	11/22/2021 14:16	CUB - online payment	Food	NaN	Dinner	90.0	Expense	NaN	90.0	INR	90.
273	11/22/2021 14:16	CUB - online payment	Food	NaN	Lunch with company	97.0	Expense	NaN	97.0	INR	97.
274	11/21/2021 17:07	CUB - online payment	Transportation	NaN	Rapido	130.0	Expense	NaN	130.0	INR	130.
275	11/21/2021 15:50	CUB - online payment	Food	NaN	Lunch	875.0	Expense	NaN	875.0	INR	875.
276	11/21/2021 13:30	CUB - online payment	Other	NaN	Got from gobi	2000.0	Income	NaN	2000.0	INR	2000.

277 rows × 11 columns

In [7]:
```
df.isna().sum()
```

Out[7]:
```
Date               0
Account            0
Category           0
Subcategory      277
Note               4
INR                0
Income/Expense     0
Note.1           277
Amount             0
Currency           0
Account.1          0
dtype: int64
```

In [8]:
```
df.drop(["Subcategory", "Note.1"], axis=1,inplace=True)
```

In [9]:
```
df.head()
```

Out[9]:

	Date	Account	Category	Note	INR	Income/Expense	Amount	Currency	Account.1
0	3/2/2022 10:11	CUB - online payment	Food	Brownie	50.0	Expense	50.0	INR	50.0
1	3/2/2022 10:11	CUB - online payment	Other	To lended people	300.0	Expense	300.0	INR	300.0
2	3/1/2022 19:50	CUB - online payment	Food	Dinner	78.0	Expense	78.0	INR	78.0
3	3/1/2022 18:56	CUB - online payment	Transportation	Metro	30.0	Expense	30.0	INR	30.0
4	3/1/2022 18:22	CUB - online payment	Food	Snacks	67.0	Expense	67.0	INR	67.0

In [10]:
```
df.describe()
```

Out[10]:

	INR	Amount	Account.1
count	277.000000	277.000000	277.000000
mean	410.750903	406.759134	406.759134
std	1065.756569	1065.158318	1065.158318
min	3.000000	3.000000	3.000000
25%	50.000000	50.000000	50.000000
50%	128.000000	125.000000	125.000000
75%	301.150000	300.000000	300.000000
max	10000.000000	10000.000000	10000.000000

In [11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 277 entries, 0 to 276
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Date            277 non-null    object 
 1   Account         277 non-null    object 
 2   Category        277 non-null    object 
 3   Note            273 non-null    object 
 4   INR             277 non-null    float64
 5   Income/Expense  277 non-null    object 
 6   Amount          277 non-null    float64
 7   Currency        277 non-null    object 
 8   Account.1       277 non-null    float64
dtypes: float64(3), object(6)
memory usage: 19.6+ KB
```

In [12]:
```
df['time']= df['Date'].str.split(" ").str[1]
df['Date']= df['Date'].str.split(" ").str[0]
```

In [13]:
```
df.head()
```

Out[13]:

	Date	Account	Category	Note	INR	Income/Expense	Amount	Currency	Account.1	time
0	3/2/2022	CUB - online payment	Food	Brownie	50.0	Expense	50.0	INR	50.0	10:11
1	3/2/2022	CUB - online payment	Other	To lended people	300.0	Expense	300.0	INR	300.0	10:11
2	3/1/2022	CUB - online payment	Food	Dinner	78.0	Expense	78.0	INR	78.0	19:50
3	3/1/2022	CUB - online payment	Transportation	Metro	30.0	Expense	30.0	INR	30.0	18:56
4	3/1/2022	CUB - online payment	Food	Snacks	67.0	Expense	67.0	INR	67.0	18:22

In [14]:
```
df['Currency'].unique()
```

Out[14]:
```
array(['INR', 'USD'], dtype=object)
```

In [15]:
```
df['Amount'] = df.apply(lambda row: row['Amount'] * 93 if row['Currency'] == 'USD' else row['Amount'], axis=1)
```

In [16]:
```
df.drop(['Currency', 'Account.1'], axis=1, inplace=True)
```

In [17]:
```
df.head()
```

Out[17]:

	Date	Account	Category	Note	INR	Income/Expense	Amount	time
0	3/2/2022	CUB - online payment	Food	Brownie	50.0	Expense	50.0	10:11
1	3/2/2022	CUB - online payment	Other	To lended people	300.0	Expense	300.0	10:11
2	3/1/2022	CUB - online payment	Food	Dinner	78.0	Expense	78.0	19:50
3	3/1/2022	CUB - online payment	Transportation	Metro	30.0	Expense	30.0	18:56
4	3/1/2022	CUB - online payment	Food	Snacks	67.0	Expense	67.0	18:22

In [18]:
```
df['Date'] = pd.to_datetime(df['Date'])
```

In [19]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 277 entries, 0 to 276
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype         
---  ------          --------------  -----         
 0   Date            277 non-null    datetime64[ns]
 1   Account         277 non-null    object        
 2   Category        277 non-null    object        
 3   Note            273 non-null    object        
 4   INR             277 non-null    float64       
 5   Income/Expense  277 non-null    object        
 6   Amount          277 non-null    float64       
 7   time            277 non-null    object        
dtypes: datetime64[ns](1), float64(2), object(5)
memory usage: 17.4+ KB
```

In [20]:
```
df.columns
```

Out[20]:
```
Index(['Date', 'Account', 'Category', 'Note', 'INR', 'Income/Expense',
       'Amount', 'time'],
      dtype='object')
```

In [21]:
```
df['Category'].unique()
```

Out[21]:
```
array(['Food', 'Other', 'Transportation', 'Social Life', 'Household',
       'Apparel', 'Education', 'Salary', 'Allowance', 'Self-development',
       'Beauty', 'Gift', 'Petty cash'], dtype=object)
```

In [22]:
```
df['Account'].unique()
```

Out[22]:
```
array(['CUB - online payment', 'Cash'], dtype=object)
```

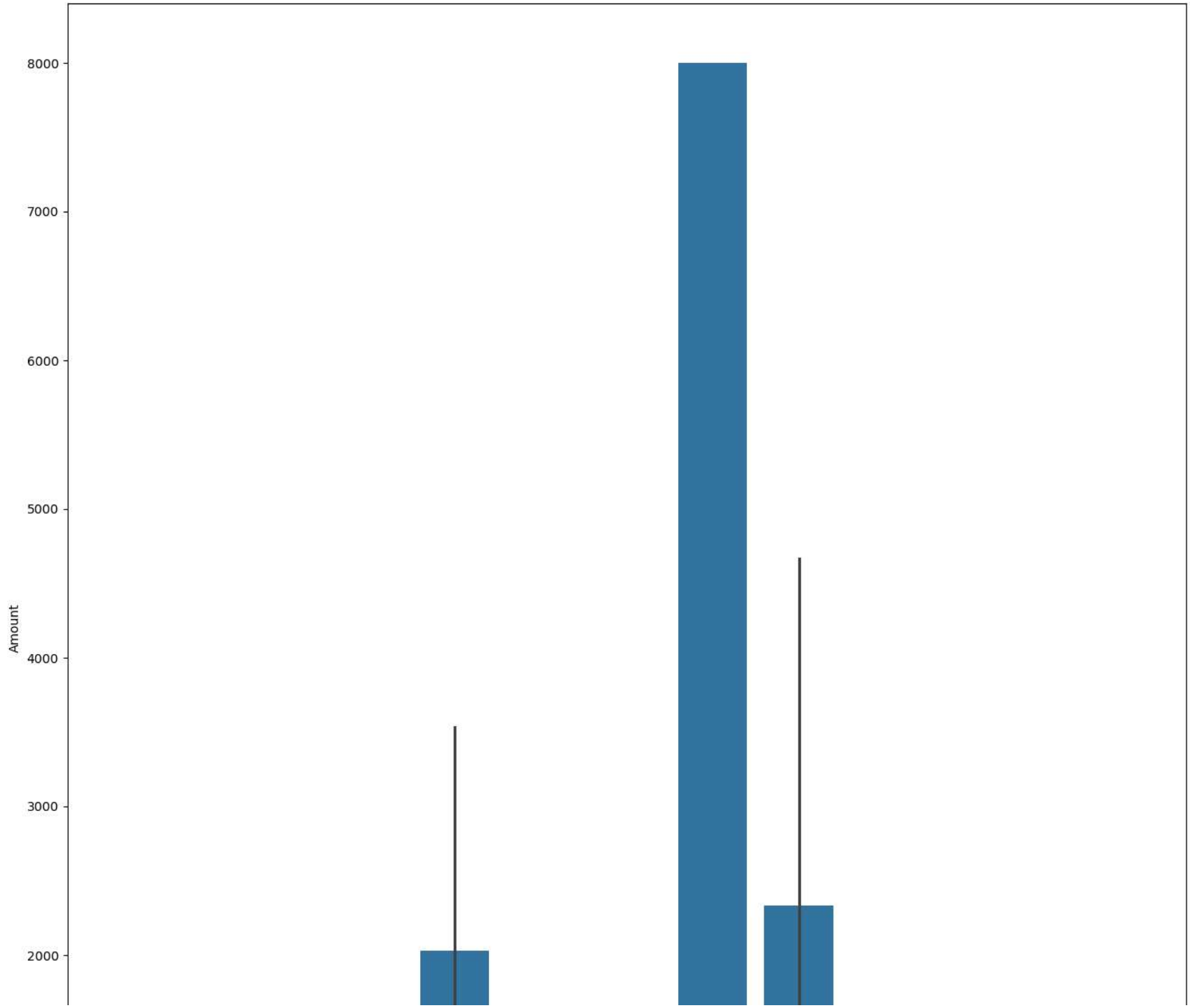
In [23]:
```
df['Account'] = df.apply(lambda row: 'online' if row['Account'] == 'CUB - online payment' else row['Account'], axis=
```

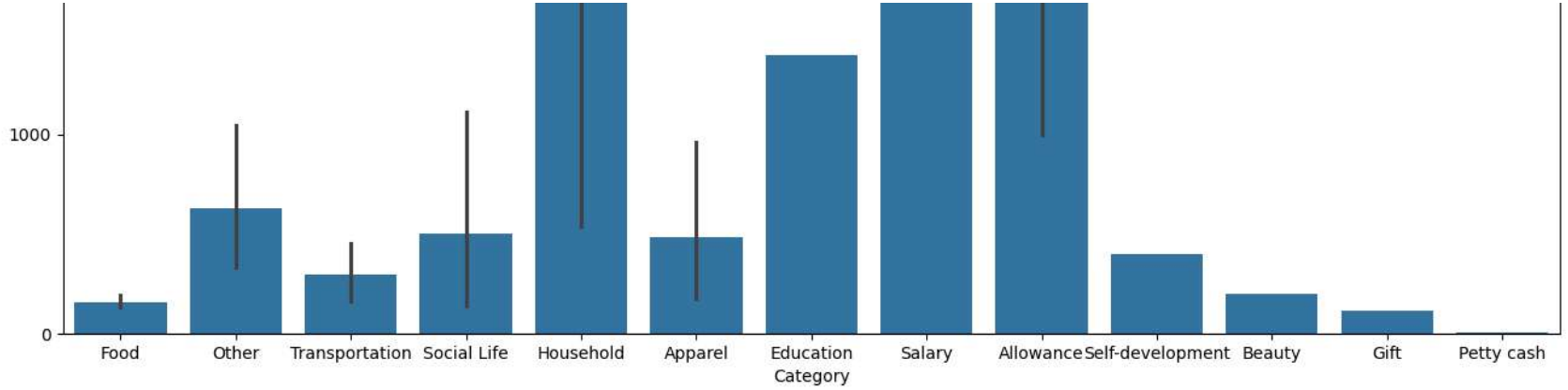
In [24]:
```
df['Account'].unique()
```

Out[24]:
```
array(['online', 'Cash'], dtype=object)
```

In [25]:
```
plt.figure(figsize=(16,18))
sns.barplot(data=df, x='Category', y= 'Amount')
```

Out[25]:
```
<Axes: xlabel='Category', ylabel='Amount'>
```

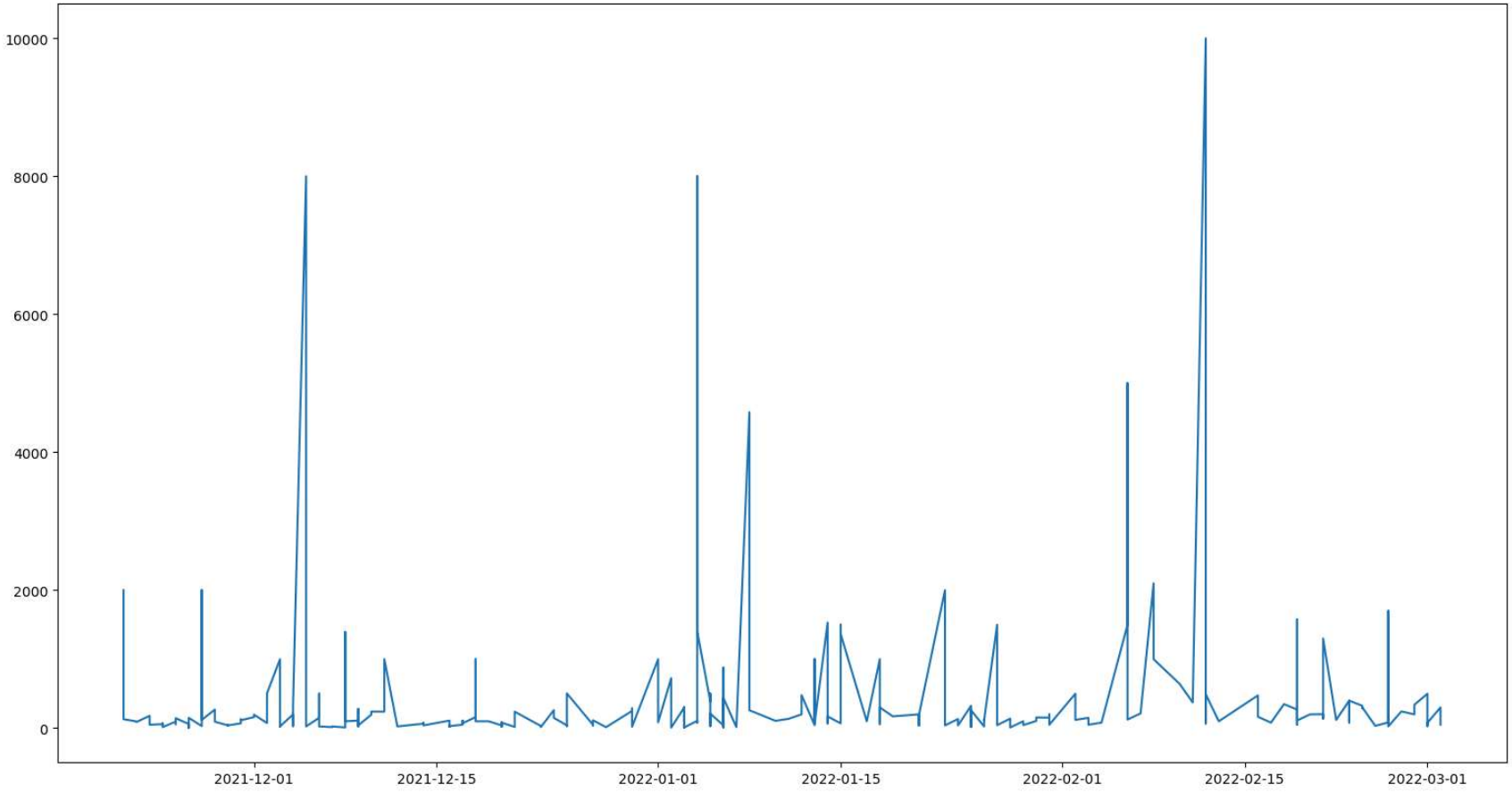

8000
7000
6000
5000
4000
3000
2000
Amount



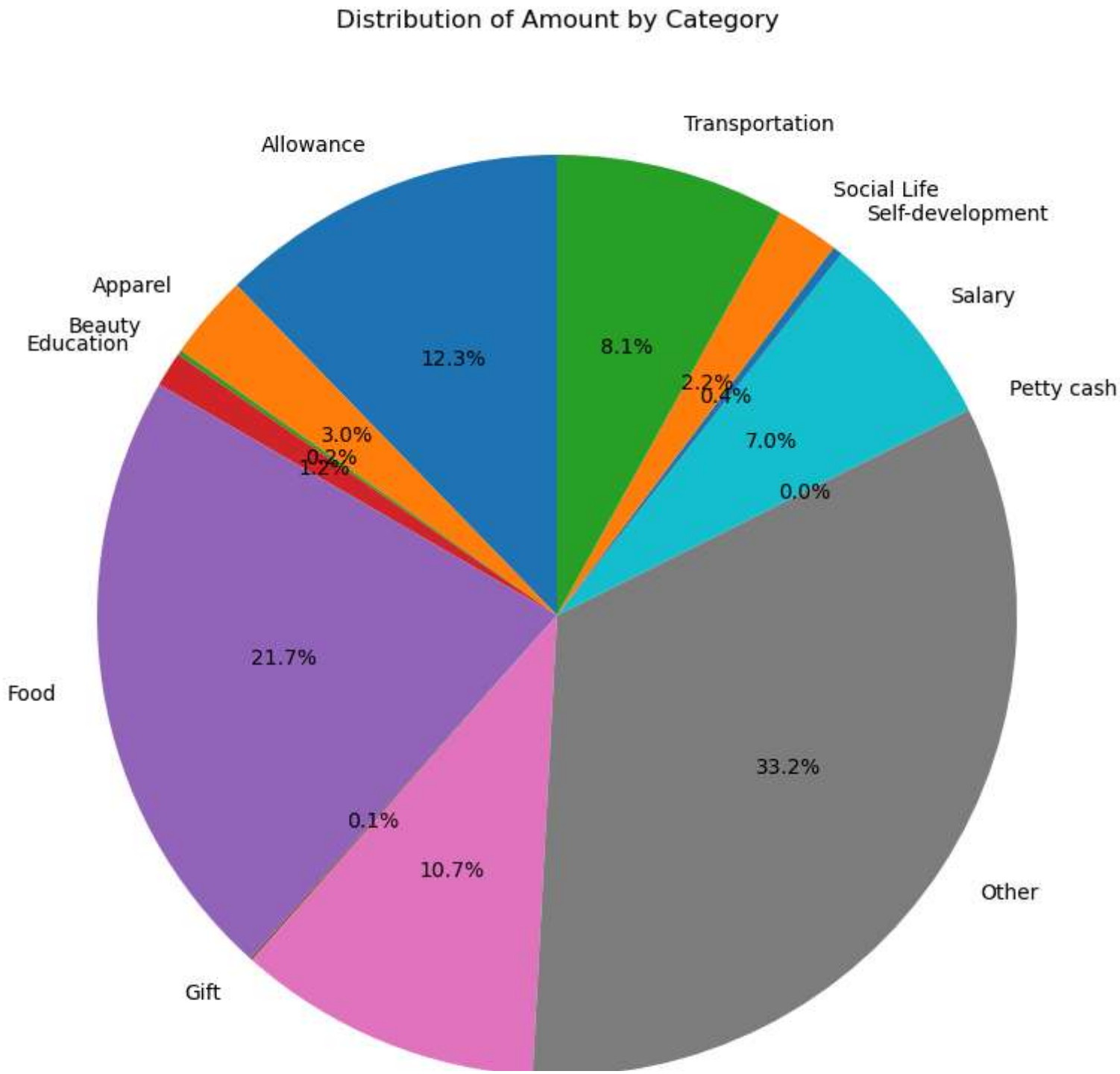
In [26]:
```
df.set_index('Date', inplace =True)
```

In [27]:
```
plt.figure(figsize=(19,10))
plt.plot(df.index, df['Amount'])
```

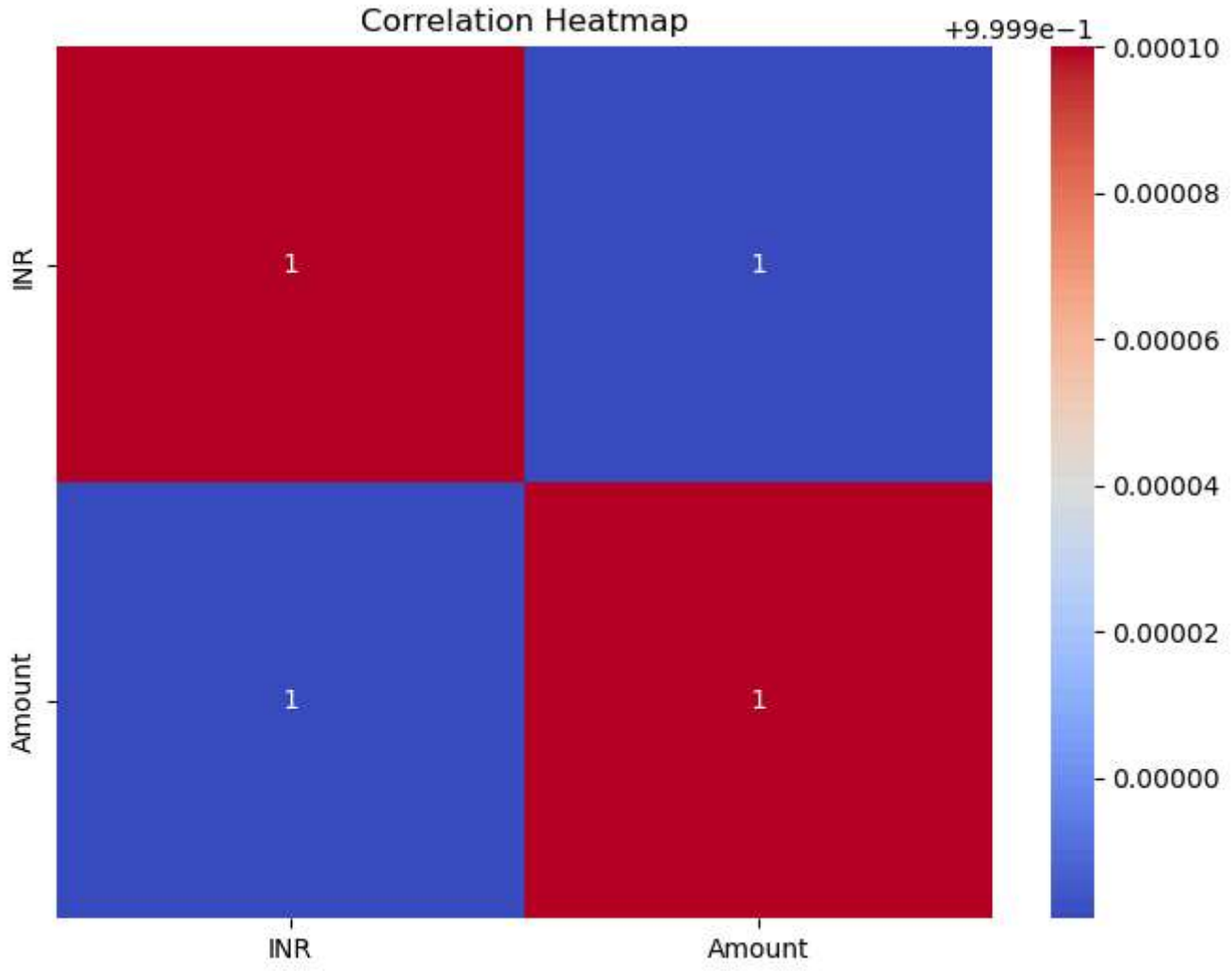
Out[27]:
```
[<matplotlib.lines.Line2D at 0x14131881d10>]
```



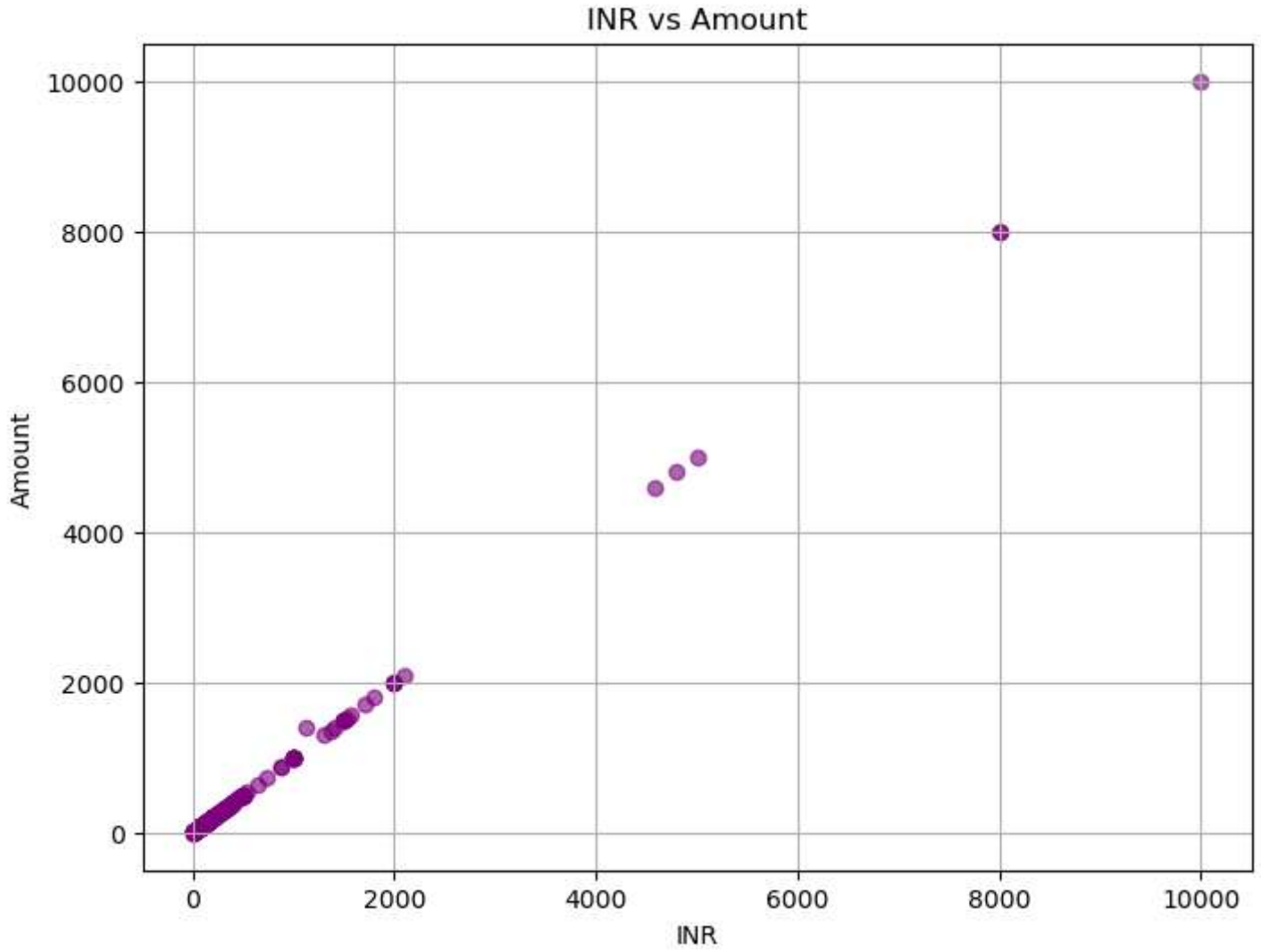
In [28]:
```
# Group by 'Category' and sum the 'Amount'
category_amount = df.groupby('Category')['Amount'].sum().reset_index()

plt.figure(figsize=(10, 10))
plt.pie(category_amount['Amount'], labels=category_amount['Category'], autopct='%1.1f%%', startangle=90)
plt.title('Distribution of Amount by Category')
plt.show()
```


Distribution of Amount by Category
Transportation
Allowance
Social Life
Self-development
Apparel
Salary
Beauty
Education
Petty cash
12.3%
8.1%
2.2%
0.4%
3.0%
7.0%
0.2%
1.2%
0.0%
21.7%
Food
33.2%
0.1%
10.7%
Other
Gift

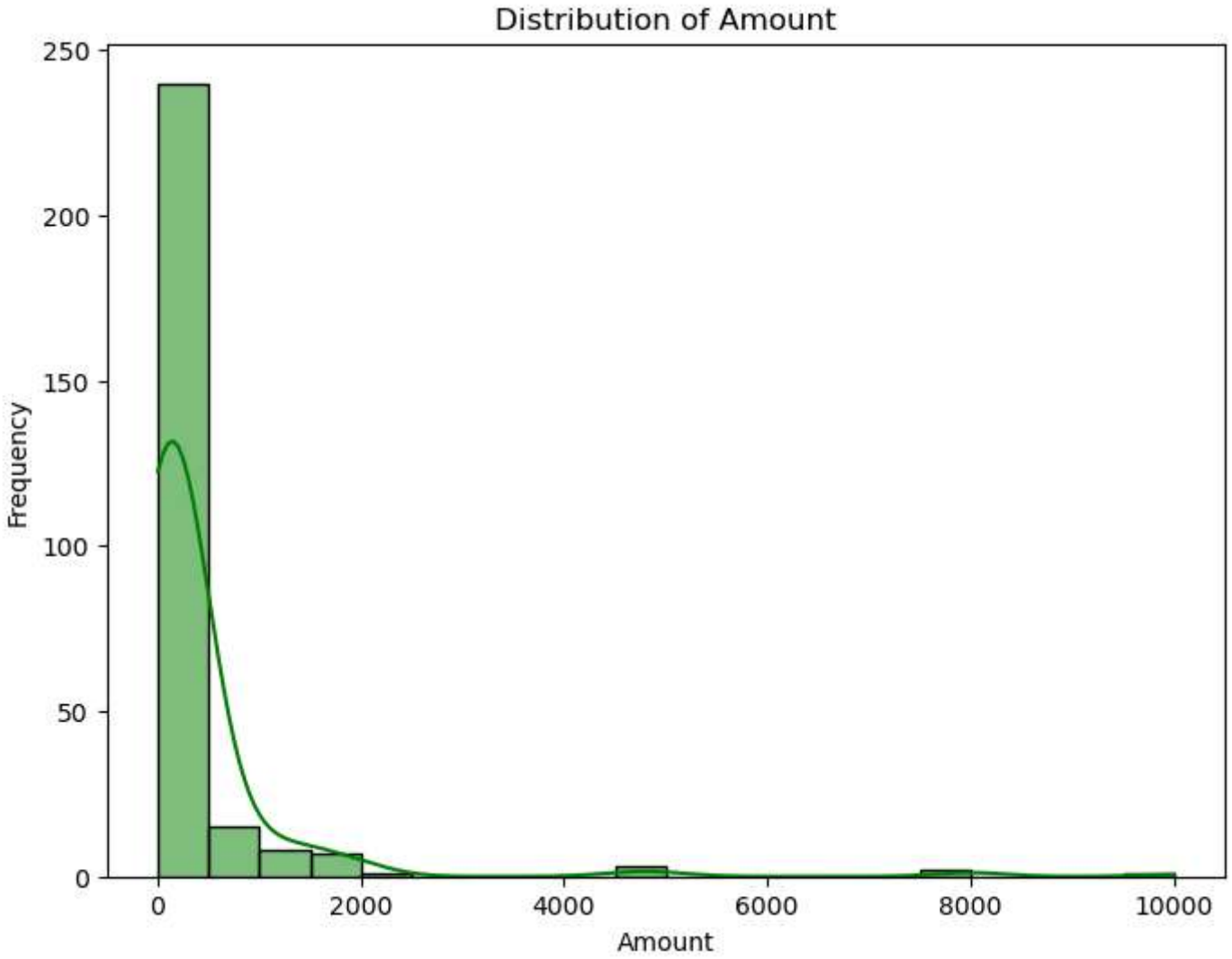
In [29]:
```
plt.figure(figsize=(8, 6))
sns.heatmap(df[['INR', 'Amount']].corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



In [30]:
```
plt.figure(figsize=(8, 6))
plt.scatter(df['INR'], df['Amount'], alpha=0.6, color='purple')
plt.title('INR vs Amount')
plt.xlabel('INR')
plt.ylabel('Amount')
plt.grid(True)
plt.show()
```



In [31]:
```
plt.figure(figsize=(8, 6))
sns.histplot(df['Amount'], bins=20, kde=True, color='green')
plt.title('Distribution of Amount')
plt.xlabel('Amount')
plt.ylabel('Frequency')
plt.show()
```



In [66]:
```
df = pd.read_csv("expense_data_1.csv")
```

In [ ]:

In [ ]:

In [ ]:

Experiment 3

In [68]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [76]:
```
df['Date'] = pd.to_datetime(df['Date'], errors='coerce')


df = df.dropna(subset=['Date'])
```

In [78]:
```
df_grouped = df.groupby('Date')['Amount'].sum().reset_index()


df_grouped['Days'] = (df_grouped['Date'] - df_grouped['Date'].min()).dt.days

# Prepare training data
X = df_grouped[['Days']]
y = df_grouped['Amount']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, shuffle=False)

model = LinearRegression()
model.fit(X_train, y_train)
```

Out[78]:

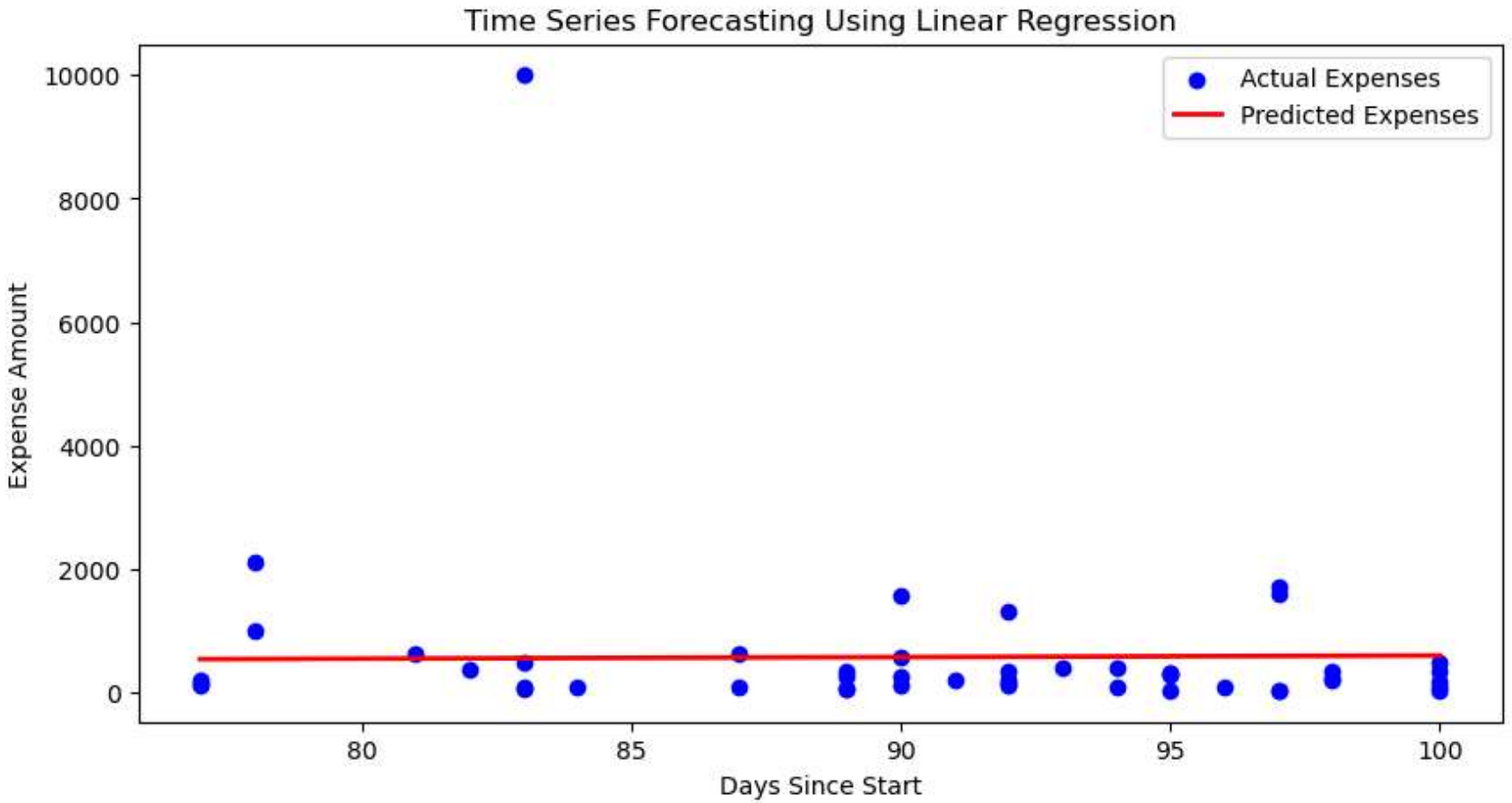
▾ LinearRegression

```
LinearRegression()
```

In [80]:
```
y_pred = model.predict(X_test)
```

In [82]:
```
plt.figure(figsize=(10, 5))
plt.scatter(X_test, y_test, color='blue', label='Actual Expenses')
plt.plot(X_test, y_pred, color='red', label='Predicted Expenses', linewidth=2)
plt.xlabel("Days Since Start")
plt.ylabel("Expense Amount")
plt.title("Time Series Forecasting Using Linear Regression")
```

```
plt.legend()
plt.show()
```



In [84]:
```
days_future = np.array([[df_grouped['Days'].max() + i] for i in range(1, 31)])  # Predict next 30 days
future_predictions = model.predict(days_future)

future_dates = pd.date_range(start=df_grouped['Date'].max(), periods=30, freq='D')
future_df = pd.DataFrame({'Date': future_dates, 'Predicted_Expense': future_predictions})
print(future_df)
```

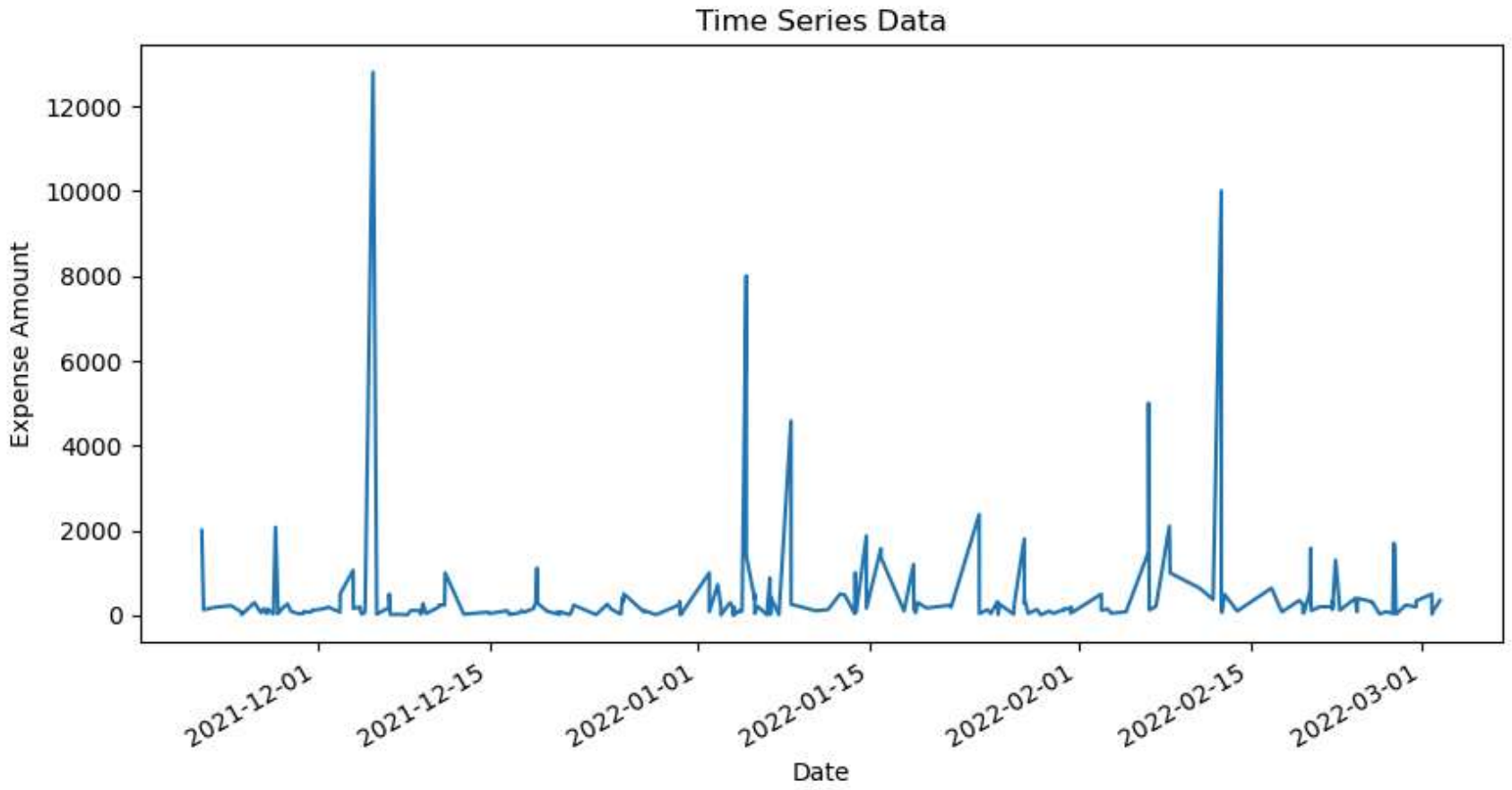
	Date	Predicted_Expense
0	2022-03-02 10:11:00	602.515742
1	2022-03-03 10:11:00	605.065485
2	2022-03-04 10:11:00	607.615228
3	2022-03-05 10:11:00	610.164971
4	2022-03-06 10:11:00	612.714714
5	2022-03-07 10:11:00	615.264457
6	2022-03-08 10:11:00	617.814199
7	2022-03-09 10:11:00	620.363942
8	2022-03-10 10:11:00	622.913685
9	2022-03-11 10:11:00	625.463428
10	2022-03-12 10:11:00	628.013171
11	2022-03-13 10:11:00	630.562913
12	2022-03-14 10:11:00	633.112656
13	2022-03-15 10:11:00	635.662399
14	2022-03-16 10:11:00	638.212142
15	2022-03-17 10:11:00	640.761885
16	2022-03-18 10:11:00	643.311628
17	2022-03-19 10:11:00	645.861370
18	2022-03-20 10:11:00	648.411113
19	2022-03-21 10:11:00	650.960856
20	2022-03-22 10:11:00	653.510599
21	2022-03-23 10:11:00	656.060342
22	2022-03-24 10:11:00	658.610085
23	2022-03-25 10:11:00	661.159827
24	2022-03-26 10:11:00	663.709570
25	2022-03-27 10:11:00	666.259313
26	2022-03-28 10:11:00	668.809056
27	2022-03-29 10:11:00	671.358799
28	2022-03-30 10:11:00	673.908542
29	2022-03-31 10:11:00	676.458284

```
C:\Users\22150\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but
LinearRegression was fitted with feature names
  warnings.warn(
```

EXP 4

In [88]:
```
df_grouped.set_index('Date')['Amount'].plot(figsize=(10, 5), title="Time Series Data")
plt.xlabel("Date")
```

```
plt.ylabel("Expense Amount")
plt.show()
```



In [90]:
```
from statsmodels.tsa.stattools import adfuller
```

In [92]:
```
adf_test = adfuller(df_grouped['Amount'])
print("ADF Statistic:", adf_test[0])
print("p-value:", adf_test[1])
print("Critical Values:", adf_test[4])
```

```
ADF Statistic: -14.213509911021116
p-value: 1.705391768844632e-26
Critical Values: {'1%': -3.458010773719797, '5%': -2.8737103617125186, '10%': -2.5732559963936206}
```

In [94]:
```
if adf_test[1] < 0.05:
    print("The time series data is stationary.")
else:
    print("The time series data is non-stationary. Consider differencing or transformation.")
```

```
The time series data is stationary.
```

In [ ]:
```

```