```
from google.colab import drive
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.arima.model import ARIMA

drive.mount("drive")
```

```
Drive already mounted at drive; to attempt to forcibly remount, call
drive.mount("drive", force_remount=True).
```

```
df = pd.read_csv("/content/drive/MyDrive/Course Work/Sem 4/Data
Analysis and Visualization/Lab 13/myExpenses1.csv")

df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 145,\n  \"fields\": [\
n    {\n      \"column\": \"Date\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 31,\n
\"samples\": [\n          \"28/3/2023\",\n          \"16/3/2023\",\n
\"24/3/2023\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Item\",\n      \"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 23,\n        \"samples\": [\n
\"biryani\",\n          \"choclate\",\n          \"chai\"\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Amount\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 56,\n
\"min\": 5,\n        \"max\": 500,\n        \"num_unique_values\":
22,\n        \"samples\": [\n          7,\n          45,\n
30\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Category\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"friend\",\n          \"alone\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Time\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 30,\n
\"samples\": [\n          \"18:30\",\n          \"9:30\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"day\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 7,\n
\"samples\": [\n          \"Wednesday\",\n          \"Thursday\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```

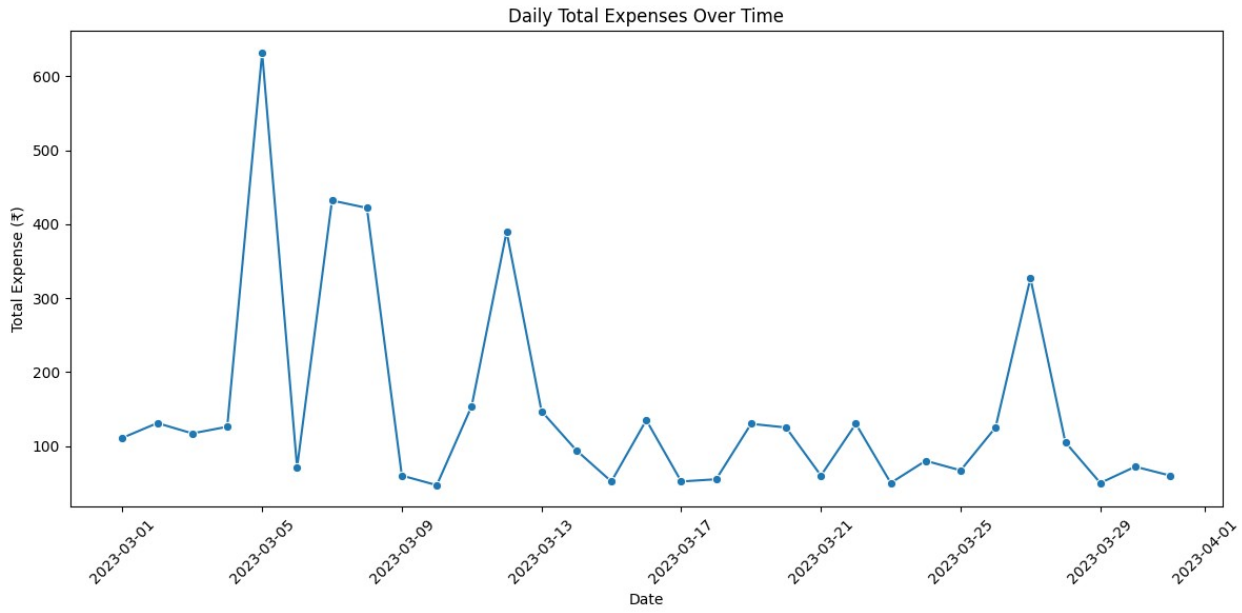

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145 entries, 0 to 144
```

```
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Date      145 non-null    object
 1   Item      145 non-null    object
 2   Amount    145 non-null    int64
 3   Category  144 non-null    object
 4   Time      145 non-null    object
 5   day       145 non-null    object
dtypes: int64(1), object(5)
memory usage: 6.9+ KB
```

1. Create a time series plot of daily total expenses. What patterns can you observe?

```
df['Date'] = pd.to_datetime(df['Date'], dayfirst=True)

daily_expense = df.groupby('Date')['Amount'].sum().reset_index()

plt.figure(figsize=(12, 6))
sns.lineplot(data=daily_expense, x='Date', y='Amount', marker='o')
plt.title('Daily Total Expenses Over Time')
plt.xlabel('Date')
plt.ylabel('Total Expense (₹)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



```
'''
Observations and trends from the plot:-
1. Some dates like 2023-03-05 show relatively higher expenses than the
others and the increase in expense is very random and abrupt
2. The expenses are sometimes decreasing between two dates and
sometimes increasing but the total plot is non-monotonic (sometimes
decreasing and sometimes increasing)
3. Seasonlity can also be observed in the dataset seen in the plot
4. There are no dates on which the spending is 0 and everyday has some
kind of spending done
'''
```

```
{"type":"string"}
```

2. Which week had the highest total spending?

```
df['week'] = df['Date'].dt.isocalendar().week

weekly_spending = df.groupby('week')['Amount'].sum()

max_week = weekly_spending.idxmax()

max_amount = weekly_spending.max()

print(f"Week {max_week} had the highest spending: ₹{max_amount}")
```

```
Week 10 had the highest spending: ₹1576
```

3.On which day of the week is spending highest on average?

```
avg_spending_by_day = df.groupby('day')
['Amount'].mean().sort_values(ascending=False)

print(avg_spending_by_day)
```

```
day
Sunday       63.850000
Tuesday      36.368421
Monday       31.904762
Wednesday    31.304348
Thursday     22.409091
Saturday     20.100000
Friday       17.800000
Name: Amount, dtype: float64
```

4.Which item is purchased most frequently over time?

```
most_frequent_item = df['Item'].value_counts().idxmax()
```

```
count = df['Item'].value_counts().max()

print(f"The most frequently purchased item is '{most_frequent_item}'
with {count} purchases.")
```

```
The most frequently purchased item is 'chai with snaks' with 48
purchases.
```

5.What was the total amount spent each day in March 2023 and also find the moving Average? (use ARIMA)

```
march_df = df[(df['Date'].dt.month == 3) & (df['Date'].dt.year ==
2023)]

march_daily = march_df.groupby('Date')['Amount'].sum()

model = ARIMA(march_daily, order=(0, 1, 1))
```

```
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/
tsa_model.py:473: ValueWarning: No frequency information was provided,
so inferred frequency D will be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model
.py:473: ValueWarning: No frequency information was provided, so
inferred frequency D will be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model
.py:473: ValueWarning: No frequency information was provided, so
inferred frequency D will be used.
  self._init_dates(dates, freq)
```

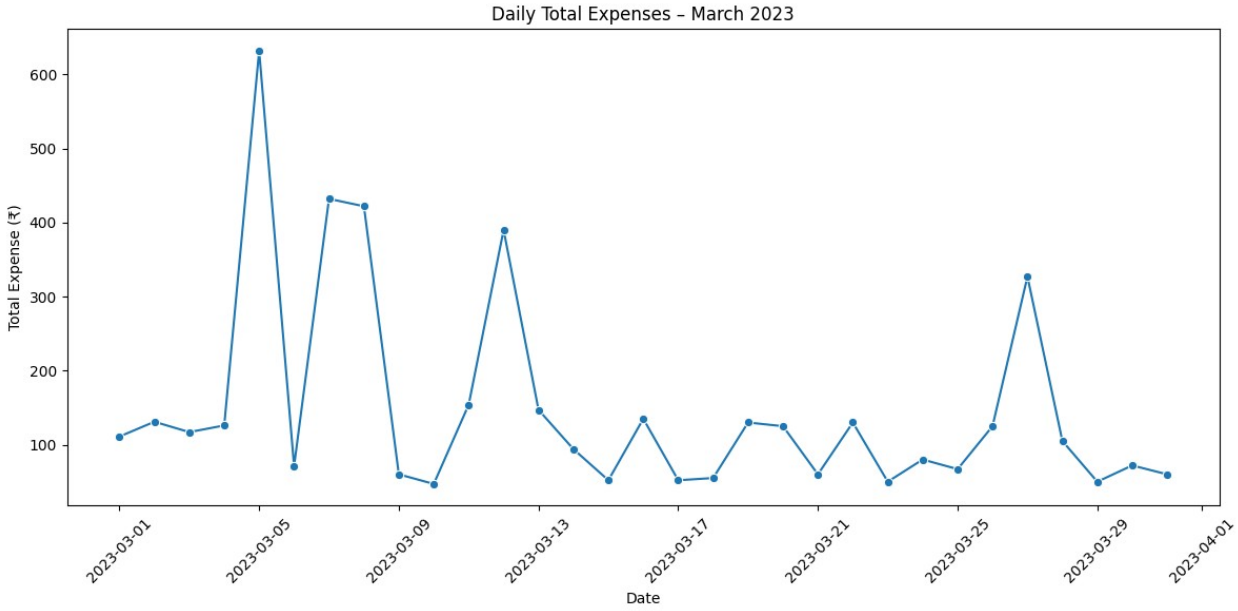
```
results = model.fit()

march_daily_ma = results.fittedvalues

march_analysis = pd.DataFrame({'Daily Total': march_daily, 'ARIMA MA':
march_daily_ma})

print(march_analysis)
```

```
            Daily Total    ARIMA MA
Date
2023-03-01          111    0.000000
2023-03-02          131  109.125362
2023-03-03          117  120.091377
2023-03-04          126  119.029179
2023-03-05          632  120.895976
2023-03-06           71  235.780369
2023-03-07          432  203.149203
2023-03-08          422  244.444746
2023-03-09           60  274.356840
```

```
2023-03-10           47  240.038576
2023-03-11          154  210.280877
2023-03-12          390  201.845523
2023-03-13          147  229.462657
2023-03-14           94  217.545491
2023-03-15           52  199.896224
2023-03-16          135  178.948874
2023-03-17           52  172.763733
2023-03-18           55  155.848415
2023-03-19          130  141.772277
2023-03-20          125  140.133429
2023-03-21           60  138.030756
2023-03-22          130  127.204626
2023-03-23           50  127.592046
2023-03-24           80  116.846884
2023-03-25           67  111.747243
2023-03-26          125  105.556907
2023-03-27          327  108.245788
2023-03-28          105  138.491045
2023-03-29           50  133.861361
2023-03-30           72  122.270213
2023-03-31           60  115.322656
```

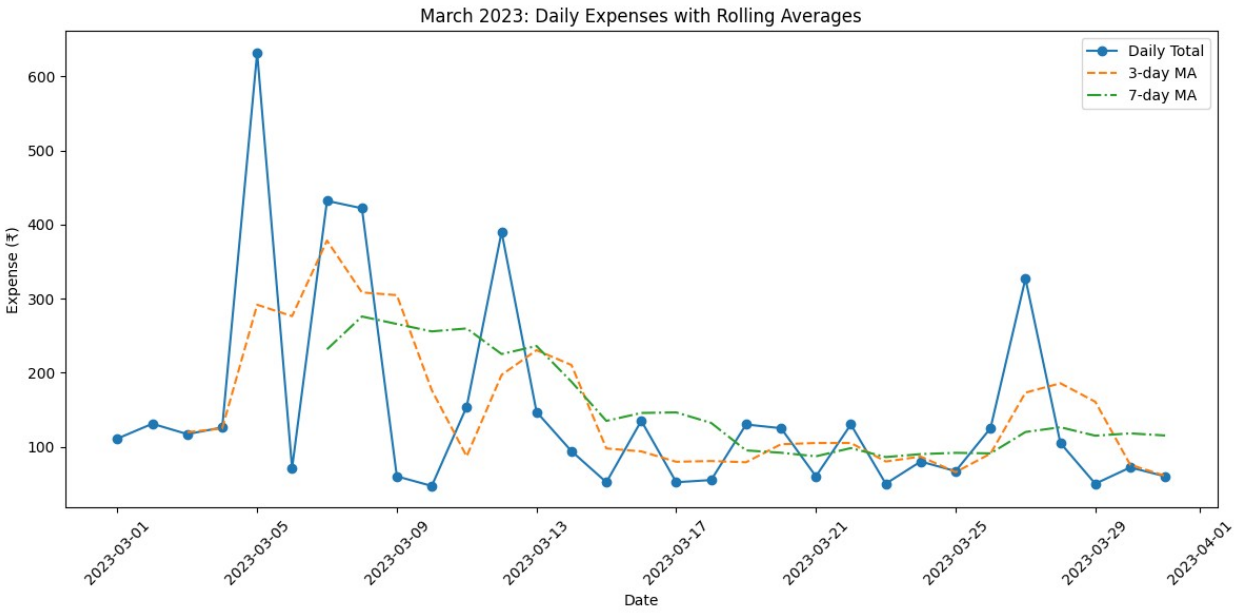
6.Create a time series plot of total daily expenses for March 2023.

```
plt.figure(figsize=(12, 6))
sns.lineplot(data=march_analysis, x=march_analysis.index, y='Daily
Total', marker='o')
plt.title('Daily Total Expenses – March 2023')
plt.xlabel('Date')
plt.ylabel('Total Expense (₹)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



7.Plot the daily expenses with a 3-day and 7-day rolling average. What trend emerges?

```
march_analysis['3-day MA'] = march_analysis['Daily
Total'].rolling(window=3).mean()

march_analysis['7-day MA'] = march_analysis['Daily
Total'].rolling(window=7).mean()

plt.figure(figsize=(12, 6))
plt.plot(march_analysis.index, march_analysis['Daily Total'],
label='Daily Total', marker='o')
plt.plot(march_analysis.index, march_analysis['3-day MA'], label='3-
day MA', linestyle='--')
plt.plot(march_analysis.index, march_analysis['7-day MA'], label='7-
day MA', linestyle='-.')
plt.title('March 2023: Daily Expenses with Rolling Averages')
plt.xlabel('Date')
plt.ylabel('Expense (₹)')
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```




```
'''
1. 3-day MA smooths short-term spikes
2. 7-day MA shows broader trends
3. Useful for detecting overall rise or fall in spending
'''
```

```
{"type":"string"}
```

8.Check if the daily expense time series in March is stationary using the Augmented Dickey-Fuller (ADF) test. What is the p-value?

```
adf_result = adfuller(march_analysis['Daily Total'].dropna())

print(f'ADF Statistic: {adf_result[0]}')
```

```
ADF Statistic: -2.0648254683724594
```

```
print(f'p-value: {adf_result[1]}')
```

```
p-value: 0.25889156352524395
```

```
'The p-value is greater than 0.05 which means that the series is non-stationary'
```

```
{"type":"string"}
```