Daily Transactions Analysis Project

Internship Project – Unified Mentor (Data Analyst Role)

Objective

The goal of this project was to analyze and forecast household daily transactions using Python. The analysis focused on understanding spending patterns, identifying dominant categories, and applying a simple machine learning model to predict future expenses.

Process

Data Preparation:
- Loaded & Cleaned data by handling missing values removing duplicates, and converting Date into proper datetime format.
- Engineered features including Year, Month, Day, Weekday, and a numerical DayNumber for modeling.

Exploratory Data Analysis (EDA):
- Found that Expenses dominated (≈92%) compared to Income.
- Food, Transportation, and Household were the top expense categories.
- Most transactions were via Cash and Bank Account.
- Visualized Income vs Expense counts, Top 10 Categories, and Monthly Trends.

Trend Analysis:
- Built daily aggregated data with 7-day rolling averages to smooth expense patterns.
- Observed recurring spikes in food and subscription-related spending.

Machine Learning Forecasting:
- Achieved reasonable accuracy with MAE/RMSE evaluation.
- Generated a 30-day expense forecast, demonstrating how ML can anticipate upcoming financial trends.

Key Insights

- Household spending is heavily skewed towards food and transportation.
- Cash remains the primary transaction mode despite digital alternatives.
- Forecasting showed expense trends are predictable with simple ML models, providing useful financial planning insights.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

file = r'C:\Users\Ankita\Data Analyst\Internship\Daily_Transaction\
Daily_Data.csv'
df = pd.read_csv(file)
df.head()
```

```
                  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
0        Expense      INR
1        Expense      INR
2        Expense      INR
3        Expense      INR
4        Expense      INR
```

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

```
Date                0
Mode                0
Category            0
Subcategory       635
Note              521
Amount              0
Income/Expense      0
Currency            0
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2461 entries, 0 to 2460
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype
```

```
---  -------          --------------  ------
 0   Date             2461 non-null   object
 1   Mode             2461 non-null   object
 2   Category         2461 non-null   object
 3   Subcategory      1826 non-null   object
 4   Note             1940 non-null   object
 5   Amount           2461 non-null   float64
 6   Income/Expense   2461 non-null   object
 7   Currency         2461 non-null   object
dtypes: float64(1), object(7)
memory usage: 153.9+ KB
```

```
df.describe()
```

```
              Amount
count    2461.000000
mean     2751.145380
std     12519.615804
min         2.000000
25%        35.000000
50%       100.000000
75%       799.000000
max    250000.000000
```

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

```
df = df.drop_duplicates()
df.duplicated().sum()
```

```
0
```

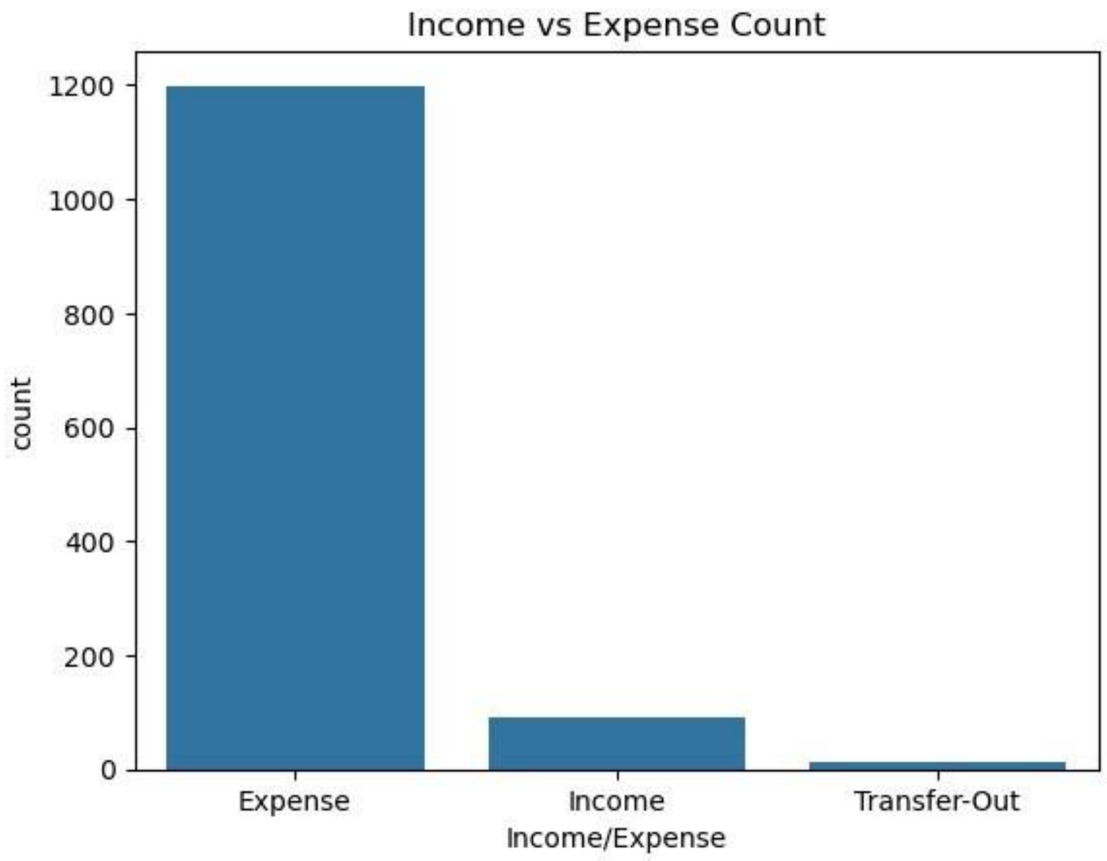
```
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
df['Day'] = df['Date'].dt.day
df['Weekday'] = df['Date'].dt.day_name()
```

```
df['Subcategory'] = df['Subcategory'].fillna("Unknown")
df['Note'] = df['Note'].fillna("No Note")
```

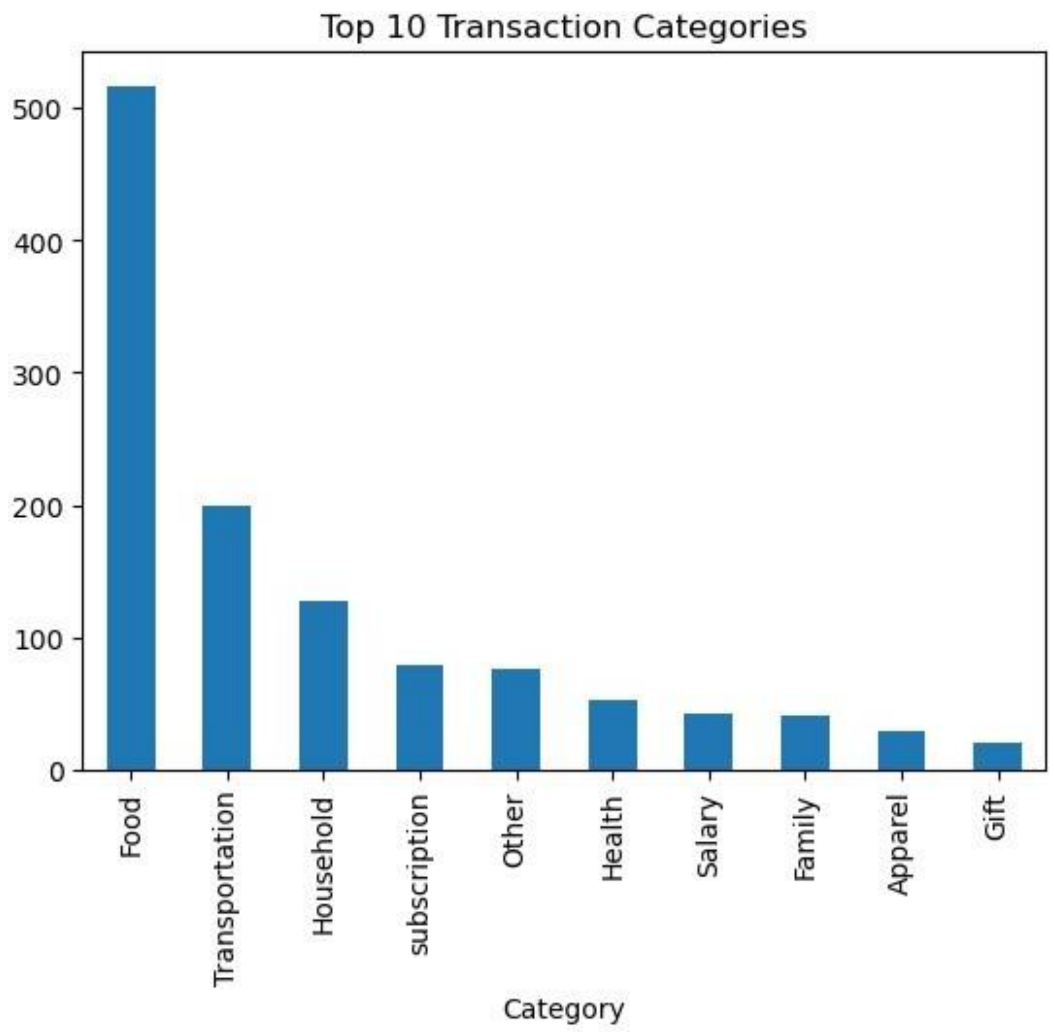
```
df.isnull().sum().sum()
```

```
0
```

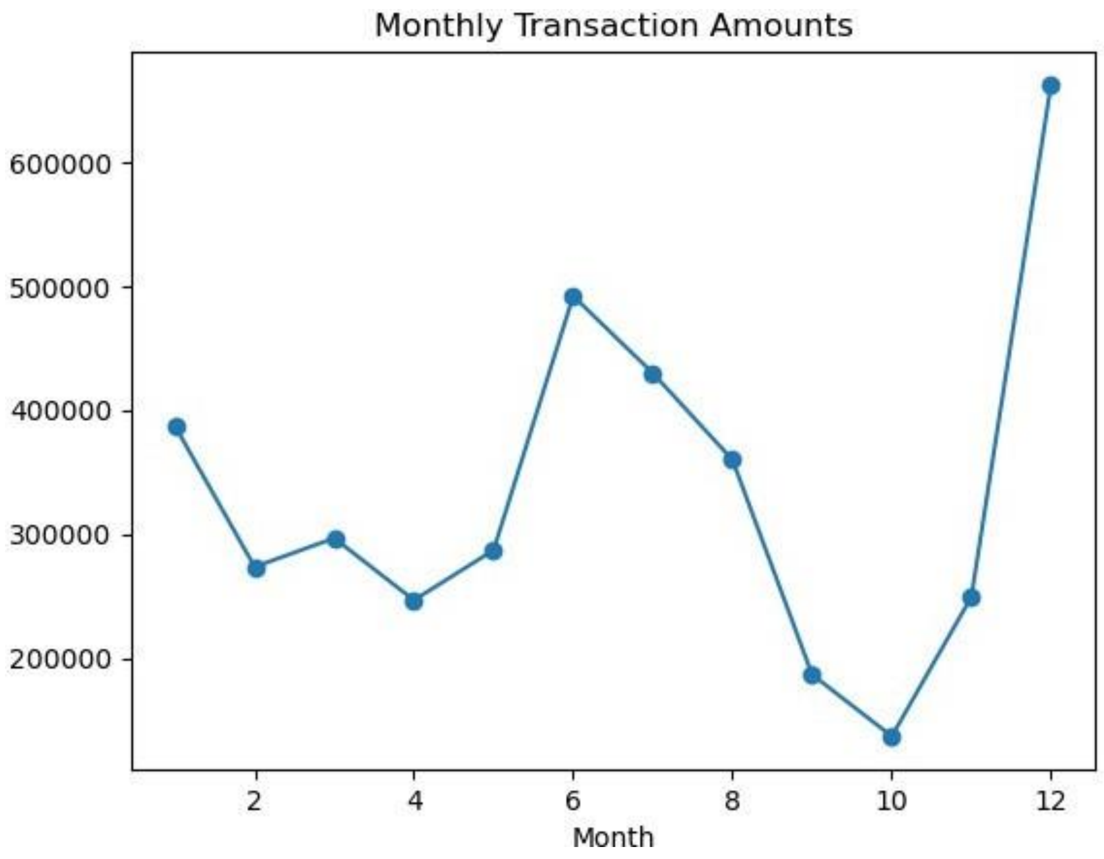
```
# Income vs Expense count
sns.countplot(data=df, x='Income/Expense')
plt.title("Income vs Expense Count")
plt.show()
```



```
# Top 10 categories by frequency
df['Category'].value_counts().head(10).plot(kind='bar')
plt.title("Top 10 Transaction Categories")
plt.show()
```

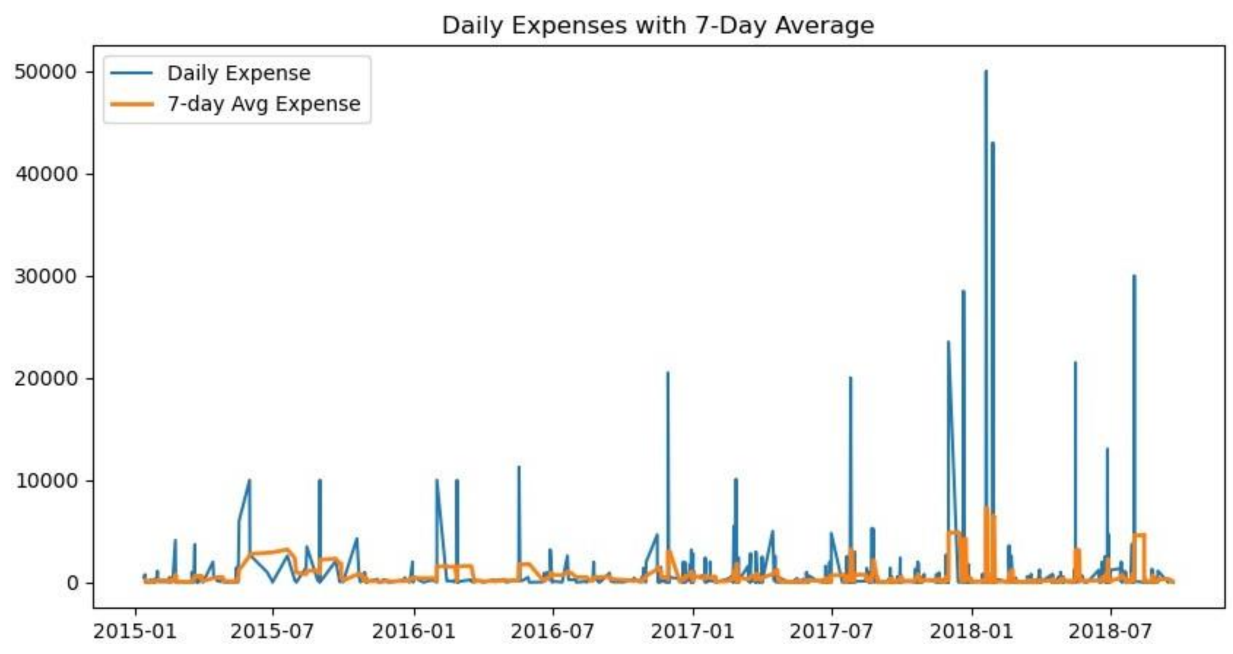


```
# Monthly spending trend
df.groupby('Month')['Amount'].sum().plot(kind='line', marker='o')
plt.title("Monthly Transaction Amounts")
plt.show()
```



```
daily_type = df.groupby(['Date', 'Income/Expense'])
['Amount'].sum().unstack().fillna(0).reset_index()
daily_type['Expense_7d_avg'] = daily_type['Expense'].rolling(7).mean()
daily_type['Income_7d_avg'] = daily_type['Income'].rolling(7).mean()

plt.figure(figsize=(10,5))
plt.plot(daily_type['Date'], daily_type['Expense'], label='Daily
Expense')
plt.plot(daily_type['Date'], daily_type['Expense_7d_avg'], label='7-
day Avg Expense', linewidth=2)
plt.legend()
plt.title("Daily Expenses with 7-Day Average")
plt.show()
```



```
df = df.sort_values('Date')
df['DayNumber'] = (df['Date'] - df['Date'].min()).dt.days

ml_df = df[df['Income/Expense'] == 'Expense'].copy()
X = ml_df[['DayNumber']]
y = ml_df['Amount']

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

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

```
LinearRegression()
```

```
y_pred = model.predict(X_test)
print("MAE:", mean_absolute_error(y_test, y_pred))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred)))
```

```
MAE: 849.6395687102219
RMSE: 2547.811376142389
```

```
last_day = df['DayNumber'].max()
future_days = list(range(last_day+1, last_day+31))

future_preds = model.predict(pd.DataFrame(future_days,
columns=['DayNumber']))

future_results = pd.DataFrame({
    'Date': pd.date_range(df['Date'].max(), periods=30, freq='D'),
    'Predicted_Expense': future_preds
})

print(future_results.head())
```

```
                 Date  Predicted_Expense
0 2018-09-20 12:04:08         700.034610
1 2018-09-21 12:04:08         700.255883
2 2018-09-22 12:04:08         700.477155
3 2018-09-23 12:04:08         700.698428
4 2018-09-24 12:04:08         700.919701
```