6.Implement program to apply moving average smoothing for data preparation and time series forecasting.

AIM:

To apply **Moving Average Smoothing** on time series data to reduce noise and forecast future expenses.

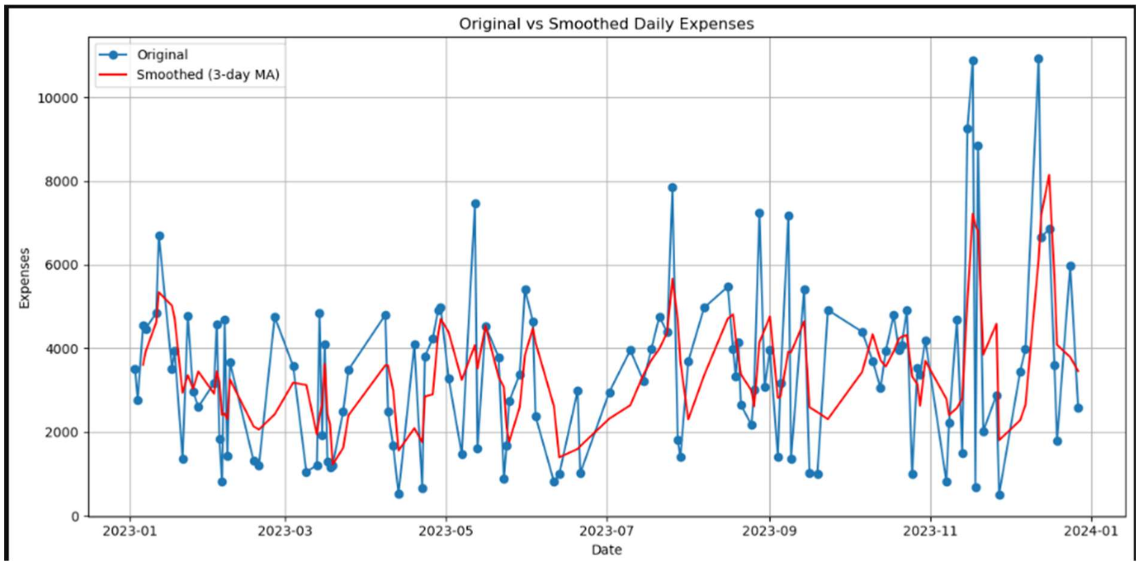
PROCEDURE:

1) **Load and preprocess** the time series data.
2) **Group by date** to get total daily expenses.
3) **Apply Moving Average Smoothing** (e.g., 3-day or 5-day window).
4) **Plot original vs smoothed series**.
5) Use **last smoothed value(s)** to forecast future expenses (naive method).

CODE:

```
data['date'] = pd.to_datetime(data['date'])


# Step 2: Aggregate total expenses per day

daily_expenses = data.groupby('date')['amount'].sum().reset_index()

daily_expenses.sort_values('date', inplace=True)


# Step 3: Apply Moving Average Smoothing (3-day window)

daily_expenses['Smoothed'] = daily_expenses['amount'].rolling(window=3).mean()


# Step 4: Plot original vs smoothed data

plt.figure(figsize=(12, 6))

plt.plot(daily_expenses['date'], daily_expenses['amount'], label='Original', marker='o')

plt.plot(daily_expenses['date'], daily_expenses['Smoothed'], label='Smoothed (3-day MA)',
linestyle='--', color='orange')

plt.title('Original vs Smoothed Daily Expenses')
plt.xlabel('Date')

plt.ylabel('Expenses')

plt.legend()

plt.grid(True)

plt.tight_layout()

plt.show()


# Step 5: Forecast next 3 days using average of last 3 smoothed values

last_3_smoothed = daily_expenses['Smoothed'].dropna().iloc[-3:]

forecast_next_3 = [last_3_smoothed.mean()] * 3


forecast_next_3
```

OUTPUT:



RESULT:

The program to apply moving average smoothing on time series is implemented successfully.