In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [39]:
```
data= pd.read_csv('ai_financial_market_daily_realistic_synthetic.csv')
data
```

Out[39]:

	Date	Company	R&D_Spending_USD_Mn	AI_Revenue_USD_Mn	AI_Revenue_Growth
0	2015-01-01	OpenAI	5.92	0.63	-36
1	2015-01-02	OpenAI	5.41	1.81	80
2	2015-01-03	OpenAI	4.50	0.61	-38
3	2015-01-04	OpenAI	5.45	0.95	-5
4	2015-01-05	OpenAI	3.40	1.48	48
...	...	...	...	...	
10954	2024-12-27	Meta	100.19	103.54	417
10955	2024-12-28	Meta	99.12	102.37	411
10956	2024-12-29	Meta	98.95	103.11	415
10957	2024-12-30	Meta	100.74	103.21	416
10958	2024-12-31	Meta	100.08	103.41	417

10959 rows × 7 columns

In [40]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10959 entries, 0 to 10958
Data columns (total 7 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Date                 10959 non-null  object 
 1   Company              10959 non-null  object 
 2   R&D_Spending_USD_Mn  10959 non-null  float64
 3   AI_Revenue_USD_Mn    10959 non-null  float64
 4   AI_Revenue_Growth_%  10959 non-null  float64
 5   Event                233 non-null    object 
 6   Stock_Impact_%       10959 non-null  float64
dtypes: float64(4), object(3)
memory usage: 599.4+ KB
```

In [ ]:
```
#Except Event there is no column with null values
```

In [41]:
```
#Converting Date to datetime format
data['Date']= pd.to_datetime(data['Date'])
```

In [43]:
```
data['Company'].unique()
```

Out[43]:
```
array(['OpenAI', 'Google', 'Meta'], dtype=object)
```

In [44]:
```
data['Year']= data['Date'].dt.year
data
```

Out[44]:

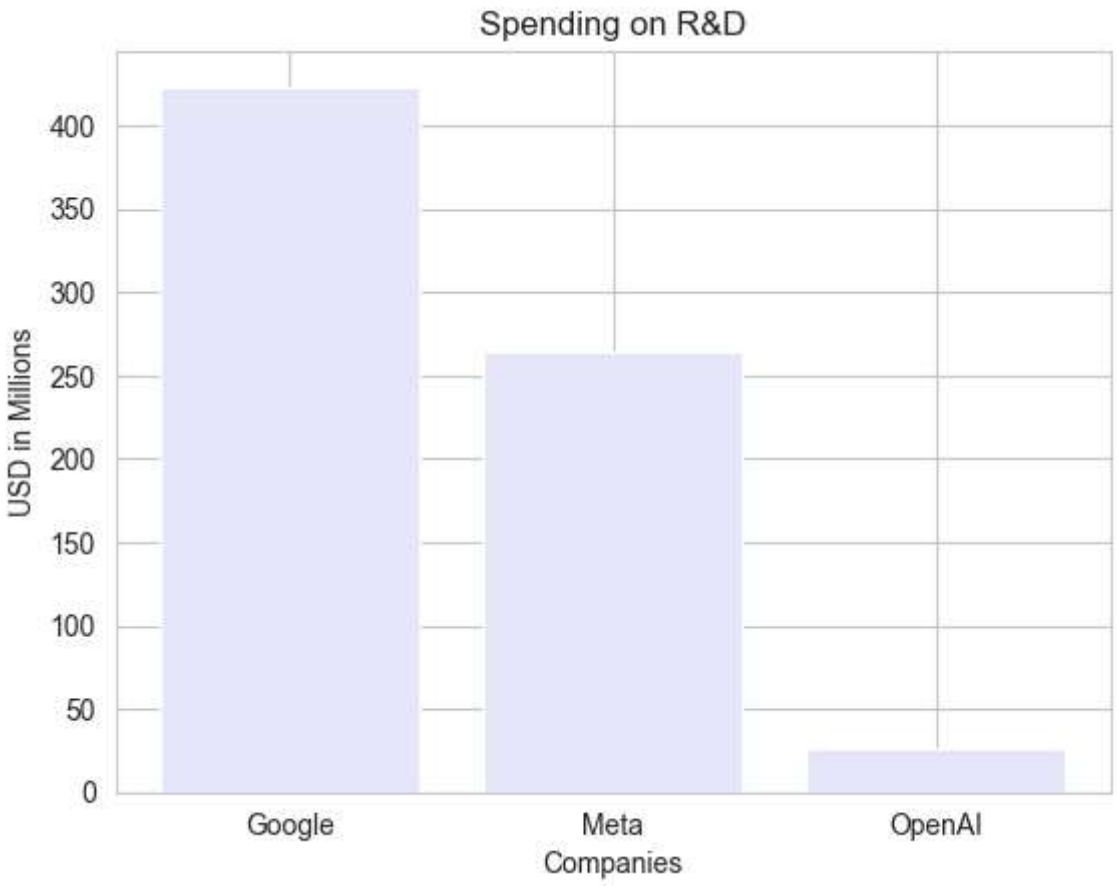
	Date	Company	R&D_Spending_USD_Mn	AI_Revenue_USD_Mn	AI_Revenue_Growth
0	2015-01-01	OpenAI	5.92	0.63	-36
1	2015-01-02	OpenAI	5.41	1.81	80
2	2015-01-03	OpenAI	4.50	0.61	-38
3	2015-01-04	OpenAI	5.45	0.95	-5
4	2015-01-05	OpenAI	3.40	1.48	48
...	...	...	...	...	
10954	2024-12-27	Meta	100.19	103.54	417
10955	2024-12-28	Meta	99.12	102.37	411
10956	2024-12-29	Meta	98.95	103.11	415
10957	2024-12-30	Meta	100.74	103.21	416
10958	2024-12-31	Meta	100.08	103.41	417

10959 rows × 8 columns

In [78]:
```
#How much amount the companies spent on R & D?
RD= data.groupby('Company')['R&D_Spending_USD_Mn'].sum()
RD
```

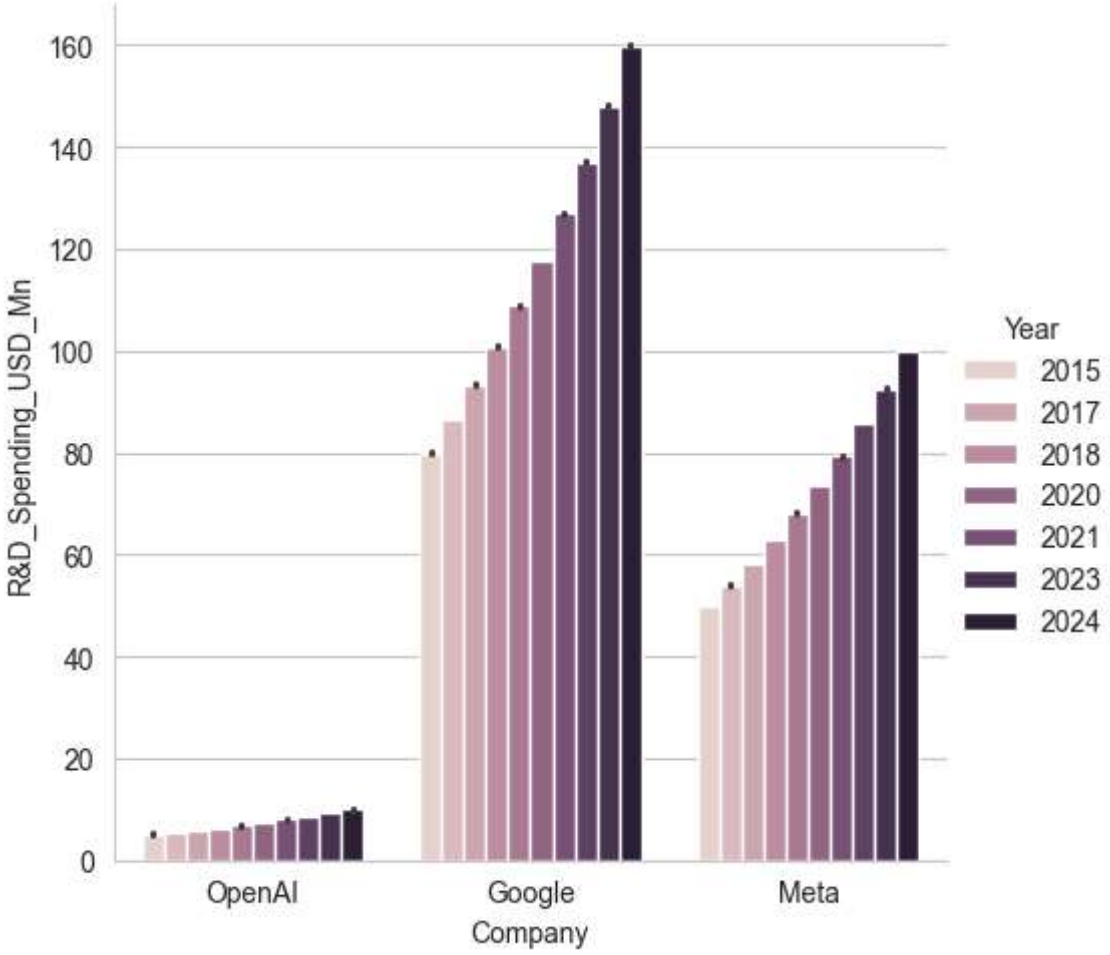
Out[78]:
```
Company
Google    423341.14
Meta      264533.07
OpenAI     26482.77
Name: R&D_Spending_USD_Mn, dtype: float64
```

In [46]:
```
plt.bar(RD.index, RD.values, color= 'lavender')
plt.title('Spending on R&D')
plt.xlabel('Companies')
plt.ylabel('USD in Millions')
plt.show()
```



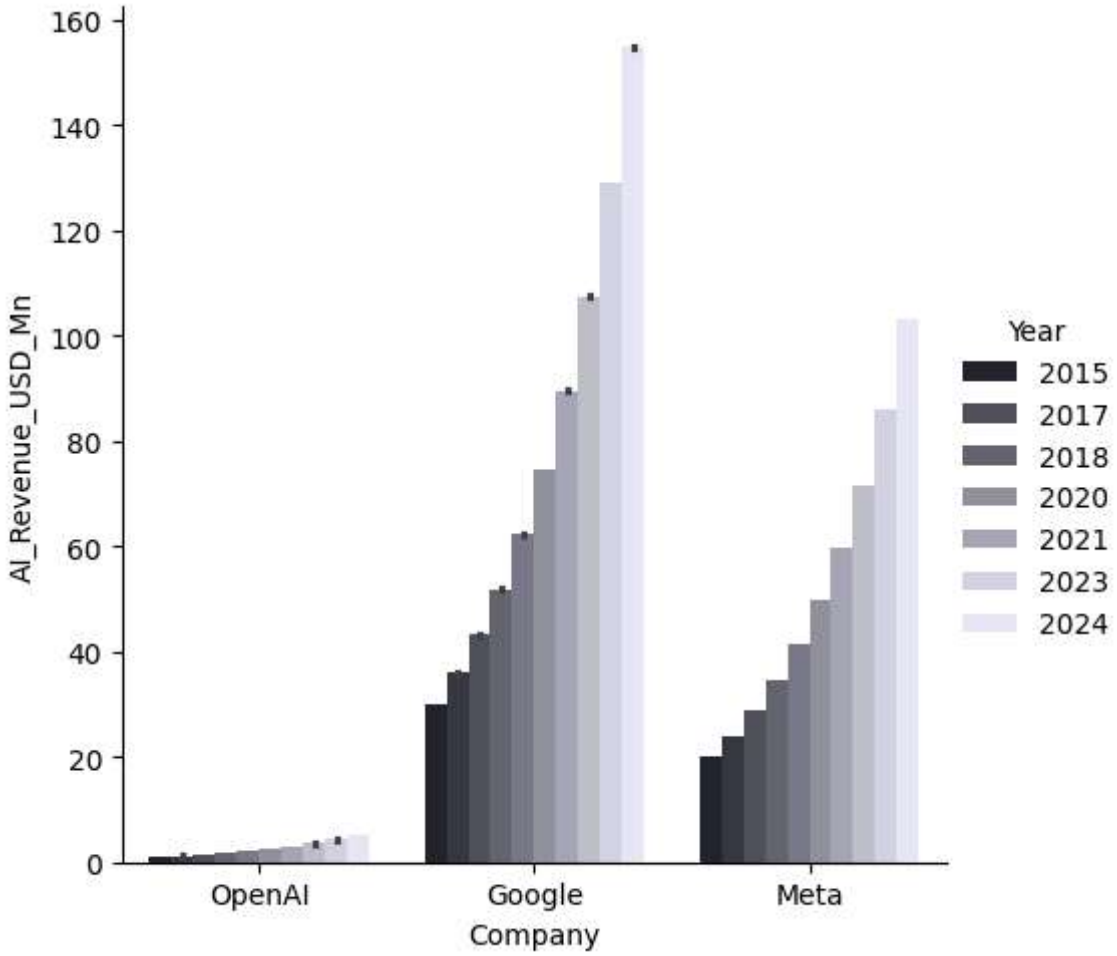
In [ ]:
```
#How much amount the companies spent on R & D Year wise?
```

In [47]:
```
sns.catplot(x='Company', y='R&D_Spending_USD_Mn', kind= 'bar', data=data, hue='Year
plt.show()
```



In [ ]:
```
#Revenue Earned by the companies yearwise
```

In [44]:
```
sns.catplot(x= 'Company', y= 'AI_Revenue_USD_Mn', data=data, hue= 'Year', kind= 'ba
plt.show()
```

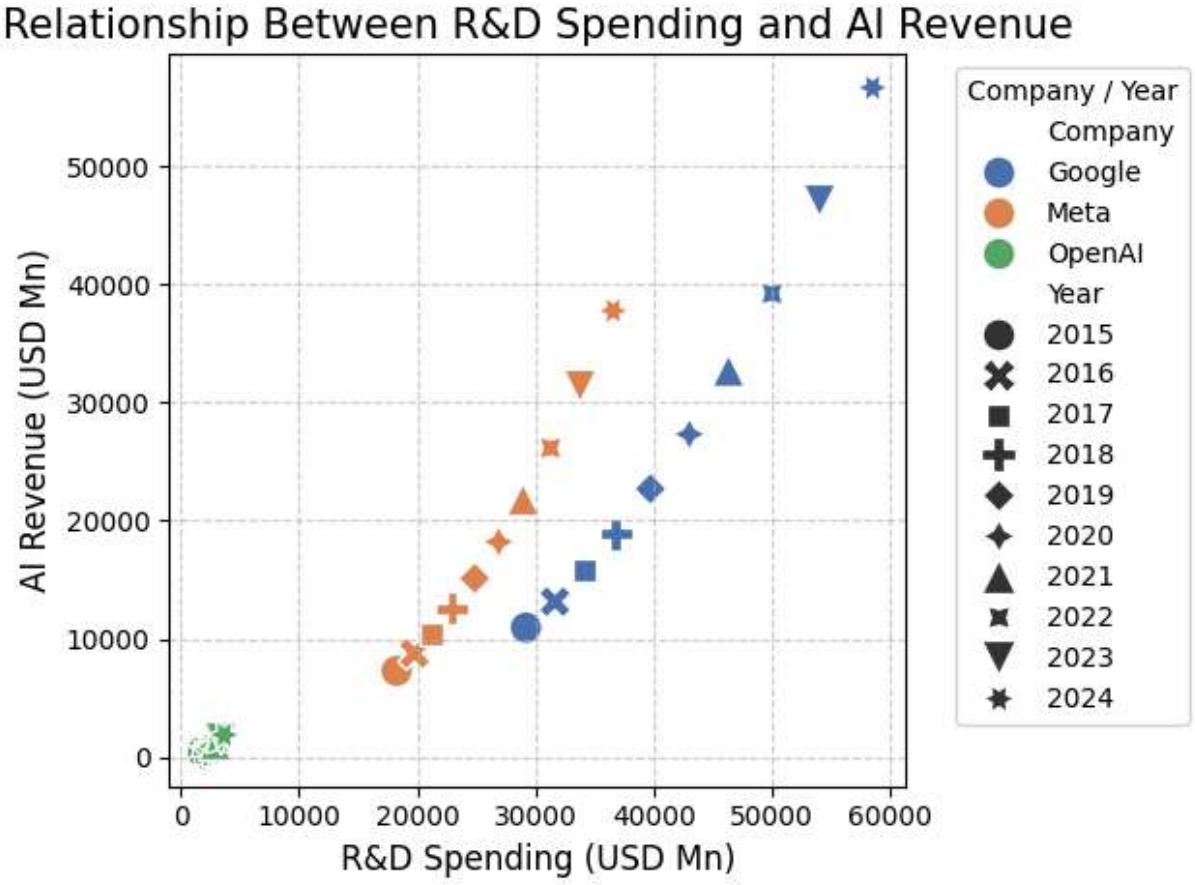


In [48]:
```
data_grouped = data.groupby(['Company', 'Year'])[['R&D_Spending_USD_Mn', 'AI_Revenu
data_grouped
```

Out[48]:

	Company	Year	R&D_Spending_USD_Mn	AI_Revenue_USD_Mn
0	Google	2015	29189.88	10962.63
1	Google	2016	31643.54	13156.27
2	Google	2017	34069.58	15756.37
3	Google	2018	36793.71	18924.12
4	Google	2019	39719.53	22701.64
5	Google	2020	43016.73	27328.95
6	Google	2021	46334.04	32705.27
7	Google	2022	50009.88	39228.04
8	Google	2023	54037.28	47088.80
9	Google	2024	58526.97	56646.29
10	Meta	2015	18248.09	7304.13
11	Meta	2016	19763.85	8768.17
12	Meta	2017	21260.37	10501.44
13	Meta	2018	22947.79	12596.20
14	Meta	2019	24861.99	15127.71
15	Meta	2020	26884.91	18214.22
16	Meta	2021	28948.55	21794.18
17	Meta	2022	31273.90	26151.89
18	Meta	2023	33777.24	31385.74
19	Meta	2024	36566.38	37778.14
20	OpenAI	2015	1866.26	356.45
21	OpenAI	2016	1942.55	436.76
22	OpenAI	2017	2145.17	518.99
23	OpenAI	2018	2260.66	644.12
24	OpenAI	2019	2489.99	750.97
25	OpenAI	2020	2700.89	913.02
26	OpenAI	2021	2924.73	1091.69
27	OpenAI	2022	3106.39	1294.31
28	OpenAI	2023	3379.40	1567.10
29	OpenAI	2024	3666.73	1889.48

In [13]:
```
sns.scatterplot(
    data=data_grouped,
    x='R&D_Spending_USD_Mn',
    y='AI_Revenue_USD_Mn',
    hue='Company',
    style='Year',
    s=150,
    palette='deep'
)

# Customizing the chart
plt.title('Relationship Between R&D Spending and AI Revenue', fontsize=15)
plt.xlabel('R&D Spending (USD Mn)', fontsize=12)
plt.ylabel('AI Revenue (USD Mn)', fontsize=12)
plt.legend(title='Company / Year', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()

plt.show()
```



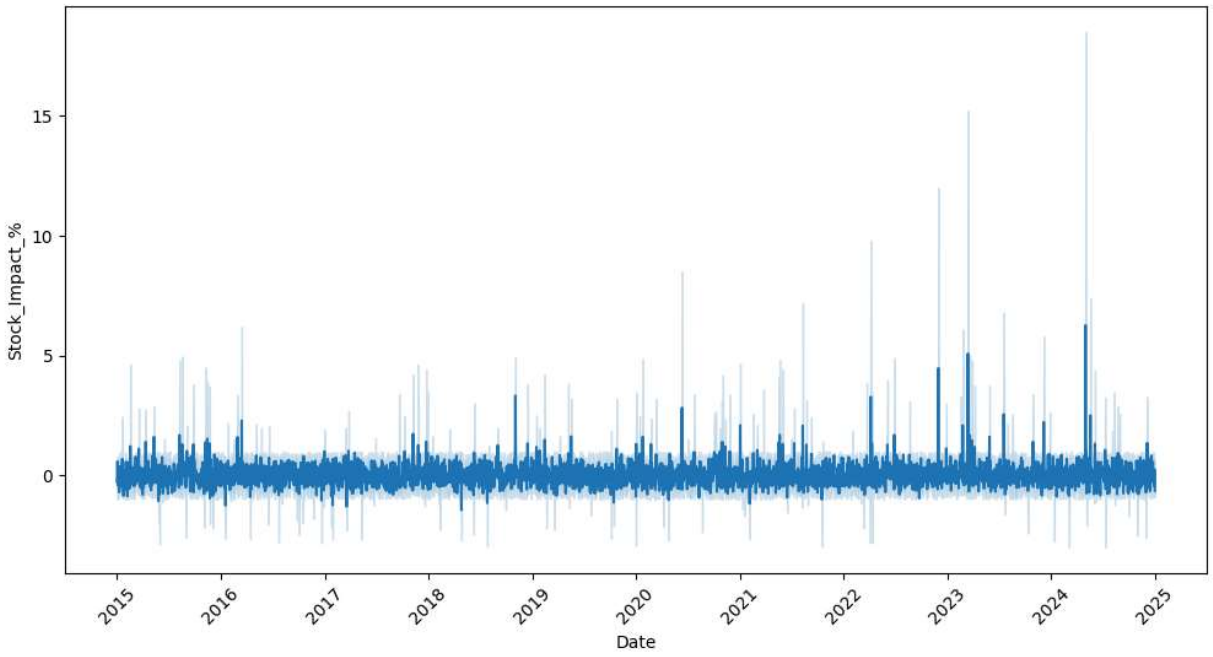
In [ ]:
```
#Datewise Impact on the Stock
```

In [45]:
```
data.head()
```

Out[45]:

	Date	Company	R&D_Spending_USD_Mn	AI_Revenue_USD_Mn	AI_Revenue_Growth_%
0	2015-01-01	OpenAI	5.92	0.63	-36.82
1	2015-01-02	OpenAI	5.41	1.81	80.59
2	2015-01-03	OpenAI	4.50	0.61	-38.88
3	2015-01-04	OpenAI	5.45	0.95	-5.34
4	2015-01-05	OpenAI	3.40	1.48	48.45

In [10]:
```
plt.figure(figsize= (12,6))
sns.lineplot(x='Date', y='Stock_Impact_%', data=data)
plt.xticks(rotation=45)
plt.show()
```



In [ ]:
```
#Lets create 3 DF for simplicity companiwise
```

In [13]:
```
data_openAI= data[data['Company']=='OpenAI']
data_openAI
```

Out[13]:

	Date	Company	R&D_Spending_USD_Mn	AI_Revenue_USD_Mn	AI_Revenue_Growth_
0	2015-01-01	OpenAI	5.92	0.63	-36.8
1	2015-01-02	OpenAI	5.41	1.81	80.5
2	2015-01-03	OpenAI	4.50	0.61	-38.8
3	2015-01-04	OpenAI	5.45	0.95	-5.3
4	2015-01-05	OpenAI	3.40	1.48	48.4
...	...	...	...	...	
3648	2024-12-27	OpenAI	10.06	4.71	370.6
3649	2024-12-28	OpenAI	9.67	5.32	432.1
3650	2024-12-29	OpenAI	9.17	5.46	445.7
3651	2024-12-30	OpenAI	10.36	6.31	530.8
3652	2024-12-31	OpenAI	10.24	4.92	391.6

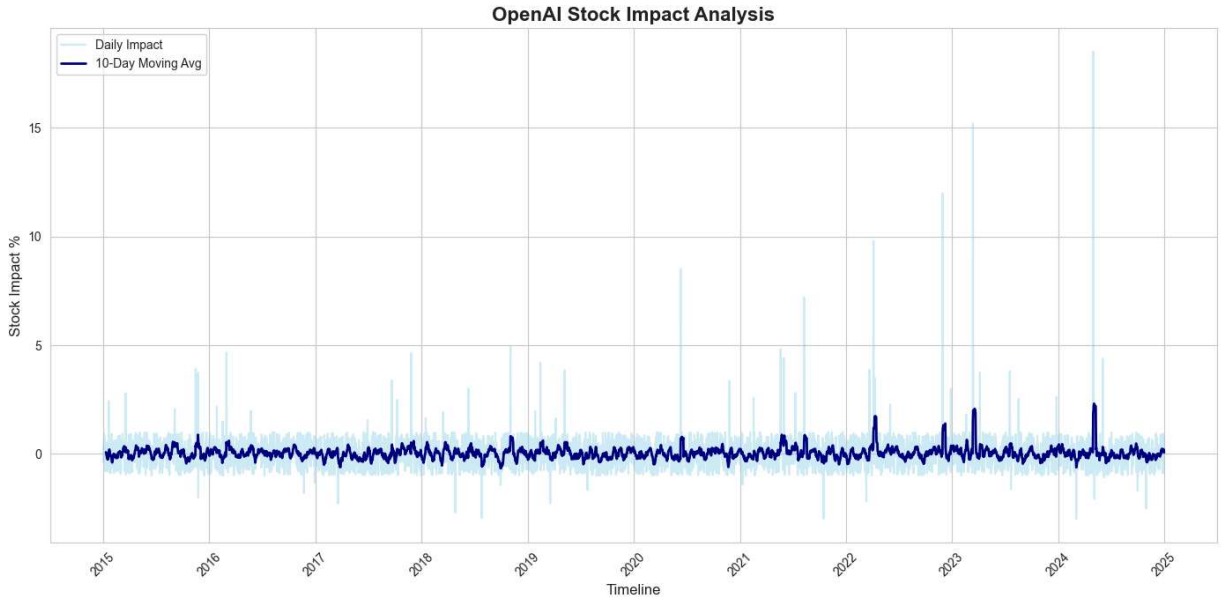
3653 rows × 8 columns

In [21]:

```
# Theme
sns.set_style("whitegrid")

plt.figure(figsize=(14, 7))


sns.lineplot(x='Date', y='Stock_Impact_%', data=data_openAI, color='skyblue', alpha

#Rolling Trend
data_openAI['Rolling_Avg'] = data_openAI['Stock_Impact_%'].rolling(window=10).mean(
sns.lineplot(x='Date', y='Rolling_Avg', data=data_openAI, color='navy', linewidth=2


plt.title('OpenAI Stock Impact Analysis', fontsize=16, fontweight='bold')
plt.xlabel('Timeline', fontsize=12)
plt.ylabel('Stock Impact %', fontsize=12)
plt.xticks(rotation=45)

plt.legend()
```

```
plt.tight_layout()
plt.show()
```

```
C:\Users\sawru\AppData\Local\Temp\ipykernel_16164\2267254980.py:10: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  data_openAI['Rolling_Avg'] = data_openAI['Stock_Impact_%'].rolling(window=10).mean
()
```



In [14]:
```
data_google= data[data['Company']=='Google']
data_google
```

Out[14]:

	Date	Company	R&D_Spending_USD_Mn	AI_Revenue_USD_Mn	AI_Revenue_Growth_
3653	2015-01-01	Google	79.89	30.19	0.6
3654	2015-01-02	Google	78.99	30.44	1.4
3655	2015-01-03	Google	79.20	30.46	1.5
3656	2015-01-04	Google	79.59	30.55	1.8
3657	2015-01-05	Google	81.50	30.59	1.9
...	...	...	...	...	
7301	2024-12-27	Google	162.16	155.36	417.8
7302	2024-12-28	Google	159.69	154.47	414.8
7303	2024-12-29	Google	161.69	154.59	415.3
7304	2024-12-30	Google	158.48	155.05	416.8
7305	2024-12-31	Google	159.48	154.77	415.9

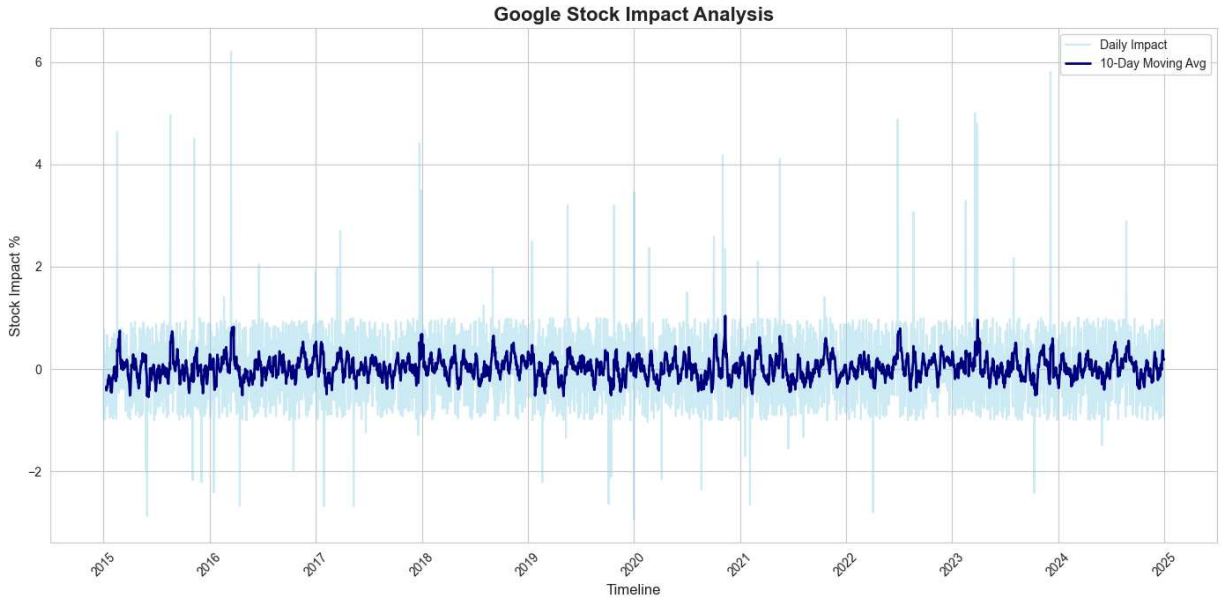
3653 rows × 8 columns

In [ ]:

In [19]:
```
# Theme
sns.set_style("whitegrid")

plt.figure(figsize=(14, 7))


sns.lineplot(x='Date', y='Stock_Impact_%', data=data_google, color='skyblue', alpha

#Rolling Trend
data_google['Rolling_Avg'] = data_google['Stock_Impact_%'].rolling(window=10).mean(
sns.lineplot(x='Date', y='Rolling_Avg', data=data_google, color='navy', linewidth=2


plt.title('Google Stock Impact Analysis', fontsize=16, fontweight='bold')
plt.xlabel('Timeline', fontsize=12)
plt.ylabel('Stock Impact %', fontsize=12)
plt.xticks(rotation=45)
```

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

```
C:\Users\sawru\AppData\Local\Temp\ipykernel_16164\1308202840.py:10: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  data_google['Rolling_Avg'] = data_google['Stock_Impact_%'].rolling(window=10).mean
()
```



In [15]:
```
data_meta= data[data['Company']=='Meta']
data_meta
```

Out[15]:

	Date	Company	R&D_Spending_USD_Mn	AI_Revenue_USD_Mn	AI_Revenue_Growth
7306	2015-01-01	Meta	50.39	18.95	-5
7307	2015-01-02	Meta	49.80	19.77	-1
7308	2015-01-03	Meta	49.09	19.96	-0
7309	2015-01-04	Meta	50.66	20.48	2
7310	2015-01-05	Meta	51.36	19.84	-0
...	...	...	...	...	
10954	2024-12-27	Meta	100.19	103.54	417
10955	2024-12-28	Meta	99.12	102.37	411
10956	2024-12-29	Meta	98.95	103.11	415
10957	2024-12-30	Meta	100.74	103.21	416
10958	2024-12-31	Meta	100.08	103.41	417

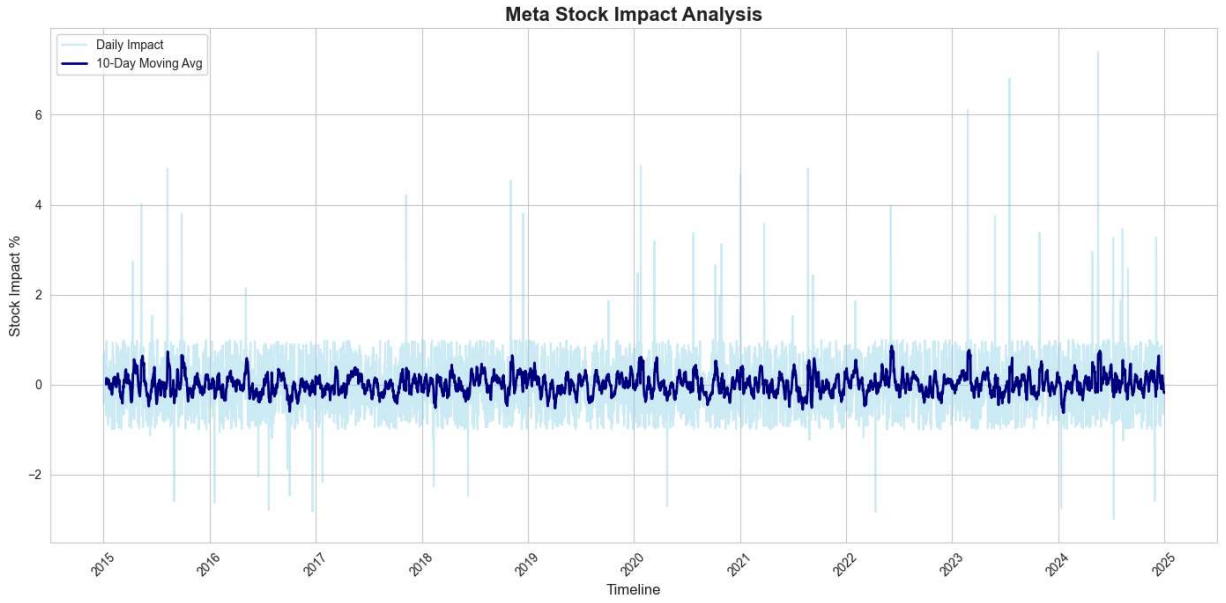
3653 rows × 8 columns

In [18]:

```
# Theme
sns.set_style("whitegrid")

plt.figure(figsize=(14, 7))


sns.lineplot(x='Date', y='Stock_Impact_%', data=data_meta, color='skyblue', alpha=0

#Rolling Trend
data_meta['Rolling_Avg'] = data_meta['Stock_Impact_%'].rolling(window=10).mean()
sns.lineplot(x='Date', y='Rolling_Avg', data=data_meta, color='navy', linewidth=2,


plt.title('Meta Stock Impact Analysis', fontsize=16, fontweight='bold')
plt.xlabel('Timeline', fontsize=12)
plt.ylabel('Stock Impact %', fontsize=12)
plt.xticks(rotation=45)

plt.legend()
```

```
plt.tight_layout()
plt.show()
```

```
C:\Users\sawru\AppData\Local\Temp\ipykernel_16164\1290863872.py:10: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  data_meta['Rolling_Avg'] = data_meta['Stock_Impact_%'].rolling(window=10).mean()
```



In [ ]:
```
#Events when Maximum Stock Impact was observed
```

In [23]:
```
data_openAI.sort_values(by= 'Stock_Impact_%',ascending= False)
```

Out[23]:

	Date	Company	R&D_Spending_USD_Mn	AI_Revenue_USD_Mn	AI_Revenue_Growth_
3408	2024-05-01	OpenAI	10.91	5.34	434.2
2994	2023-03-14	OpenAI	7.78	4.05	304.5
2890	2022-11-30	OpenAI	10.60	3.18	217.7
2652	2022-04-06	OpenAI	9.24	3.48	247.9
1988	2020-06-11	OpenAI	5.90	2.62	161.5
...	...	...	...	...	
3590	2024-10-30	OpenAI	11.05	5.27	427.5
1212	2018-04-27	OpenAI	5.38	1.76	76.4
1303	2018-07-27	OpenAI	7.44	0.97	-2.6
2480	2021-10-16	OpenAI	8.39	2.78	178.2
3350	2024-03-04	OpenAI	10.82	4.77	376.5

3653 rows × 9 columns

In [24]:
```
data_google.sort_values(by= 'Stock_Impact_%',ascending= False)
```

Out[24]:

	Date	Company	R&D_Spending_USD_Mn	AI_Revenue_USD_Mn	AI_Revenue_Growth_
4092	2016-03-15	Google	84.56	36.22	20.7
6914	2023-12-06	Google	146.59	129.17	330.5
6654	2023-03-21	Google	149.34	129.76	332.5
3883	2015-08-19	Google	79.27	30.89	2.9
6388	2022-06-28	Google	137.71	107.96	259.8
...	...	...	...	...	
4122	2016-04-14	Google	87.19	36.08	20.2
4412	2017-01-29	Google	92.21	43.25	44.1
6303	2022-04-04	Google	138.00	106.81	256.0
3803	2015-05-31	Google	81.14	29.30	-2.3
5479	2020-01-01	Google	117.24	74.82	149.4

3653 rows × 9 columns

In [25]:
```
data_meta.sort_values(by= 'Stock_Impact_%',ascending= False)
```

Out[25]:

	Date	Company	R&D_Spending_USD_Mn	AI_Revenue_USD_Mn	AI_Revenue_Growth
10731	2024-05-18	Meta	103.64	103.05	415
10426	2023-07-18	Meta	92.44	85.67	328
10282	2023-02-24	Meta	93.71	86.98	334
9156	2020-01-25	Meta	72.73	49.10	145
7526	2015-08-09	Meta	48.97	19.95	-0
...	...	...	...	...	
10604	2024-01-12	Meta	99.37	104.28	421
7875	2016-07-23	Meta	53.76	23.97	19
8026	2016-12-21	Meta	53.18	23.42	17
9964	2022-04-12	Meta	84.90	71.99	259
10785	2024-07-11	Meta	100.78	102.92	414

3653 rows × 9 columns

In [ ]:
```
#AI Revenue Growth of the companies
```

In [26]:
```
data
```

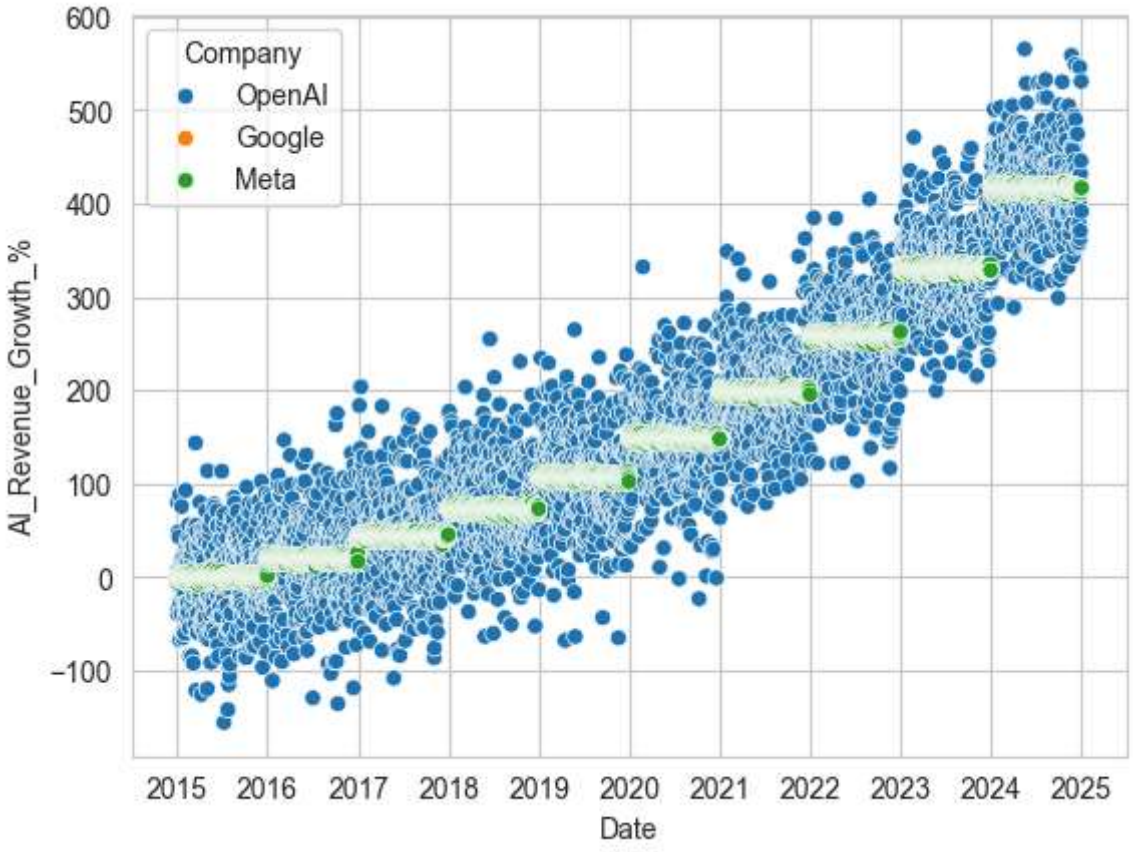
Out[26]:

	Date	Company	R&D_Spending_USD_Mn	AI_Revenue_USD_Mn	AI_Revenue_Growth
0	2015-01-01	OpenAI	5.92	0.63	-36
1	2015-01-02	OpenAI	5.41	1.81	80
2	2015-01-03	OpenAI	4.50	0.61	-38
3	2015-01-04	OpenAI	5.45	0.95	-5
4	2015-01-05	OpenAI	3.40	1.48	48
...	...	...	...	...	
10954	2024-12-27	Meta	100.19	103.54	417
10955	2024-12-28	Meta	99.12	102.37	411
10956	2024-12-29	Meta	98.95	103.11	415
10957	2024-12-30	Meta	100.74	103.21	416
10958	2024-12-31	Meta	100.08	103.41	417

10959 rows × 8 columns

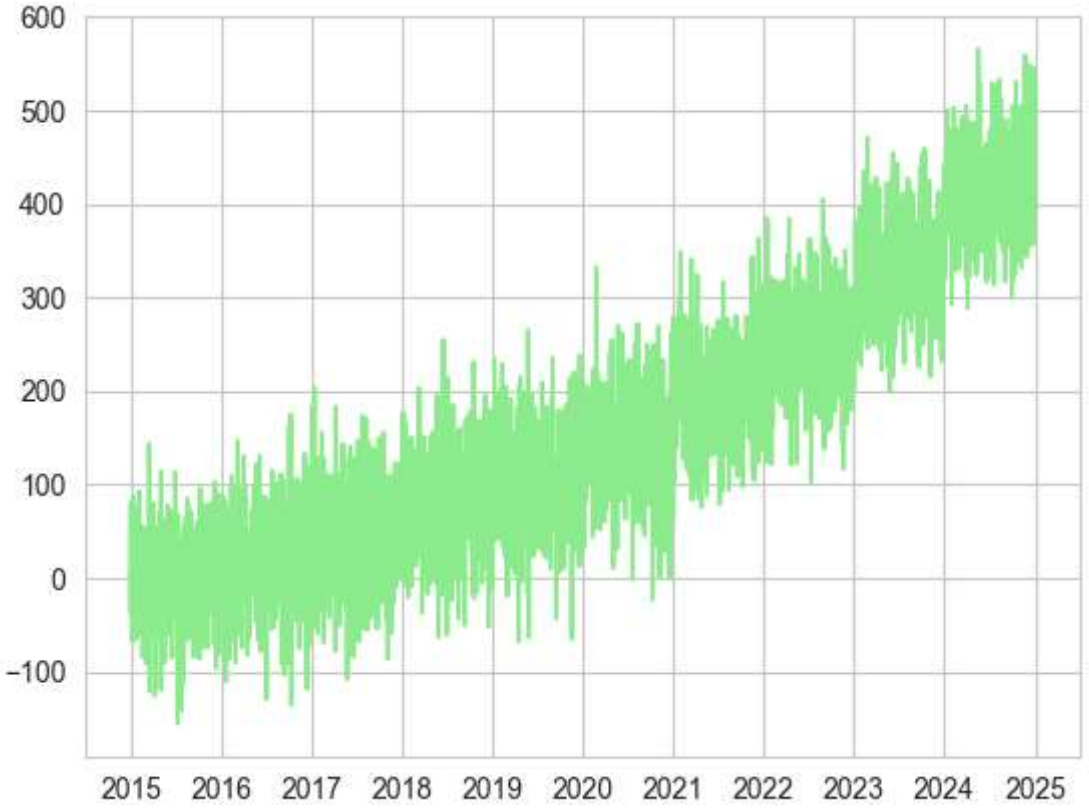
In [30]:
```
sns.scatterplot(x= 'Date', y= 'AI_Revenue_Growth_%', data=data, hue= 'Company')
```

Out[30]:
```
<Axes: xlabel='Date', ylabel='AI_Revenue_Growth_%'>
```

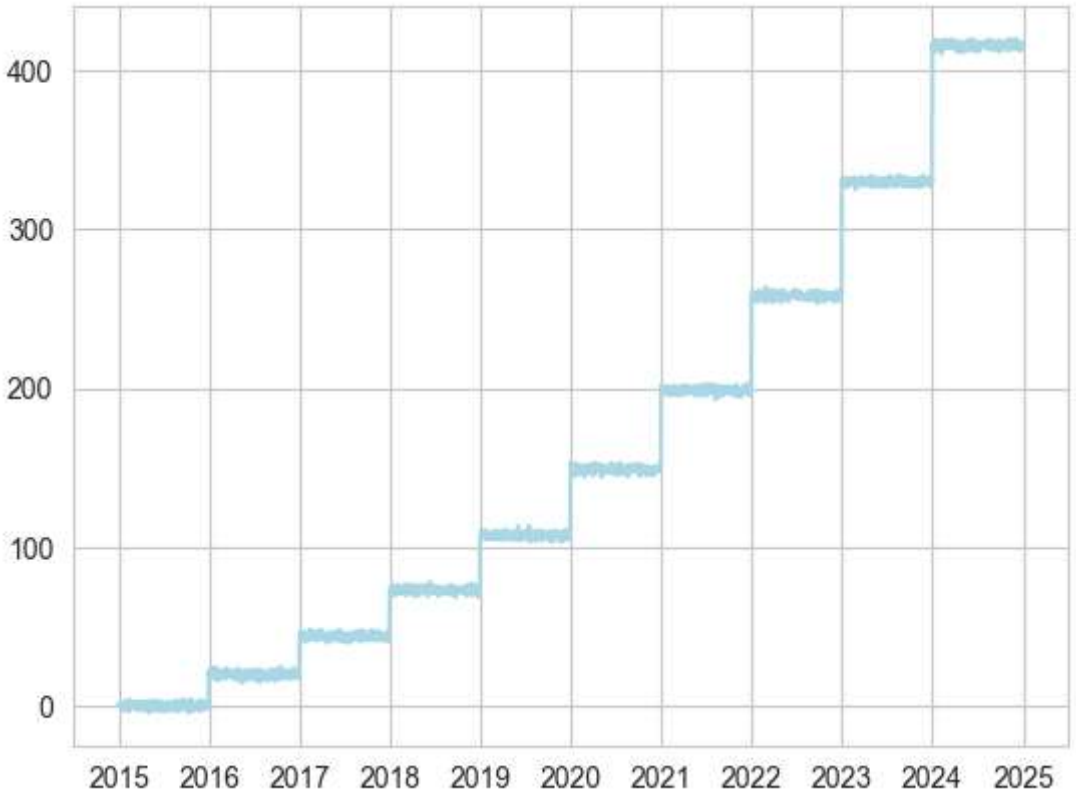


In [ ]:
```
#AI Revenue Growth of the companies
```

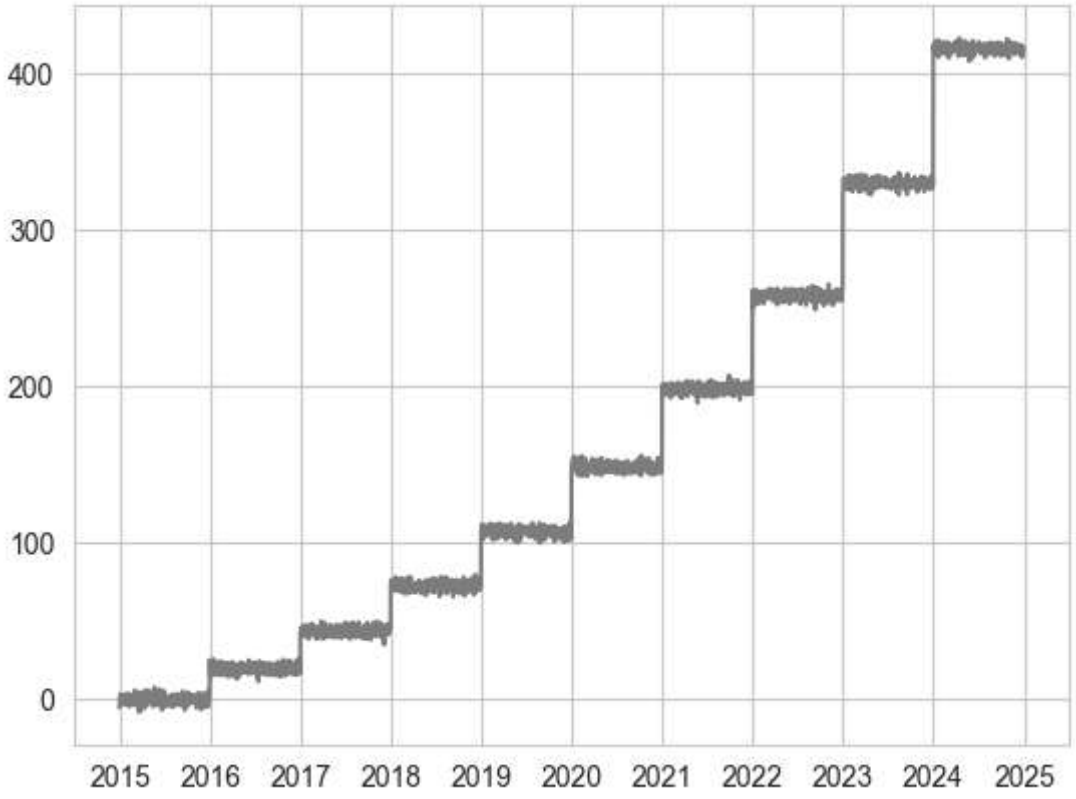
In [32]:
```
plt.plot(data_openAI['Date'], data_openAI['AI_Revenue_Growth_%'], color= 'lightgree
plt.show()
```



In [94]:
```
plt.plot(data_google['Date'], data_google['AI_Revenue_Growth_%'], color= 'lightblue
plt.show()
```

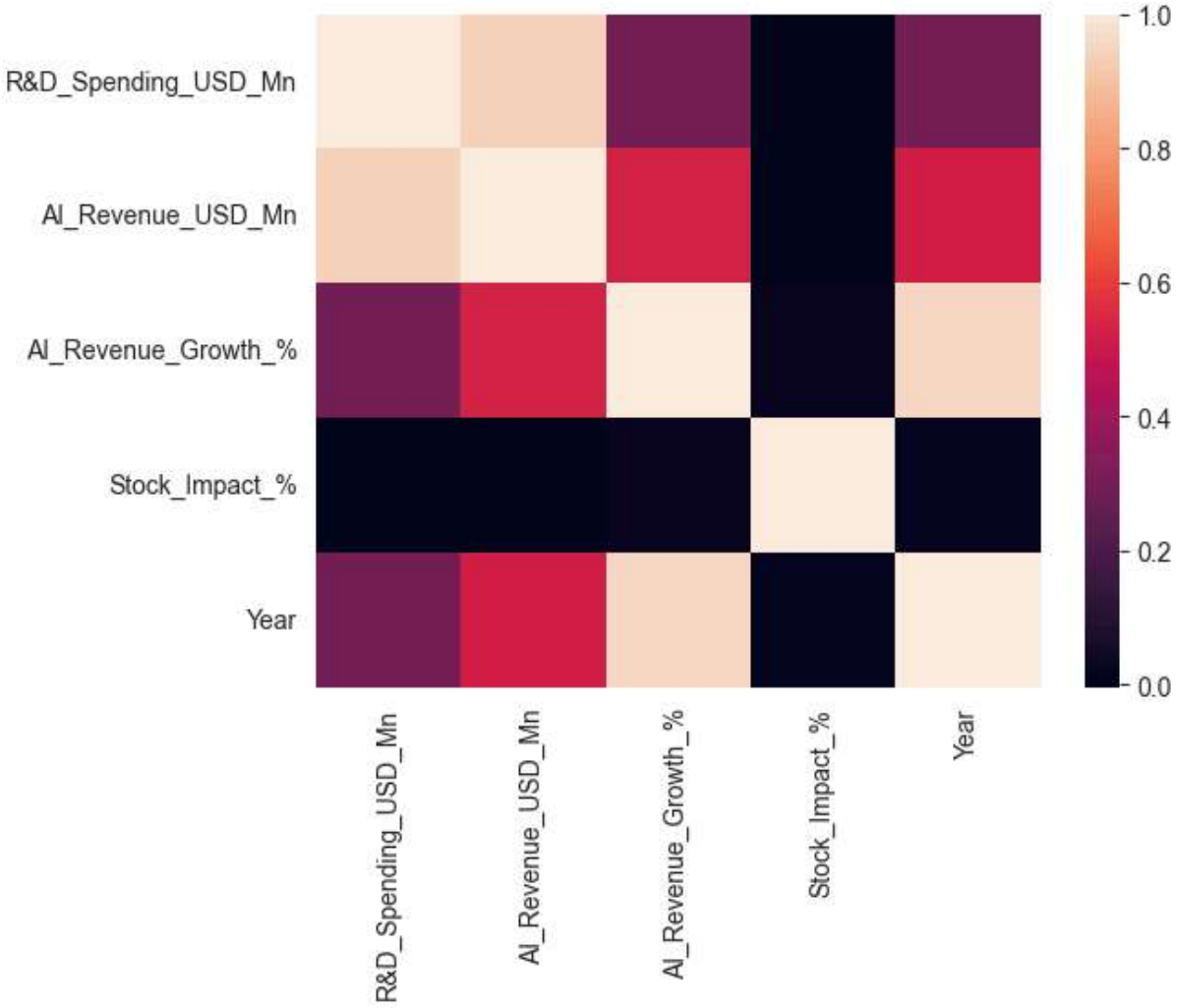


In [34]:
```
plt.plot(data_meta['Date'], data_meta['AI_Revenue_Growth_%'], color= 'grey')
plt.show()
```



In [38]:
```
sns.heatmap(data.corr(numeric_only=True))
```

Out[38]: <Axes: >



In [ ]:
```
#Expenditure vs Revenue year-by-year
```

In [72]:
```
data_openAI.groupby('Date')[['R&D_Spending_USD_Mn', 'AI_Revenue_USD_Mn']].sum()
```

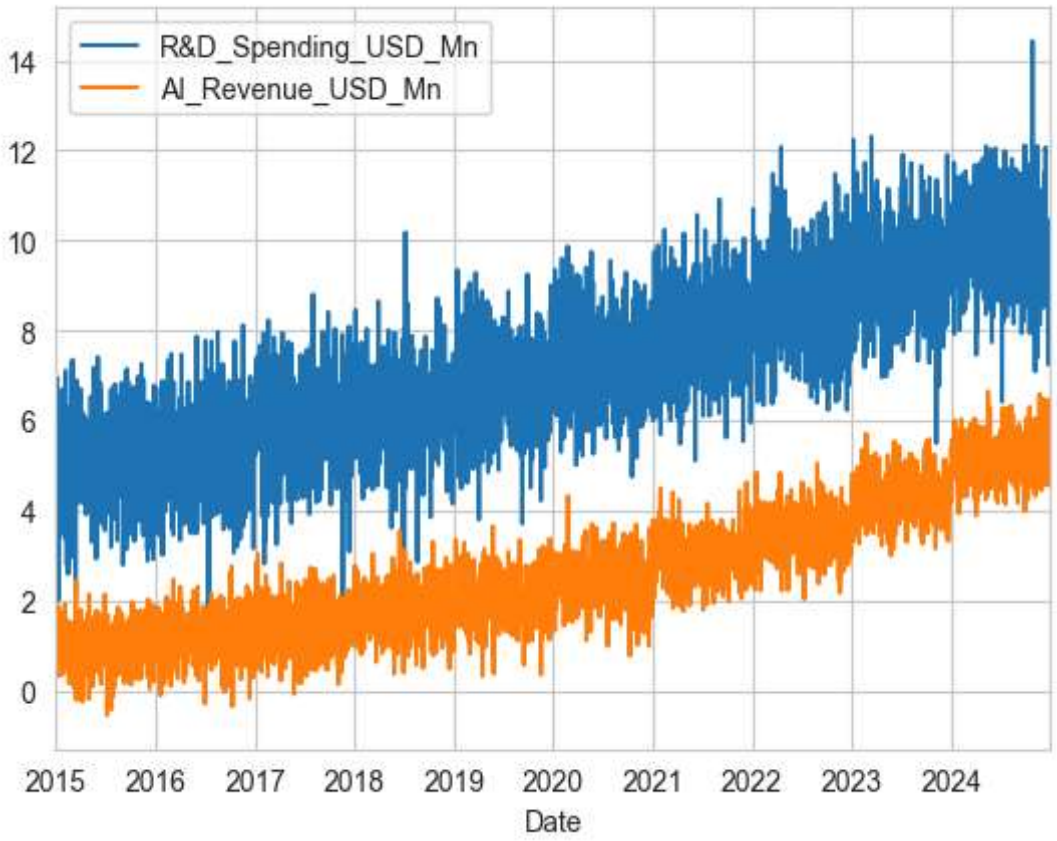
Out[72]:

Date	R&D_Spending_USD_Mn	AI_Revenue_USD_Mn
2015-01-01	5.92	0.63
2015-01-02	5.41	1.81
2015-01-03	4.50	0.61
2015-01-04	5.45	0.95
2015-01-05	3.40	1.48
...	...	...
2024-12-27	10.06	4.71
2024-12-28	9.67	5.32
2024-12-29	9.17	5.46
2024-12-30	10.36	6.31
2024-12-31	10.24	4.92

3653 rows × 2 columns

In [71]:
```
data_openAI.groupby('Date')[['R&D_Spending_USD_Mn', 'AI_Revenue_USD_Mn']].sum().plo
```

Out[71]: <Axes: xlabel='Date'>



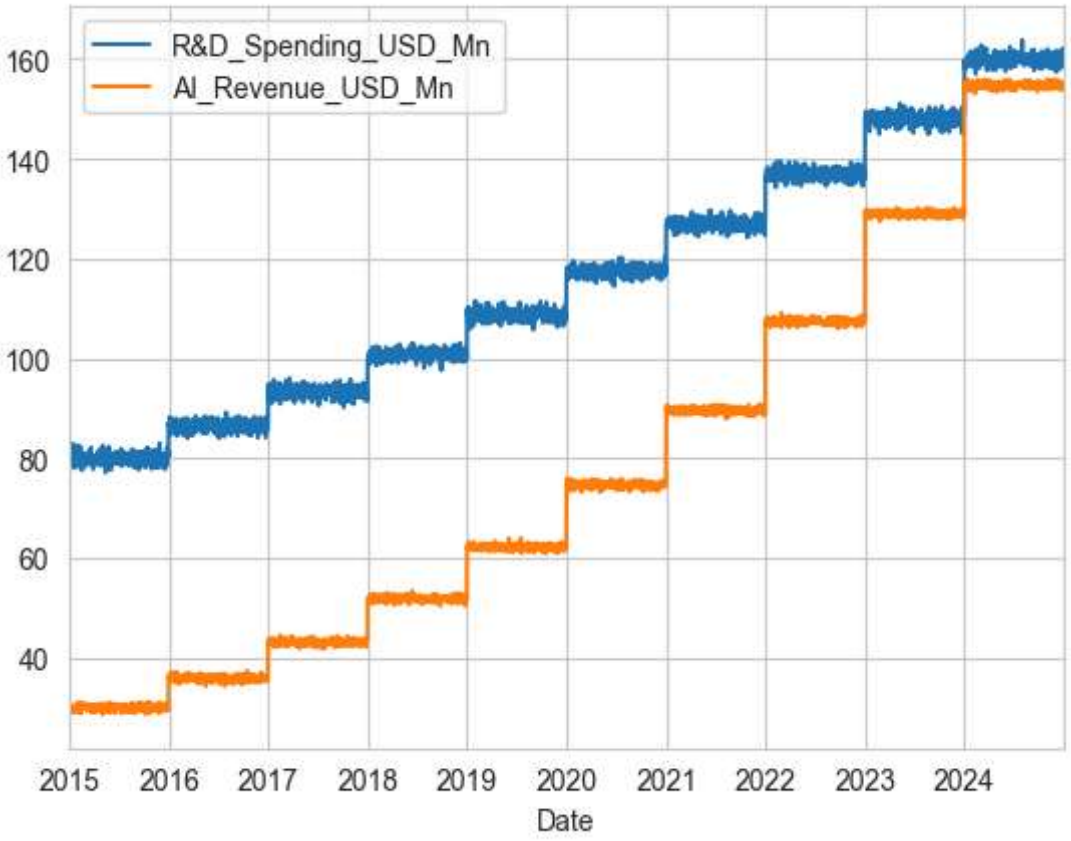
In [73]:
```
data_google.groupby('Date')[['R&D_Spending_USD_Mn', 'AI_Revenue_USD_Mn']].sum()
```

Out[73]:

Date	R&D_Spending_USD_Mn	AI_Revenue_USD_Mn
2015-01-01	79.89	30.19
2015-01-02	78.99	30.44
2015-01-03	79.20	30.46
2015-01-04	79.59	30.55
2015-01-05	81.50	30.59
...	...	...
2024-12-27	162.16	155.36
2024-12-28	159.69	154.47
2024-12-29	161.69	154.59
2024-12-30	158.48	155.05
2024-12-31	159.48	154.77

3653 rows × 2 columns

In [75]:
```
data_google.groupby('Date')[['R&D_Spending_USD_Mn', 'AI_Revenue_USD_Mn']].sum().plo
plt.show()
```



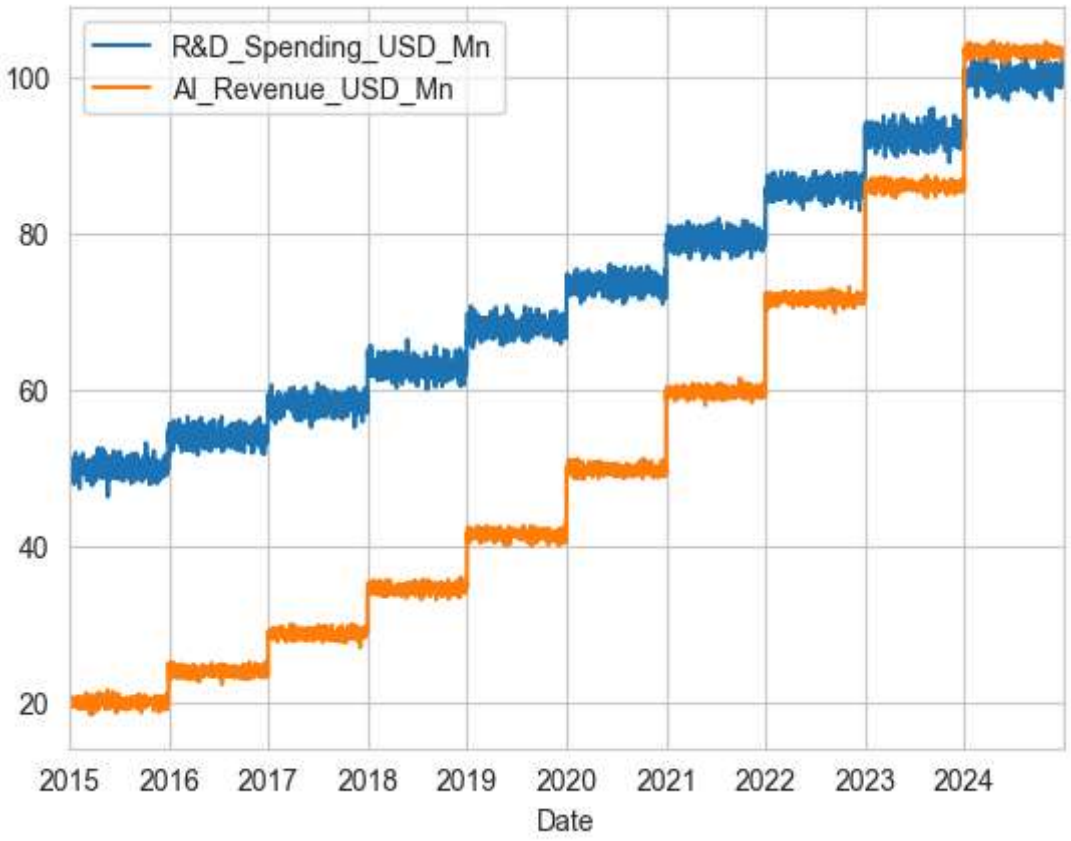
In [76]:
```
data_meta.groupby('Date')[['R&D_Spending_USD_Mn', 'AI_Revenue_USD_Mn']].sum()
```

Out[76]:

Date	R&D_Spending_USD_Mn	AI_Revenue_USD_Mn
2015-01-01	50.39	18.95
2015-01-02	49.80	19.77
2015-01-03	49.09	19.96
2015-01-04	50.66	20.48
2015-01-05	51.36	19.84
...	...	...
2024-12-27	100.19	103.54
2024-12-28	99.12	102.37
2024-12-29	98.95	103.11
2024-12-30	100.74	103.21
2024-12-31	100.08	103.41

3653 rows × 2 columns

In [77]:
```
data_meta.groupby('Date')[['R&D_Spending_USD_Mn', 'AI_Revenue_USD_Mn']].sum().plot(
plt.show()
```



In [84]:
```
#Efficiency Metric

data_openAI['Efficiency_Ratio'] = data_openAI['AI_Revenue_USD_Mn'] / data_openAI['R
```

```
C:\Users\sawru\AppData\Local\Temp\ipykernel_16164\3865025352.py:3: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  data_openAI['Efficiency_Ratio'] = data_openAI['AI_Revenue_USD_Mn'] / data_openAI
['R&D_Spending_USD_Mn']
```

In [90]:
```
data_openAI = data[data['Company'] == 'OpenAI'].copy()

# Ab aap purana wala code bhi chalayenge toh error nahi aayega
data_openAI['Efficiency_Ratio'] = data_openAI['AI_Revenue_USD_Mn'] / data_openAI['R
data_openAI['Efficiency_Ratio'].mean()
```

Out[90]:
```
np.float64(0.34258894227308895)
```

In [91]:
```
data_google = data[data['Company'] == 'Google'].copy()

# Ab aap purana wala code bhi chalayenge toh error nahi aayega
```

```
data_google['Efficiency_Ratio'] = data_google['AI_Revenue_USD_Mn'] / data_google['R
data_google['Efficiency_Ratio'].mean()
```

Out[91]: np.float64(0.6305355973171272)

In [93]:
```
data_meta = data[data['Company'] == 'Meta'].copy()

# Ab aap purana wala code bhi chalayenge toh error nahi aayega
data_meta['Efficiency_Ratio'] = data_meta['AI_Revenue_USD_Mn'] / data_meta['R&D_Spe
data_meta['Efficiency_Ratio'].mean()
```

Out[93]: np.float64(0.6725879763975073)

In [ ]:
```
Strategic Analysis: R&D Expenditure vs. AI Revenue Performance
1. R&D Expenditure & Financial Efficiency
Data ke mutabiq, R&D spending aur revenue generation ke beech ek direct correlation

Dominant Spender: Google sabse zyada invest kar raha hai ($423,341.14 Mn), jo unke

Efficiency Leader: Meta sabse efficient performer bankar ubhri hai, jiska Efficienc

Comparative Efficiency: Google (0.63) aur OpenAI (0.34) ke muqable Meta ne spending

Growth Constraint: Kisi bhi company ne R&D costs ko badhaye bina revenue grow nahi

2. Stock Market & Volatility Dynamics
Stock impact analysis se market sentiment ka pata chalta hai:

Stable Performer: OpenAI sabse kam volatile nazar aati hai, jo investors ke liye ek

High Volatility: Google market mein sabse zyada volatile hai, jahan news aur events

3. Growth Patterns: OpenAI vs. Tech Giants
Growth trajectory mein OpenAI aur traditional tech giants ke beech ek bada contrast

Event-Driven Growth (Google & Meta): In dono companies ki growth "Inconsistent" hai

Continuous Growth (OpenAI): Inke vipreet, OpenAI ka growth model zyada gradual aur

4. Efficiency Trends & Future Outlook
Revenue vs. Spending ke gap ka analysis future profitability ki taraf ishara karta

Positive Convergence (Google): Google ki efficiency lagatar badh rahi hai kyunki Sp

The Break-even Milestone (Meta): Meta ne ek bada milestone achieve kiya hai jahan R

Variable Performance (OpenAI): OpenAI ki efficiency thodi volatile rahi hai, jo dik
```