In [25]:
```
!pip install pandas numpy fuzzywuzzy python-Levenshtein
```

```
Requirement already satisfied: pandas in c:\users\praty\anaconda3\lib\site-packag
es (2.2.2)
Requirement already satisfied: numpy in c:\users\praty\anaconda3\lib\site-package
s (1.26.4)
Collecting fuzzywuzzy
  Downloading fuzzywuzzy-0.18.0-py2.py3-none-any.whl.metadata (4.9 kB)
Collecting python-Levenshtein
  Downloading python_levenshtein-0.27.1-py3-none-any.whl.metadata (3.7 kB)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\praty\anaconda3
\lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\praty\anaconda3\lib\site-
packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\praty\anaconda3\lib\sit
e-packages (from pandas) (2023.3)
Collecting Levenshtein==0.27.1 (from python-Levenshtein)
  Downloading levenshtein-0.27.1-cp312-cp312-win_amd64.whl.metadata (3.6 kB)
Collecting rapidfuzz<4.0.0,>=3.9.0 (from Levenshtein==0.27.1->python-Levenshtein)
  Downloading rapidfuzz-3.13.0-cp312-cp312-win_amd64.whl.metadata (12 kB)
Requirement already satisfied: six>=1.5 in c:\users\praty\anaconda3\lib\site-pack
ages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Downloading fuzzywuzzy-0.18.0-py2.py3-none-any.whl (18 kB)
Downloading python_levenshtein-0.27.1-py3-none-any.whl (9.4 kB)
Downloading levenshtein-0.27.1-cp312-cp312-win_amd64.whl (100 kB)
Downloading rapidfuzz-3.13.0-cp312-cp312-win_amd64.whl (1.6 MB)
   ---------------------------------------- 0.0/1.6 MB ? eta -:--:--
   ---------------------------------- ------- 1.3/1.6 MB 7.5 MB/s eta 0:00:01
   ---------------------------------------- 1.6/1.6 MB 6.7 MB/s eta 0:00:00
Installing collected packages: fuzzywuzzy, rapidfuzz, Levenshtein, python-Levensh
tein
Successfully installed Levenshtein-0.27.1 fuzzywuzzy-0.18.0 python-Levenshtein-0.
27.1 rapidfuzz-3.13.0
```

In [27]:
```
import pandas as pd
import numpy as np
from fuzzywuzzy import process
```

In [29]:
```
df = pd.read_csv("C:\\Panda notebook file\\Fin.csv")
```

In [31]:
```
df.columns = df.columns.str.strip().str.replace(" ", "_").str.replace(".", "").s
```

In [33]:
```
if 'Total_Revenue' in df.columns:
    df['Revenue_Growth'] = df.groupby('Company')['Total_Revenue'].pct_change() *

if 'Net_Income' in df.columns:
    df['Net_Income_Growth'] = df.groupby('Company')['Net_Income'].pct_change() *
```

In [35]:
```
known_metrics = {
    "revenue": "Total_Revenue",
    "net income": "Net_Income",
    "revenue growth": "Revenue_Growth",
    "net income growth": "Net_Income_Growth"
}

def extract_metric(query):
    best_match, score = process.extractOne(query, known_metrics.keys())
```

```
    return known_metrics[best_match] if score > 70 else None

def extract_company(query):
    companies = df['Company'].dropna().unique()
    best_match, score = process.extractOne(query, companies)
    return best_match if score > 70 else None
```

In [37]:

```
def smart_chatbot(query):
    query = query.lower()
    company = extract_company(query)
    metric = extract_metric(query)

    if not company:
        return "I couldn't recognize the company you're referring to."

    if "total" in query:
        if metric:
            try:
                total = df[df['Company'].str.lower() == company.lower()][metric]
                return f"Total {metric.replace('_', ' ')} for {company} is ${tot
            except:
                return f"Couldn't fetch total for {metric}."

    if "average" in query or "mean" in query:
        if metric:
            try:
                avg = df[df['Company'].str.lower() == company.lower()][metric].m
                return f"Average {metric.replace('_', ' ')} for {company} is {av
            except:
                return f"Couldn't fetch average for {metric}."

    if "min" in query or "lowest" in query:
        if metric:
            try:
                value = df[df['Company'].str.lower() == company.lower()][metric]
                return f"Minimum {metric.replace('_', ' ')} for {company} is {va
            except:
                return f"Couldn't fetch min for {metric}."

    if "max" in query or "highest" in query:
        if metric:
            try:
                value = df[df['Company'].str.lower() == company.lower()][metric]
                return f"Maximum {metric.replace('_', ' ')} for {company} is {va
            except:
                return f"Couldn't fetch max for {metric}."

    if "trend" in query or "growth over time" in query or "yearly" in query:
        if metric:
            try:
                data = df[df['Company'].str.lower() == company.lower()][['Year',
                return f"{metric.replace('_', ' ')} over time for {company}:\n"
            except:
                return f"Couldn't fetch trend for {metric}."

    return "Sorry, I didn't understand that. Try asking about total, average, mi
```

In [39]:

```
while True:
    user_input = input("You: ")
```

```
    if user_input.lower() in ['exit', 'quit']:
        print("Chatbot: Goodbye!")
        break
    print("Chatbot:", smart_chatbot(user_input))
```

```
Chatbot: Average Total Revenue for Microsoft is 218435.67%
Chatbot: Total Revenue over time for Tesla:
 Year  Total_Revenue
 2022          81462
 2023          96773
 2024          97690
Chatbot: Sorry, I didn't understand that. Try asking about total, average, min/ma
x, or trend.
Chatbot: Total Total Revenue for Apple is $1,168,648
Chatbot: Sorry, I didn't understand that. Try asking about total, average, min/ma
x, or trend.
Chatbot: Minimum Net Income for Tesla is 7,153
Chatbot: Sorry, I didn't understand that. Try asking about total, average, min/ma
x, or trend.
Chatbot: Maximum Total Revenue for Tesla is 97,690
Chatbot: I couldn't recognize the company you're referring to.
Chatbot: Goodbye!
```

In [55]:
```
while True:
    user_input = input("You: ")
    if user_input.lower() in ['exit', 'quit']:
        print("Chatbot: Goodbye!")
        break
    print("Chatbot:", smart_chatbot(user_input))
```

```
Chatbot: Maximum Total Revenue for Tesla is 97,690
Chatbot: I couldn't recognize the company you're referring to.
Chatbot: Goodbye!
```

In [63]:
```
def get_min_revenue_among_companies(companies, column="Total_Revenue"):
    companies = [c.strip().lower() for c in companies]
    df['Company_clean'] = df['Company'].str.strip().str.lower()
    filtered = df[df['Company_clean'].isin(companies)]

    if filtered.empty:
        return "No data found for the specified companies."

    # Group by company and sum or average revenue
    grouped = filtered.groupby('Company_clean')[column].sum().reset_index()
    min_row = grouped.loc[grouped[column].idxmin()]

    return f"{min_row['Company_clean'].title()} has the minimum total revenue: {
```

In [65]:
```
def chatbot_response(user_input):
    user_input = user_input.lower()

    if "revenue growth" in user_input:
        company = user_input.split("of")[-1].strip()
        return calculate_growth(company, "Total_Revenue")

    elif "net income growth" in user_input or "profit growth" in user_input:
        company = user_input.split("of")[-1].strip()
        return calculate_growth(company, "Net_Income")
```

```
    elif "net income" in user_input or "profit" in user_input:
        company = user_input.split("of")[-1].strip()
        return get_trend(company, "Net_Income")

    elif "min revenue company" in user_input:
        companies = ['tesla', 'apple', 'microsoft']
        return get_min_revenue_among_companies(companies)

    else:
        return "Sorry, I didn't understand that. Try rephrasing or ask about tot
```

In [67]:
```
while True:
    user_input = input("You: ")
    if user_input.lower() in ['exit', 'quit']:
        print("Chatbot: Goodbye!")
        break
    print("Chatbot:", smart_chatbot(user_input))
```

```
Chatbot: I couldn't recognize the company you're referring to.
Chatbot: Total Revenue Growth for Microsoft is $23
Chatbot: Goodbye!
```

In [69]:
```
known_companies = ['apple', 'microsoft', 'tesla', 'google', 'amazon', 'meta']
```

In [71]:
```
known_companies = ['apple', 'microsoft', 'tesla', 'google', 'amazon', 'meta']
```

In [73]:
```
def chatbot_response(user_input):
    user_input = user_input.lower()
    companies_in_input = extract_companies(user_input, known_companies)

    if "revenue growth" in user_input:
        company = user_input.split("of")[-1].strip()
        return calculate_growth(company, "Total_Revenue")

    elif "net income growth" in user_input or "profit growth" in user_input:
        company = user_input.split("of")[-1].strip()
        return calculate_growth(company, "Net_Income")

    elif "net income" in user_input or "profit" in user_input:
        company = user_input.split("of")[-1].strip()
        return get_trend(company, "Net_Income")

    elif "min revenue" in user_input and companies_in_input:
        return get_min_revenue_among_companies(companies_in_input)

    elif "max revenue" in user_input and companies_in_input:
        return get_max_revenue_among_companies(companies_in_input)

    else:
        return "Sorry, I didn't understand that. Try rephrasing or ask about tot
```

In [75]:
```
def get_max_revenue_among_companies(companies, column="Total_Revenue"):
    companies = [c.strip().lower() for c in companies]
    df['Company_clean'] = df['Company'].str.strip().str.lower()
    filtered = df[df['Company_clean'].isin(companies)]

    if filtered.empty:
        return "No data found for the specified companies."
```

```
    grouped = filtered.groupby('Company_clean')[column].sum().reset_index()
    max_row = grouped.loc[grouped[column].idxmax()]

    return f"{max_row['Company_clean'].title()} has the maximum total revenue: {
```

In [77]:
```
while True:
    user_input = input("You: ")
    if user_input.lower() in ['exit', 'quit']:
        print("Chatbot: Goodbye!")
        break
    print("Chatbot:", smart_chatbot(user_input))
```

```
Chatbot: Minimum Revenue Growth for Microsoft is 7
Chatbot: I couldn't recognize the company you're referring to.
Chatbot: Goodbye!
```

In [ ]: