import pandas as pd

import matplotlib.pyplot as plt

from wordcloud import WordCloud

import warnings

# Suppress warnings from openpyxl

warnings.filterwarnings("ignore", category=UserWarning, module="openpyxl")

# --- File Paths ---

connections\_file = "/Users/pconnor/Desktop/Custom GPT Files/LinkedIn Data Analyzer/Connections.csv"

content\_file = "/Users/pconnor/Desktop/Custom GPT Files/LinkedIn Data Analyzer/Content\_2024-08-25\_2024-11-22\_C. PeteConnor MS, CCCM (1).xlsx"

messages\_file = "/Users/pconnor/Desktop/Custom GPT Files/LinkedIn Data Analyzer/messages.csv"

# --- CONNECTIONS DATA ANALYSIS ---

try:

connections\_data = pd.read\_csv(connections\_file, skiprows=2)

connections\_data.columns = connections\_data.columns.str.strip()

# Verify required columns

required\_columns = {'Position', 'Company', 'Connected On'}

if not required\_columns.issubset(connections\_data.columns):

print(f"Error: Missing columns {required\_columns - set(connections\_data.columns)}")

else:

# Process connections data

connections\_data['Connected On'] = pd.to\_datetime(connections\_data['Connected On'], errors='coerce')

connections\_data['Year'] = connections\_data['Connected On'].dt.year

# Yearly Growth Visualization (Bar)

yearly\_trends = connections\_data.groupby('Year').size()

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

yearly\_trends.plot(kind='bar', color='skyblue', alpha=0.8, edgecolor='black')

plt.title('Yearly Growth in Connections', fontsize=16)

plt.xlabel('Year', fontsize=12)

plt.ylabel('Number of Connections', fontsize=12)

plt.tight\_layout()

plt.show()

# Word Cloud: Positions

wordcloud\_positions = WordCloud(width=800, height=400, background\_color='white').generate(" ".join(connections\_data['Position'].dropna()))

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

plt.imshow(wordcloud\_positions, interpolation='bilinear')

plt.axis('off')

plt.title('Word Cloud: Positions in Connections', fontsize=16)

plt.tight\_layout()

plt.show()

# Underrepresented Companies (Horizontal Bar)

underrepresented\_companies = connections\_data['Company'].value\_counts()

underrepresented\_companies = underrepresented\_companies[underrepresented\_companies == 1].head(10)

plt.figure(figsize=(10, 6))

underrepresented\_companies.plot(kind='barh', color='orange', alpha=0.8, edgecolor='black')

plt.title('Underrepresented Companies (Least Represented)', fontsize=16)

plt.xlabel('Number of Connections', fontsize=12)

plt.ylabel('Company', fontsize=12)

plt.tight\_layout()

plt.show()

except Exception as e:

print(f"Error processing Connections data: {e}")

# --- CONTENT PERFORMANCE ANALYSIS ---

try:

content\_data = pd.read\_excel(content\_file)

# Verify required columns

required\_columns = {'Impressions', 'Engagement', 'Content Type'}

if not required\_columns.issubset(content\_data.columns):

print(f"Error: Missing columns {required\_columns - set(content\_data.columns)}")

else:

# Calculate engagement rates

content\_data['Engagement Rate'] = content\_data['Engagement'] / content\_data['Impressions'] \* 100

avg\_engagement\_rate = content\_data.groupby('Content Type')['Engagement Rate'].mean().sort\_values(ascending=False)

# Bar Chart: Engagement Rate by Content Type

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

avg\_engagement\_rate.plot(kind='bar', color='green', alpha=0.8, edgecolor='black')

plt.title('Average Engagement Rate by Content Type', fontsize=16)

plt.xlabel('Content Type', fontsize=12)

plt.ylabel('Engagement Rate (%)', fontsize=12)

plt.tight\_layout()

plt.show()

# Pie Chart: Content Distribution

content\_distribution = content\_data['Content Type'].value\_counts()

plt.figure(figsize=(10, 6))

plt.pie(content\_distribution, labels=content\_distribution.index, autopct='%1.1f%%', startangle=140, colors=plt.cm.tab10.colors)

plt.title('Content Distribution by Type', fontsize=16)

plt.tight\_layout()

plt.show()

except Exception as e:

print(f"Error processing Content Performance data: {e}")

# --- MESSAGING DATA ANALYSIS ---

try:

messages\_data = pd.read\_csv(messages\_file)

messages\_data['DATE'] = pd.to\_datetime(messages\_data['DATE'], errors='coerce')

# Messaging Trends

self\_name = "C. Pete Connor MS, CCCM"

messages\_data['Direction'] = messages\_data['TO'].apply(lambda x: 'Outbound' if x == self\_name else 'Inbound')

messages\_data['Year-Month'] = messages\_data['DATE'].dt.to\_period('M')

message\_trends = messages\_data.groupby(['Year-Month', 'Direction']).size().unstack(fill\_value=0)

message\_trends.plot(kind='line', figsize=(12, 6), marker='o')

plt.title('Messaging Trends (Inbound vs Outbound)', fontsize=16)

plt.xlabel('Year-Month', fontsize=12)

plt.ylabel('Number of Messages', fontsize=12)

plt.legend()

plt.tight\_layout()

plt.show()

# Top Keywords from Subjects

stopwords = {"the", "to", "and", "a", "of", "is", "in", "for", "on", "with", "your", "you", "it", "at", "this", "an", "be"}

keywords\_subjects = pd.Series(" ".join(messages\_data['SUBJECT'].fillna("")).lower().split()).value\_counts().head(10)

keywords\_subjects = keywords\_subjects[~keywords\_subjects.index.isin(stopwords)]

plt.figure(figsize=(10, 6))

keywords\_subjects.plot(kind='bar', color='purple', alpha=0.7)

plt.title('Top Keywords in Message Subjects', fontsize=16)

plt.xlabel('Keywords', fontsize=12)

plt.ylabel('Frequency', fontsize=12)

plt.tight\_layout()

plt.show()

except Exception as e:

print(f"Error processing Messaging data: {e}")

import matplotlib.pyplot as plt

import numpy as np

# Sample Data

categories = ['Motivational', 'Educational', 'Promotional', 'Technical Insights', 'Networking']

post\_counts = [10, 15, 5, 25, 20] # Number of posts per category

likes = [100, 150, 50, 250, 200] # Total likes per category

comments = [30, 45, 10, 75, 50] # Total comments per category

reposts = [10, 20, 5, 30, 25] # Total reposts per category

# Total and Average Engagement Calculations

total\_engagement = [likes[i] + comments[i] + reposts[i] for i in range(len(categories))]

average\_engagement = [total\_engagement[i] / post\_counts[i] for i in range(len(categories))]

# Pie Chart: Feed Content Distribution

plt.figure(figsize=(10, 7))

category\_percentages = [(count / sum(post\_counts)) \* 100 for count in post\_counts]

plt.pie(category\_percentages, labels=categories, autopct='%1.1f%%', startangle=140, colors=['#66c2a5', '#fc8d62', '#8da0cb', '#e78ac3', '#a6d854'])

plt.title('Feed Content Distribution by Category', fontsize=16)

plt.show()

# Bar Chart: Volume vs Engagement Efficiency

plt.figure(figsize=(12, 7))

x = np.arange(len(categories))

width = 0.35

plt.bar(x - width/2, post\_counts, width, label='Volume (Post Count)', color='blue', alpha=0.7)

plt.bar(x + width/2, average\_engagement, width, label='Engagement Efficiency (Per Post)', color='orange', alpha=0.7)

plt.title('Volume vs Engagement Efficiency by Category', fontsize=16)

plt.xlabel('Categories', fontsize=12)

plt.ylabel('Metrics', fontsize=12)

plt.xticks(x, categories, fontsize=10)

plt.legend()

plt.tight\_layout()

plt.show()

import matplotlib.pyplot as plt

# Simulated data for LinkedIn Analytics

data = {

"Discovery": {

"Dates": ["2024-10-30", "2024-11-01", "2024-11-03", "2024-11-05"],

"Impressions": [1000, 1500, 1200, 1800],

"Unique Views": [300, 450, 400, 500]

},

"Engagement": {

"Dates": ["2024-10-30", "2024-11-01", "2024-11-03", "2024-11-05"],

"Likes": [50, 75, 60, 90],

"Comments": [20, 30, 25, 35],

"Shares": [10, 15, 10, 20]

},

"Followers": {

"Dates": ["2024-10-30", "2024-11-01", "2024-11-03", "2024-11-05"],

"New Followers": [10, 20, 15, 25]

},

"Top Posts": {

"URLs": ["post1", "post2", "post3", "post4"],

"Published Date": ["2024-10-30", "2024-11-01", "2024-11-03", "2024-11-05"],

"Engagements": [150, 200, 180, 250],

"Impressions": [1200, 1600, 1400, 2000]

},

"Demographics": {

"Location": ["USA", "Canada", "UK", "Germany"],

"Values": [50, 30, 15, 5]

}

}

plt.plot(data["Followers"]["Dates"], data["Followers"]["New Followers"], label="New Followers", marker="o")

# Followers Growth Visualization

plt.figure(figsize=(10, 6))

plt.plot(data["Followers"]["Dates"], data["Followers"]["New Followers"], label="New Followers", marker="o")

plt.title("Followers Growth Over Time", fontsize=16)

plt.xlabel("Dates", fontsize=12)

plt.ylabel("Count", fontsize=12)

plt.legend()

plt.tight\_layout()

plt.show()

# Audience Demographics Visualization

plt.figure(figsize=(10, 6))

plt.pie(data["Demographics"]["Values"], labels=data["Demographics"]["Location"], autopct='%1.1f%%', startangle=140, colors=['#66c2a5', '#fc8d62', '#8da0cb', '#e78ac3'])

plt.title("Audience Demographics by Location", fontsize=16)

plt.tight\_layout()

plt.show()

# File path for content performance data

content\_file = "/Users/pconnor/Desktop/Custom GPT Files/LinkedIn Data Analyzer/Content\_2024-08-25\_2024-11-22\_C. PeteConnor MS, CCCM (1).xlsx"

# Load the dataset

content\_data = pd.read\_excel(content\_file)

# Rename columns for clarity

content\_data.columns = ['Category', 'Value']

# Display basic stats

print("\nDataset Summary:")

print(content\_data)

# Create a bar chart to represent the categories and their respective values

plt.figure(figsize=(10, 6))

plt.bar(content\_data['Category'], content\_data['Value'], color='skyblue', edgecolor='black', alpha=0.8)

plt.title('Performance Metrics Overview', fontsize=16)

plt.xlabel('Category', fontsize=12)

plt.ylabel('Value', fontsize=12)

plt.xticks(rotation=30, ha='right', fontsize=10)

plt.tight\_layout()

plt.show()

# Recommendations based on available data

print("\nRecommendations:")

print("- Focus on increasing metrics like 'Impressions' and 'Unique Views' by experimenting with new content strategies.")

print("- Track progress over future timeframes to identify emerging trends or drops in performance.")

import matplotlib.dates as mdates

import matplotlib.pyplot as plt

import pandas as pd

# --- Ensure 'Date' Column Exists ---

if 'Date' not in content\_data.columns:

print("\nAdding synthetic 'Date' column for visualization...")

content\_data['Date'] = pd.date\_range(start='2024-01-01', periods=len(content\_data), freq='M')

# --- Ensure 'Value' Column is Numeric ---

content\_data['Value'] = pd.to\_numeric(content\_data['Value'], errors='coerce')

# --- Correct Time-Series Visualization ---

try:

plt.figure(figsize=(10, 6))

ax = plt.gca() # Get current axis

content\_data.plot(

x='Date',

y='Value',

kind='scatter',

color='orange',

title='Value Trends Over Time',

xlabel='Month',

ylabel='Value',

ax=ax

)

# Dynamically adjust tick spacing based on dataset range

if len(content\_data['Date'].unique()) <= 12: # Monthly or fewer ticks

ax.xaxis.set\_major\_locator(mdates.MonthLocator()) # Show all months

elif len(content\_data['Date'].unique()) > 365: # Daily large range

ax.xaxis.set\_major\_locator(mdates.YearLocator()) # Reduce to yearly ticks

else:

ax.xaxis.set\_major\_locator(mdates.WeekdayLocator(interval=4)) # Weekly ticks

ax.xaxis.set\_major\_formatter(mdates.DateFormatter('%Y-%m-%d')) # Format dates

plt.xticks(rotation=45) # Rotate x-axis labels for better readability

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight\_layout()

plt.show()

except Exception as e:

print(f"Error during time-series visualization: {e}")

# --- Simulate Adjusted Data (if necessary) ---

adjustment\_percentage = 20

try:

simulated\_data = simulate\_metrics(content\_data, 'Value', adjustment\_percentage)

print(f"\nSimulated data with {adjustment\_percentage}% increase in Value:")

print(simulated\_data[['Category', 'Value']])

except Exception as e:

print(f"Error during simulation: {e}")

# --- Recommendations ---

try:

recommendations = generate\_recommendations(content\_data)

print("\nRecommendations based on analysis:")

for rec in recommendations:

print(f"- {rec}")

except Exception as e:

print(f"Error generating recommendations: {e}")

# --- Enhanced Category Analysis ---

try:

# Calculate total, average, and maximum values per category

category\_summary = content\_data.groupby('Category')['Value'].agg(

Total='sum',

Average='mean',

Max='max'

).sort\_values(by='Total', ascending=False)

print("\nCategory Summary:")

print(category\_summary)

# Visualize total values by category

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

category\_summary['Total'].plot(kind='bar', color='skyblue', edgecolor='black', alpha=0.8)

plt.title('Total Value by Category', fontsize=16)

plt.xlabel('Category', fontsize=12)

plt.ylabel('Total Value', fontsize=12)

plt.xticks(rotation=30, ha='right', fontsize=10)

plt.tight\_layout()

plt.show()

# Highlight top-performing categories

top\_category = category\_summary['Total'].idxmax()

print(f"\nTop-performing category: {top\_category}")

print(f"Details:\n{category\_summary.loc[top\_category]}")

except Exception as e:

print(f"Error during enhanced category analysis: {e}")

# --- Export Data for Future Analysis ---

try:

# Save simulated data

simulated\_data.to\_csv("Simulated\_Data.csv", index=False)

print("\nSimulated data exported to 'Simulated\_Data.csv'.")

# Save category summary

category\_summary.to\_csv("Category\_Summary.csv")

print("Category summary exported to 'Category\_Summary.csv'.")

except Exception as e:

print(f"Error exporting data: {e}")

# --- Final Recommendations ---

try:

print("\nRecommendations based on category performance:")

for category, row in category\_summary.iterrows():

if row['Total'] > category\_summary['Total'].mean():

print(f"- Focus on increasing engagement for high-performing category: {category} (Total: {row['Total']:.2f})")

else:

print(f"- Explore strategies to improve performance for: {category} (Total: {row['Total']:.2f})")

except Exception as e:

print(f"Error generating final recommendations: {e}")