import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.metrics import pairwise\_distances

import numpy as np

from matplotlib import cm

# Load file

df = pd.read\_excel("Thematic\_Analysis.xlsx", sheet\_name=0)

# Use consistent plot style

plt.style.use('default')

sns.set\_style("whitegrid")

# SECTION 1 - Frequency Descriptives ---------------------------------------------------------------------------------------------------------------

print("\n---------------------- SECTION 1 ----------------------------")

# 1 - Theme Frequency

theme\_counts = df["theme\_construct"].value\_counts().reset\_index()

theme\_counts.columns = ["theme\_construct", "frequency"]

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

ax = sns.barplot(data=theme\_counts, y="theme\_construct", x="frequency", color='#003f5c')

for i, (freq, theme) in enumerate(zip(theme\_counts["frequency"], theme\_counts["theme\_construct"])):

    ax.text(freq + 0.5, i, str(freq), va='center', fontsize=9)

plt.title("Theme Frequency")

plt.xlabel("Frequency")

plt.ylabel("Theme (Construct)")

plt.tight\_layout()

plt.show()

# 2 - Code Frequency – Top 15

grouped\_code\_counts = df["grouped\_code"].value\_counts().nlargest(15).reset\_index()

grouped\_code\_counts.columns = ["grouped\_code", "frequency"]

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

ax = sns.barplot(data=grouped\_code\_counts, y="grouped\_code", x="frequency", color='#003f5c')

for i, (freq, code) in enumerate(zip(grouped\_code\_counts["frequency"], grouped\_code\_counts["grouped\_code"])):

    ax.text(freq + 0.5, i, str(freq), va='center', fontsize=9)

plt.title("Code Frequency - Top 15")

plt.xlabel("Frequency")

plt.ylabel("Code")

plt.tight\_layout()

plt.show()

# 3 - Theme Frequency by Interviewee Type

theme\_by\_group = df.groupby(['theme\_construct', 'interviewee\_type']).size().reset\_index(name='frequency')

theme\_pivot = theme\_by\_group.pivot(index='theme\_construct', columns='interviewee\_type', values='frequency').fillna(0)

theme\_pivot['total'] = theme\_pivot.sum(axis=1)

theme\_pivot = theme\_pivot.sort\_values(by='total', ascending=False).drop(columns='total')

theme\_pivot.plot(kind='bar', stacked=False, figsize=(14, 8), color=['#003f5c', '#7a9eb1'])

plt.title("Theme Frequency by Interviewee Group")

plt.xlabel("Theme (Construct)")

plt.ylabel("Frequency")

plt.xticks(rotation=45, ha='right')

plt.legend(title='Interviewee Type')

plt.tight\_layout()

plt.show()

print("Theme Frequency by Interviewee Group")

display(theme\_pivot)

# 4 - Quote Density Map

quote\_counts = df.groupby("interviewee\_id")["quote\_id"].nunique().reset\_index()

quote\_counts = quote\_counts.sort\_values("quote\_id", ascending=False)

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

sns.barplot(data=quote\_counts, x="interviewee\_id", y="quote\_id", color='#003f5c')

plt.title("Quote Density per Interviewee")

plt.xlabel("Interviewee ID")

plt.ylabel("Number of Quotes")

plt.xticks(rotation=45, ha="right")

plt.tight\_layout()

plt.show()

# SECTION 2 - Grouped Comparison: Auditors vs. Innovation Team ---------------------------------------------------------------------------------------------------------------

# Answers Research Question RQ2b

print("\n---------------------- SECTION 2 ----------------------------")

# 0 - Prep the Data

# Group and count theme occurrences

#binary\_theme = df.groupby(["theme\_construct", "interviewee\_type", "seniority\_binary"]).size().reset\_index(name="frequency")

#binary\_theme["Group\_Label"] = binary\_theme["interviewee\_type"] + ", " + binary\_theme["seniority\_binary"]

# Pivot to wide format

#pivot = binary\_theme.pivot(index="theme\_construct", columns="Group\_Label", values="frequency").fillna(0)

#pivot["Total"] = pivot.sum(axis=1)

#pivot = pivot.sort\_values("Total", ascending=False).drop(columns="Total")

# Count unique interviewees per group

interviewee\_counts = df.groupby(["interviewee\_type", "seniority\_binary"])["interviewee\_id"].nunique().reset\_index(name="n\_interviewees")

# Group theme frequencies

binary\_theme = df.groupby(["theme\_construct", "interviewee\_type", "seniority\_binary"]).size().reset\_index(name="frequency")

# Merge and normalize

binary\_theme = binary\_theme.merge(interviewee\_counts, on=["interviewee\_type", "seniority\_binary"])

binary\_theme["normalized\_frequency"] = binary\_theme["frequency"] / binary\_theme["n\_interviewees"]

# Label and pivot

binary\_theme["Group\_Label"] = binary\_theme["interviewee\_type"] + ", " + binary\_theme["seniority\_binary"]

pivot = binary\_theme.pivot(index="theme\_construct", columns="Group\_Label", values="normalized\_frequency").fillna(0)

pivot["Total"] = pivot.sum(axis=1)

pivot = pivot.sort\_values("Total", ascending=False).drop(columns="Total")

# Define groups

auditor\_groups = ["Auditor, Junior", "Auditor, Senior"]

innovator\_groups = ["Innovator, Junior", "Innovator, Senior"]

custom\_colors = ['#002f4b', '#3d5d74', '#5f7d94', '#7f9eb1']

blues\_palette = sns.color\_palette("Blues", n\_colors=5)[1:]

auditor\_data = pivot[auditor\_groups]

innovator\_data = pivot[innovator\_groups]

# Bar setup

themes = pivot.index.tolist()

x = np.arange(len(themes))

width = 0.35

fig, ax = plt.subplots(figsize=(14, 8))

auditor\_bottom = np.zeros(len(themes))

innovator\_bottom = np.zeros(len(themes))

# 1 - Theme Frequency by Interviewee Group × Seniority (Grouped Stacked Bars)

for idx, group in enumerate(auditor\_groups):

    values = auditor\_data[group]

    ax.bar(x - width/2, values, width, label=group, bottom=auditor\_bottom, color=custom\_colors[idx])

    for i, val in enumerate(values):

        if val > 0:

            ax.text(x[i] - width/2, auditor\_bottom[i] + val / 2, f'{int(val)}', ha='center', va='center', color='white', fontsize=9)

    auditor\_bottom += values

for idx, group in enumerate(innovator\_groups):

    values = innovator\_data[group]

    ax.bar(x + width/2, values, width, label=group, bottom=innovator\_bottom, color=custom\_colors[idx + 2])

    for i, val in enumerate(values):

        if val > 0:

            ax.text(x[i] + width/2, innovator\_bottom[i] + val / 2, f'{int(val)}', ha='center', va='center', color='white', fontsize=9)

    innovator\_bottom += values

ax.set\_title("Theme Frequency by Interviewee Group × Seniority (Grouped Stacked Bars)")

ax.set\_xlabel("Theme (Construct)")

ax.set\_ylabel("Frequency")

ax.set\_xticks(x)

ax.set\_xticklabels(themes, rotation=45, ha='right')

ax.legend(title="Group & Seniority", bbox\_to\_anchor=(1.02, 1), loc='upper left')

plt.tight\_layout()

plt.show()

# 2 - Theme Frequency Tables

theme\_group\_ct = pd.crosstab([df["theme\_construct"]], [df["interviewee\_type"], df["seniority\_binary"]])

theme\_group\_pct = theme\_group\_ct.div(theme\_group\_ct.sum(axis=1), axis=0).round(2)

print("Frequency Table")

display(theme\_group\_ct.sort\_index())

print("\nPercentage Table")

display(theme\_group\_pct.sort\_index())

# 3 - Code Frequency by Group – Top 15

code\_counts\_binary = df.groupby(["grouped\_code", "interviewee\_type"]).size().reset\_index(name="frequency")

pivot\_code = code\_counts\_binary.pivot\_table(index="grouped\_code", columns="interviewee\_type", values="frequency", fill\_value=0)

top\_15\_codes = pivot\_code.sum(axis=1).sort\_values(ascending=False).head(15).index

pivot\_code\_top15 = pivot\_code.loc[top\_15\_codes]

plt.figure(figsize=(12, 8))

sns.heatmap(pivot\_code\_top15, annot=True, fmt=".0f", cmap="Blues", linewidths=0.5, annot\_kws={"size": 10})

plt.title("Code Frequency: Auditor vs Innovation Team – Top 15")

plt.xlabel("Interviewee Group")

plt.ylabel("Code")

plt.tight\_layout()

plt.show()

# 4 - Normalized Code Frequency by Group - Top 15

group\_counts = df.groupby("interviewee\_type")["interviewee\_id"].nunique()

normalized\_code = pivot\_code.div(group\_counts, axis=1)

top\_15\_codes\_norm = normalized\_code.mean(axis=1).sort\_values(ascending=False).head(15).index

pivot\_code\_top15\_norm = normalized\_code.loc[top\_15\_codes\_norm]

plt.figure(figsize=(12, 8))

sns.heatmap(pivot\_code\_top15\_norm, annot=True, fmt=".2f", cmap="Blues", linewidths=0.5, annot\_kws={"size": 10})

plt.title("Code Frequency: Auditor vs Innovation Team – Top 15 (Normalized)")

plt.xlabel("Interviewee Group")

plt.ylabel("Code")

plt.tight\_layout()

plt.show()

# SECTION 3 - Seniority-Based Comparison (Auditors Only) ---------------------------------------------------------------------------------------------------------------

# It helps explain moderation of seniority

print("\n---------------------- SECTION 3 ----------------------------")

# 0 - Prep the Data

auditor\_df = df[df["interviewee\_type"] == "Auditor"]

seniority\_table = pd.crosstab(auditor\_df["grouped\_code"], auditor\_df["seniority\_binary"])

seniority\_table["Total"] = seniority\_table.sum(axis=1)

seniority\_table = seniority\_table.sort\_values("Total", ascending=False).drop(columns="Total")

# Normalization

auditor\_counts = auditor\_df.groupby("seniority\_binary")["interviewee\_id"].nunique()

normalized\_table = seniority\_table.div(auditor\_counts, axis=1)

# Top 15 by total frequency & normalized

top\_codes\_raw = seniority\_table.sum(axis=1).nlargest(15).index

top\_codes\_norm = normalized\_table.mean(axis=1).nlargest(15).index

# 1 - Raw Code Freq (Top 15)

plt.figure(figsize=(12, 10))

sns.heatmap(seniority\_table.loc[top\_codes\_raw], annot=True, fmt='g', cmap='Blues', linewidths=0.5)

plt.title("Code Frequency by Auditor Seniority - Top 15")

plt.xlabel("Seniority Level")

plt.ylabel("Code")

plt.tight\_layout()

plt.show()

# 2 - Normalized Freq (Top 15)

plt.figure(figsize=(12, 8))

sns.heatmap(normalized\_table.loc[top\_codes\_norm], annot=True, fmt=".2f", cmap='Blues', linewidths=0.5)

plt.title("Code Frequency by Auditor Seniority (Normalized) - Top 15")

plt.xlabel("Seniority Level")

plt.ylabel("Code")

plt.tight\_layout()

plt.show()

# SECTION 4 - Co-Occurrence Analyses ---------------------------------------------------------------------------------------------------------------

# It helps to answer Research Question RQ2b

print("\n---------------------- SECTION 4 ----------------------------")

# 1 - Theme Co-occurrence by Quote

df\_theme\_unique = df[['quote\_id', 'theme\_construct']].drop\_duplicates()

theme\_matrix = df\_theme\_unique.groupby(['quote\_id', 'theme\_construct']).size().unstack(fill\_value=0)

theme\_matrix[theme\_matrix > 0] = 1

theme\_cooccurrence = theme\_matrix.T.dot(theme\_matrix)

np.fill\_diagonal(theme\_cooccurrence.values, 0)

plt.figure(figsize=(10, 8))

sns.heatmap(theme\_cooccurrence, cmap='Blues', annot=True, fmt='d')

plt.title('Theme Co-Occurrence Matrix')

plt.xlabel('Theme')

plt.ylabel('Theme')

plt.xticks(rotation=45, ha='right')

plt.tight\_layout()

plt.show()

# 2 - Code Co-occurrence

grouped\_matrix = df.groupby(["quote\_id", "grouped\_code"]).size().unstack(fill\_value=0)

grouped\_matrix[grouped\_matrix > 0] = 1

co\_occurrence = np.dot(grouped\_matrix.T, grouped\_matrix)

co\_occ\_df = pd.DataFrame(co\_occurrence, index=grouped\_matrix.columns, columns=grouped\_matrix.columns)

np.fill\_diagonal(co\_occ\_df.values, 0)

top\_codes = df["grouped\_code"].value\_counts().nlargest(15).index

filtered\_matrix = co\_occ\_df.loc[top\_codes, top\_codes]

plt.figure(figsize=(12, 10))

sns.heatmap(filtered\_matrix, annot=True, fmt='g', cmap='Blues', linewidths=0.5)

plt.title("Code Co-Occurrence Matrix - Top 15")

plt.xlabel("Code")

plt.ylabel("Code")

plt.xticks(rotation=45, ha='right')

plt.yticks(rotation=0)

plt.tight\_layout()

plt.show()

# 3 - Theme Frequency by Interviewee Matrix

theme\_matrix = pd.crosstab(df["interviewee\_id"], df["theme\_construct"])

plt.figure(figsize=(10, 8))

sns.heatmap(theme\_matrix, annot=True, cmap="Blues", fmt="d", linewidths=0.5)

plt.title("Theme Frequency by Interviewee")

plt.xlabel("Theme Construct")

plt.ylabel("Interviewee ID")

plt.xticks(rotation=45, ha="right")

plt.yticks(rotation=0)

plt.tight\_layout()

plt.show()

# SECTION 5 - Advanced ---------------------------------------------------------------------------------------------------------------

print("\n---------------------- SECTION 5 ----------------------------")

#do valence tagging afterwards