import pandas as pd

import plotly.graph\_objects as go

# Load the dataset (Assuming a CSV format or DataFrame is already available)

# Replace 'data.csv' with your actual file path

df = pd.read\_csv("data.csv")

# Inspect the data

print("Data Overview:")

print(df.head())

# Step 1: Filter relevant data

print("\nStep 1: Filtering relevant data...")

# Start with journeys having `page\_number = 1`

starting\_pages = df[df['page\_number'] == 1]

print("Starting pages:")

print(starting\_pages['page'].value\_counts())

# Filter journeys with specific `journey\_name` (focus on "mobile springboard", etc.)

relevant\_journeys = df[df['journey\_name'].str.contains('check allowances|mobile springboard', na=False, case=False)]

print("\nFiltered Relevant Journeys:")

print(relevant\_journeys['journey\_name'].value\_counts())

# Step 2: Analyze common page transitions

print("\nStep 2: Analyzing common page transitions...")

# Create a transition dataset (current page -> next page)

df['next\_page'] = df['next\_page'].fillna('Exit') # Handle missing next pages

transitions = df.groupby(['page', 'next\_page']).size().reset\_index(name='count')

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

print("Most common transitions:")

print(transitions.head())

# Step 3: Visualizing the transitions using Sankey Diagram

print("\nStep 3: Visualizing transitions using Sankey Diagram...")

source = transitions['page']

target = transitions['next\_page']

value = transitions['count']

# Get unique nodes and map them to indices

unique\_nodes = list(pd.concat([source, target]).unique())

node\_indices = {node: i for i, node in enumerate(unique\_nodes)}

source\_indices = source.map(node\_indices)

target\_indices = target.map(node\_indices)

# Create the Sankey diagram

fig = go.Figure(data=[go.Sankey(

node=dict(

pad=15,

thickness=20,

line=dict(color="black", width=0.5),

label=unique\_nodes

),

link=dict(

source=source\_indices,

target=target\_indices,

value=value

)

)])

fig.update\_layout(title\_text="Page Transition Sankey Diagram", font\_size=10)

fig.show()

# Step 4: Identify the Happy Path

print("\nStep 4: Identifying the Happy Path...")

# Find the most common path starting from 'Springboard'

start\_page = "Springboard"

happy\_path = [start\_page]

while True:

next\_step = transitions[transitions['page'] == happy\_path[-1]]

if next\_step.empty:

break

most\_common\_next = next\_step.loc[next\_step['count'].idxmax()]

if most\_common\_next['next\_page'] == 'Exit':

break

happy\_path.append(most\_common\_next['next\_page'])

print("Happy Path:", " → ".join(happy\_path))

# Step 5: Visualize the Happy Path using a Sankey Diagram

print("\nStep 5: Visualizing Happy Path using Sankey Diagram...")

happy\_source = happy\_path[:-1]

happy\_target = happy\_path[1:]

happy\_value = [1] \* len(happy\_source)

# Map the Happy Path nodes to indices

happy\_unique\_nodes = list(set(happy\_source + happy\_target))

happy\_node\_indices = {node: i for i, node in enumerate(happy\_unique\_nodes)}

happy\_source\_indices = [happy\_node\_indices[s] for s in happy\_source]

happy\_target\_indices = [happy\_node\_indices[t] for t in happy\_target]

fig = go.Figure(data=[go.Sankey(

node=dict(

pad=15,

thickness=20,

line=dict(color="black", width=0.5),

label=happy\_unique\_nodes

),

link=dict(

source=happy\_source\_indices,

target=happy\_target\_indices,

value=happy\_value

)

)])

fig.update\_layout(title\_text="Happy Path Sankey Diagram", font\_size=10)

fig.show()

# Step 6: Save Results

print("\nStep 6: Saving results to CSV...")

transitions.to\_csv("page\_transitions.csv", index=False)

# Step 1: Pivot Data for Heatmap heatmap\_data = df.pivot(index='journey\_name', columns='path', values='count') heatmap\_data.fillna(0, inplace=True) # Replace NaNs with 0 for missing counts # Step 2: Plot Heatmap plt.figure(figsize=(12, 8)) sns.heatmap(heatmap\_data, annot=True, fmt=".0f", cmap="YlGnBu", cbar\_kws={'label': 'Frequency'}) plt.title("Heatmap of Paths for Each Journey") plt.xlabel("Paths") plt.ylabel("Journeys") plt.xticks(rotation=45, ha="right") plt.tight\_layout() plt.show()

import pandas as pd

import matplotlib.pyplot as plt

# Example DataFrame

data = {

'page': ['Page A', 'Page B', 'Page C', 'Page D', 'Page E'],

'next\_page': ['Page B', 'Page C', 'Page D', 'Page E', None],

'is\_exit': [0, 0, 1, 1, 0]

}

df = pd.DataFrame(data)

# Step 1: Determine the exit page

df['exit\_page'] = df.apply(

lambda row: row['next\_page'] if pd.notnull(row['next\_page']) and row['is\_exit'] == 1 else row['page'],

axis=1

)

# Step 2: Count the number of exits per page

exit\_counts = df['exit\_page'].value\_counts().reset\_index()

exit\_counts.columns = ['page', 'exit\_count']

# Step 3: Funnel Visualization

plt.figure(figsize=(8, 6))

# Plot each bar with decreasing width to create a funnel effect

for i, row in exit\_counts.iterrows():

plt.barh(row['page'], row['exit\_count'], color='lightblue', edgecolor='black', height=0.8 - i \* 0.1)

# Add labels and title

plt.xlabel("Number of Exits")

plt.ylabel("Pages")

plt.title("Funnel Analysis: Exit Pages")

plt.gca().invert\_yaxis() # Reverse the order of pages for funnel-like appearance

plt.tight\_layout()

# Show plot

plt.show()