import pandas as pd

import networkx as nx

import matplotlib.pyplot as plt

# Load the dataset (replace 'file.csv' with the actual file path)

df = pd.read\_csv('file.csv')

# Convert `hit\_date\_time` to datetime

df['hit\_date\_time'] = pd.to\_datetime(df['hit\_date\_time'])

# Sort data by journey and time

df = df.sort\_values(by=['journey\_name', 'hit\_date\_time'])

# 1. Identify Common Paths

common\_paths = df.groupby(['journey\_name', 'page', 'next\_page']).size().reset\_index(name='count')

print("Most Common Paths:")

print(common\_paths.sort\_values(by='count', ascending=False).head())

# 2. Time Spent on Pages

time\_spent = df.groupby('page')['time\_spent\_seconds'].mean().reset\_index()

print("\nAverage Time Spent on Pages:")

print(time\_spent)

# 3. Exit Pages

exit\_pages = df[df['next\_page'] == 'Exit']['page'].value\_counts().reset\_index()

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

print("\nTop Exit Pages:")

print(exit\_pages)

# 4. Visualize Journey Paths

# Example: Visualize a specific journey (e.g., "make a payment")

journey\_to\_plot = "make a payment"

journey\_df = df[df['journey\_name'] == journey\_to\_plot]

# Create a directed graph

G = nx.DiGraph()

for \_, row in journey\_df.iterrows():

G.add\_edge(row['page'], row['next\_page'])

# Plot the graph

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

pos = nx.spring\_layout(G, seed=42)

nx.draw\_networkx\_nodes(G, pos, node\_size=700, node\_color='lightblue')

nx.draw\_networkx\_edges(G, pos, width=2, edge\_color='blue', arrowsize=20)

nx.draw\_networkx\_labels(G, pos, font\_size=10, font\_color='black')

plt.title(f"Journey Path Visualization: {journey\_to\_plot}")

plt.tight\_layout()

plt.show()

import pandas as pd

# Example DataFrame

data = {

'channel\_visit\_id': [1, 1, 1, 2, 2],

'hit\_date\_time': ['2024-10-08 10:00:00', '2024-10-08 10:05:00', '2024-10-08 10:10:00', '2024-10-08 11:00:00', '2024-10-08 11:05:00'],

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

'next\_page': ['Page B', 'Page C', 'Exit', 'Page E', 'Exit']

}

df = pd.DataFrame(data)

# Convert `hit\_date\_time` to datetime for proper sorting

df['hit\_date\_time'] = pd.to\_datetime(df['hit\_date\_time'])

# Sort by `channel\_visit\_id` and `hit\_date\_time`

df\_sorted = df.sort\_values(['channel\_visit\_id', 'hit\_date\_time'])

# Group by `channel\_visit\_id` to extract ordered paths

paths = df\_sorted.groupby('channel\_visit\_id').apply(

lambda x: " → ".join(x['page'].tolist() + [x['next\_page'].iloc[-1]])

).reset\_index(name='path')

# Display resulting paths

print("Paths Taken by Users:")

print(paths)

import pandas as pd

import networkx as nx

import matplotlib.pyplot as plt

# Example DataFrame (Replace with actual data)

data = {

'channel\_visit\_id': [1, 1, 1, 2, 2, 3],

'hit\_date\_time': [

'2024-10-08 10:00:00', '2024-10-08 10:05:00', '2024-10-08 10:10:00',

'2024-10-08 11:00:00', '2024-10-08 11:05:00', '2024-10-08 12:00:00'

],

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

'next\_page': ['Page B', 'Page C', 'Exit', 'Page E', 'Exit', 'Exit'],

'time\_spent\_seconds': [5, 10, 15, 20, 25, 30]

}

df = pd.DataFrame(data)

# Convert `hit\_date\_time` to datetime

df['hit\_date\_time'] = pd.to\_datetime(df['hit\_date\_time'])

# Sort data by `channel\_visit\_id` and `hit\_date\_time`

df\_sorted = df.sort\_values(['channel\_visit\_id', 'hit\_date\_time'])

# --- 1. Identify Common Paths ---

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

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

print("\nMost Common Paths:")

print(common\_paths)

# --- 2. Drop-off Analysis ---

exit\_pages = df\_sorted[df\_sorted['next\_page'] == 'Exit']['page'].value\_counts().reset\_index()

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

print("\nDrop-off Analysis:")

print(exit\_pages)

# --- 3. Time Spent Analysis ---

time\_spent = df\_sorted.groupby('page')['time\_spent\_seconds'].mean().reset\_index()

time\_spent = time\_spent.sort\_values(by='time\_spent\_seconds', ascending=False)

print("\nAverage Time Spent on Pages:")

print(time\_spent)

# --- 4. Journey Length ---

journey\_length = df\_sorted.groupby('channel\_visit\_id').size().reset\_index(name='num\_pages')

print("\nJourney Lengths (Number of Pages):")

print(journey\_length)

# --- 5. Transition Frequencies ---

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

print("\nTransition Frequencies:")

print(transitions)

# --- 6. Visualizations ---

# Sankey Diagram

import plotly.graph\_objects as go

# Prepare Sankey Data

source = transitions['page'].tolist()

target = transitions['next\_page'].tolist()

value = transitions['transition\_count'].tolist()

# Create unique mapping for nodes

unique\_nodes = list(set(source + target))

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

# Map source and target to node indices

source\_indices = [node\_mapping[node] for node in source]

target\_indices = [node\_mapping[node] for node in target]

# Plot 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="Sankey Diagram of Customer Journeys", font\_size=10)

fig.show()

# Directed Graph

G = nx.DiGraph()

for \_, row in transitions.iterrows():

G.add\_edge(row['page'], row['next\_page'], weight=row['transition\_count'])

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

pos = nx.spring\_layout(G, seed=42)

nx.draw\_networkx\_nodes(G, pos, node\_size=700, node\_color='lightblue')

nx.draw\_networkx\_edges(G, pos, width=2, edge\_color='blue', arrowsize=20)

nx.draw\_networkx\_labels(G, pos, font\_size=10, font\_color='black')

plt.title("Customer Journey Transitions")

plt.tight\_layout()

plt.show()

# Bar Chart for Time Spent

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

plt.bar(time\_spent['page'], time\_spent['time\_spent\_seconds'], color='lightblue', edgecolor='black')

plt.xlabel("Page")

plt.ylabel("Average Time Spent (seconds)")

plt.title("Average Time Spent on Pages")

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

import pandas as pd

import networkx as nx

import matplotlib.pyplot as plt

# Example DataFrame (Replace with actual data)

data = {

'user\_id': [101, 101, 101, 102, 102, 101],

'channel\_visit\_id': [1, 1, 2, 3, 3, 2],

'hit\_date\_time': [

'2024-10-08 10:00:00', '2024-10-08 10:05:00', '2024-10-08 10:10:00',

'2024-10-08 11:00:00', '2024-10-08 11:05:00', '2024-10-08 10:15:00'

],

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

'next\_page': ['Page B', 'Page C', 'Exit', 'Page E', 'Exit', 'Exit']

}

df = pd.DataFrame(data)

# Convert `hit\_date\_time` to datetime

df['hit\_date\_time'] = pd.to\_datetime(df['hit\_date\_time'])

# Filter data for a specific user

user\_to\_visualize = 101

user\_data = df[df['user\_id'] == user\_to\_visualize]

# Sort by `hit\_date\_time` across all `channel\_visit\_id`s

user\_data\_sorted = user\_data.sort\_values(['hit\_date\_time'])

# Create journey paths for visualization

journey\_edges = list(zip(user\_data\_sorted['page'], user\_data\_sorted['next\_page']))

# Remove any transitions where `next\_page` is NaN

journey\_edges = [edge for edge in journey\_edges if pd.notnull(edge[1])]

# Create a directed graph

G = nx.DiGraph()

G.add\_edges\_from(journey\_edges)

# Visualize the journey graph

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

pos = nx.spring\_layout(G, seed=42)

nx.draw\_networkx\_nodes(G, pos, node\_size=700, node\_color='lightblue')

nx.draw\_networkx\_edges(G, pos, width=2, edge\_color='blue', arrowsize=20)

nx.draw\_networkx\_labels(G, pos, font\_size=10, font\_color='black')

plt.title(f"Journey Visualization for User ID: {user\_to\_visualize}")

plt.tight\_layout()

plt.show()

# Print the full ordered journey

full\_journey = " → ".join(user\_data\_sorted['page'].tolist() + [user\_data\_sorted['next\_page'].iloc[-1]])

print(f"Full Journey for User {user\_to\_visualize}:")

print(full\_journey)

import pandas as pd

import networkx as nx

import matplotlib.pyplot as plt

# Example DataFrame

data = {

'user\_id': [101, 101, 101, 102, 102, 101],

'channel\_visit\_id': [1, 1, 2, 3, 3, 2],

'hit\_date\_time': [

'2024-10-08 10:00:00', '2024-10-08 10:05:00', '2024-10-08 10:10:00',

'2024-10-08 11:00:00', '2024-10-08 11:05:00', '2024-10-08 10:15:00'

],

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

'next\_page': ['Page B', 'Page C', 'Exit', 'Page E', 'Exit', 'Exit'],

'time\_spent\_seconds': [300, 600, 900, 1200, 600, 300] # Seconds

}

df = pd.DataFrame(data)

# Convert `hit\_date\_time` to datetime

df['hit\_date\_time'] = pd.to\_datetime(df['hit\_date\_time'])

# Calculate exit time

df['exit\_time'] = df['hit\_date\_time'] + pd.to\_timedelta(df['time\_spent\_seconds'], unit='s')

# Sort data by user, `channel\_visit\_id`, and `exit\_time`

df\_sorted = df.sort\_values(['user\_id', 'channel\_visit\_id', 'exit\_time'])

# For each `user\_id`, construct the journey path

journeys = df\_sorted.groupby('user\_id').apply(

lambda x: list(zip(x['page'], x['next\_page']))

).reset\_index(name='journey')

# Display journeys

print("Customer Journeys with Time Consideration:")

print(journeys)

# Visualize a journey for a specific `user\_id`

user\_to\_visualize = 101

journey\_edges = journeys[journeys['user\_id'] == user\_to\_visualize]['journey'].iloc[0]

# Create a directed graph

G = nx.DiGraph()

G.add\_edges\_from(journey\_edges)

# Plot the graph

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

pos = nx.spring\_layout(G, seed=42)

nx.draw\_networkx\_nodes(G, pos, node\_size=700, node\_color='lightblue')

nx.draw\_networkx\_edges(G, pos, width=2, edge\_color='blue', arrowsize=20)

nx.draw\_networkx\_labels(G, pos, font\_size=10, font\_color='black')

plt.title(f"Journey Visualization for User ID: {user\_to\_visualize}")

plt.tight\_layout()

plt.show()

# Print the ordered journey

ordered\_journey = " → ".join(df\_sorted[df\_sorted['user\_id'] == user\_to\_visualize]['page'].tolist())

print(f"Ordered Journey for User {user\_to\_visualize}:")

print(ordered\_journey)

import pandas as pd

import networkx as nx

import matplotlib.pyplot as plt

# Example DataFrame (Replace with actual data)

data = {

'user\_id': [101, 101, 101, 102, 102, 101],

'channel\_visit\_id': [1, 1, 2, 3, 3, 2],

'hit\_date\_time': [

'2024-10-08 10:00:00', '2024-10-08 10:05:00', '2024-10-08 10:10:00',

'2024-10-08 11:00:00', '2024-10-08 11:05:00', '2024-10-08 10:15:00'

],

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

'next\_page': ['Page B', 'Page C', 'Exit', 'Page E', 'Exit', 'Exit'],

'time\_spent\_seconds': [300, 600, 900, 1200, 600, 300],

'page\_referrer': [None, 'Page A', 'Page B', None, 'Page D', 'Page B']

}

df = pd.DataFrame(data)

# Convert `hit\_date\_time` to datetime

df['hit\_date\_time'] = pd.to\_datetime(df['hit\_date\_time'])

# Calculate exit time

df['exit\_time'] = df['hit\_date\_time'] + pd.to\_timedelta(df['time\_spent\_seconds'], unit='s')

# Sort data by user, `channel\_visit\_id`, and `exit\_time`

df\_sorted = df.sort\_values(['user\_id', 'channel\_visit\_id', 'exit\_time'])

# Build the journey path by validating `page\_referrer`

def build\_valid\_journey(group):

journey = []

previous\_page = None

for \_, row in group.iterrows():

# Validate the transition using `page\_referrer`

if previous\_page is None or row['page\_referrer'] == previous\_page:

journey.append((row['page'], row['next\_page']))

previous\_page = row['page']

return journey

# Apply the logic for each user

journeys = df\_sorted.groupby('user\_id').apply(build\_valid\_journey).reset\_index(name='journey')

# Display journeys

print("Customer Journeys with Time and Referrer Validation:")

print(journeys)

# Visualize a journey for a specific `user\_id`

user\_to\_visualize = 101

journey\_edges = journeys[journeys['user\_id'] == user\_to\_visualize]['journey'].iloc[0]

# Create a directed graph

G = nx.DiGraph()

G.add\_edges\_from(journey\_edges)

# Plot the graph

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

pos = nx.spring\_layout(G, seed=42)

nx.draw\_networkx\_nodes(G, pos, node\_size=700, node\_color='lightblue')

nx.draw\_networkx\_edges(G, pos, width=2, edge\_color='blue', arrowsize=20)

nx.draw\_networkx\_labels(G, pos, font\_size=10, font\_color='black')

plt.title(f"Journey Visualization for User ID: {user\_to\_visualize}")

plt.tight\_layout()

plt.show()

# Print the ordered journey

ordered\_journey = " → ".join([edge[0] for edge in journey\_edges] + [journey\_edges[-1][1]])

print(f"Ordered Journey for User {user\_to\_visualize}:")

print(ordered\_journey)