import pandas as pd

import networkx as nx

import matplotlib.pyplot as plt

from collections import Counter

# 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 Network Graph

print("\nStep 3: Visualizing transitions...")

G = nx.DiGraph()

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

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

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

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

edges = G.edges(data=True)

weights = [edge[2]['weight'] for edge in edges]

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

nx.draw\_networkx\_edges(G, pos, width=weights, alpha=0.6, edge\_color='gray')

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

plt.title("Page Transition Network")

plt.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: Additional Analysis

print("\nStep 5: Additional Analyses...")

# (a) Time Spent Analysis

time\_spent = df.groupby('page')['time\_spent\_seconds'].mean().sort\_values(ascending=False)

print("\nAverage time spent on pages:")

print(time\_spent.head())

# (b) Exit Analysis

exit\_pages = df[df['is\_exit'] == 1]['page'].value\_counts()

print("\nTop Exit Pages:")

print(exit\_pages.head())

# (c) Drop-off Rate Analysis

df['drop\_off'] = (df['next\_page'] == 'Exit').astype(int)

drop\_off\_rate = df.groupby('page')['drop\_off'].mean().sort\_values(ascending=False)

print("\nDrop-off rates by page:")

print(drop\_off\_rate.head())

# Step 6: Visualize Happy Path Transitions

print("\nStep 6: Visualizing Happy Path...")

path\_graph = nx.DiGraph()

for i in range(len(happy\_path) - 1):

path\_graph.add\_edge(happy\_path[i], happy\_path[i+1])

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

path\_pos = nx.spring\_layout(path\_graph, seed=42)

nx.draw\_networkx\_nodes(path\_graph, path\_pos, node\_size=700, node\_color='lightgreen')

nx.draw\_networkx\_edges(path\_graph, path\_pos, width=2, edge\_color='blue')

nx.draw\_networkx\_labels(path\_graph, path\_pos, font\_size=10, font\_color='black')

plt.title("Happy Path Visualization")

plt.show()

# Step 7: Save Results

print("\nStep 7: Saving results to CSV...")

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

time\_spent.to\_csv("time\_spent\_analysis.csv", header=['average\_time\_spent'])

drop\_off\_rate.to\_csv("drop\_off\_rate.csv", header=['drop\_off\_rate'])

import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
  
# Load your dataset (replace 'data.csv' with your actual file path)  
data = pd.read\_csv('data.csv')  
  
# Basic Dataset Overview  
print("Dataset Shape:", data.shape)  
print("Dataset Info:")  
data.info()  
print("\nSummary Statistics:")  
print(data.describe(include='all'))  
  
# Checking for Missing Values  
print("\nMissing Values:")  
missing\_values = data.isnull().sum()  
print(missing\_values[missing\_values > 0])  
  
# Checking for Duplicates  
duplicates = data.duplicated().sum()  
print(f"\nNumber of duplicate rows: {duplicates}")  
  
# Data Type Distribution  
print("\nData Types Distribution:")  
print(data.dtypes.value\_counts())  
  
# Checking Column-Wise Null Percentage  
print("\nNull Percentage by Column:")  
null\_percentage = (data.isnull().sum() / len(data)) \* 100  
print(null\_percentage[null\_percentage > 0])  
  
# Advanced Univariate Analysis  
for column in data.columns:  
 if data[column].dtype == 'object':  
 print(f"\nValue Counts for {column}:")  
 print(data[column].value\_counts())  
 elif np.issubdtype(data[column].dtype, np.number):  
 print(f"\nStatistics for {column}:")  
 print(data[column].describe())  
  
# Visualizing Numeric Data  
numeric\_columns = data.select\_dtypes(include=[np.number]).columns  
for column in numeric\_columns:  
 plt.figure()  
 sns.histplot(data[column], kde=True, bins=30)  
 plt.title(f"Distribution of {column}")  
 plt.xlabel(column)  
 plt.ylabel("Frequency")  
 plt.show()  
  
# Visualizing Categorical Data  
categorical\_columns = data.select\_dtypes(include=['object']).columns  
for column in categorical\_columns:  
 plt.figure()  
 data[column].value\_counts().head(10).plot(kind='bar', color='skyblue')  
 plt.title(f"Top 10 Categories in {column}")  
 plt.xlabel(column)  
 plt.ylabel("Frequency")  
 plt.show()  
  
# Correlation Analysis  
if len(numeric\_columns) > 1:  
 plt.figure(figsize=(12, 10))  
 correlation\_matrix = data[numeric\_columns].corr()  
 sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt=".2f")  
 plt.title("Correlation Matrix")  
 plt.show()  
  
# Outlier Detection Using Boxplots  
for column in numeric\_columns:  
 plt.figure()  
 sns.boxplot(x=data[column], color='orange')  
 plt.title(f"Boxplot of {column}")  
 plt.xlabel(column)  
 plt.show()  
  
# Checking for Unique Values per Column  
print("\nUnique Values per Column:")  
print(data.nunique())  
  
# Pairplot for Numerical Features  
if len(numeric\_columns) > 1:  
 sns.pairplot(data[numeric\_columns])  
 plt.suptitle("Pairplot of Numeric Columns", y=1.02)  
 plt.show()  
  
# Feature Engineering Suggestions  
print("\nFeature Engineering Suggestions:")  
print("1. Convert categorical columns with high cardinality to numeric using label encoding or one-hot encoding.")  
print("2. Aggregate numeric columns for group-level insights.")  
print("3. Create new features based on domain knowledge (e.g., ratios, trends).")  
  
# Hypothesis 1: Experimental Limb Bias  
if 'experiment\_limb\_id' in data.columns and 'customer\_segment' in data.columns:  
 print("\nAnalyzing Experimental Limb Bias Towards Specific Customer Segments:")  
 limb\_bias = data.groupby(['experiment\_limb\_id', 'customer\_segment']).size().unstack()  
 print(limb\_bias)  
 limb\_bias.plot(kind='bar', stacked=True, figsize=(10, 6))  
 plt.title("Bias in Experimental Limb Towards Customer Segments")  
 plt.xlabel("Experiment Limb ID")  
 plt.ylabel("Count")  
 plt.show()  
  
# Hypothesis 2: Conversion Effectiveness of Experimental Limb  
if 'experiment\_limb\_id' in data.columns and 'conversion\_flag' in data.columns:  
 print("\nAnalyzing Conversion Effectiveness of Experimental Limb:")  
 conversion\_analysis = data.groupby('experiment\_limb\_id')['conversion\_flag'].value\_counts(normalize=True).unstack()  
 print(conversion\_analysis)  
 conversion\_analysis.plot(kind='bar', stacked=True, figsize=(10, 6))  
 plt.title("Conversion Rates by Experimental Limb")  
 plt.xlabel("Experiment Limb ID")  
 plt.ylabel("Proportion")  
 plt.show()  
  
# Hypothesis 3: Revenue Impact of Conversion  
if 'conversion\_flag' in data.columns and 'service\_revenue' in data.columns:  
 print("\nAnalyzing Revenue Impact of Conversion:")  
 revenue\_impact = data.groupby('conversion\_flag')['service\_revenue'].mean()  
 print(revenue\_impact)  
 revenue\_impact.plot(kind='bar', color='green', figsize=(8, 5))  
 plt.title("Average Revenue by Conversion Flag")  
 plt.xlabel("Conversion Flag")  
 plt.ylabel("Average Revenue")  
 plt.show()  
  
# Hypothesis 4: Tenure Influence on Conversion  
if 'tenure\_months' in data.columns and 'conversion\_flag' in data.columns:  
 print("\nAnalyzing Influence of Tenure on Conversion:")  
 sns.boxplot(x='conversion\_flag', y='tenure\_months', data=data, palette='Set2')  
 plt.title("Tenure Distribution by Conversion Flag")  
 plt.xlabel("Conversion Flag")  
 plt.ylabel("Tenure (Months)")  
 plt.show()  
  
# Hypothesis 5: Region-Based Conversion Rates  
if 'region' in data.columns and 'conversion\_flag' in data.columns:  
 print("\nAnalyzing Conversion Rates Across Regions:")  
 region\_conversion = data.groupby('region')['conversion\_flag'].value\_counts(normalize=True).unstack()  
 print(region\_conversion)  
 region\_conversion.plot(kind='bar', stacked=True, figsize=(12, 6), colormap='viridis')  
 plt.title("Conversion Rates by Region")  
 plt.xlabel("Region")  
 plt.ylabel("Proportion")  
 plt.show()  
  
# Hypothesis 6: Device Type Impact on Conversion  
if 'device\_manufacturer' in data.columns and 'conversion\_flag' in data.columns:  
 print("\nAnalyzing Device Manufacturer Impact on Conversion:")  
 device\_conversion = data.groupby('device\_manufacturer')['conversion\_flag'].value\_counts(normalize=True).unstack()  
 print(device\_conversion)  
 device\_conversion.plot(kind='bar', stacked=True, figsize=(12, 6), colormap='coolwarm')  
 plt.title("Conversion Rates by Device Manufacturer")  
 plt.xlabel("Device Manufacturer")  
 plt.ylabel("Proportion")  
 plt.show()  
  
# Hypothesis 7: Impact of Promotions on Conversion  
if 'promotion\_group\_lvl1' in data.columns and 'conversion\_flag' in data.columns:  
 print("\nAnalyzing Impact of Promotions on Conversion:")  
 promo\_conversion = data.groupby('promotion\_group\_lvl1')['conversion\_flag'].value\_counts(normalize=True).unstack()  
 print(promo\_conversion)  
 promo\_conversion.plot(kind='bar', stacked=True, figsize=(12, 6), colormap='Pastel1')  
 plt.title("Conversion Rates by Promotion Level 1")  
 plt.xlabel("Promotion Group Level 1")  
 plt.ylabel("Proportion")  
 plt.show()  
  
# Marketing-Specific Insights  
# Example: Group by 'brand' to analyze customer activity by brand  
if 'brand' in data.columns:  
 print("\nMarketing Insights by Brand:")  
 brand\_summary = data.groupby('brand').agg({  
 'service\_id': 'count',  
 'last\_activity': 'nunique',  
 'account\_num': 'nunique',  
 'service\_revenue': 'sum',  
 'service\_product\_revenue\_net\_inc\_vat': 'mean',  
 'tenure\_months': 'mean'  
 }).rename(columns={  
 'service\_id': 'Service Count',  
 'last\_activity': 'Unique Activities',  
 'account\_num': 'Unique Accounts',  
 'service\_revenue': 'Total Revenue',  
 'service\_product\_revenue\_net\_inc\_vat': 'Avg Revenue (Net VAT)',  
 'tenure\_months': 'Avg Tenure (Months)'  
 })  
 print(brand\_summary)  
 brand\_summary.plot(kind='bar', figsize=(12, 6))  
 plt.title("Brand-Level Summary")  
 plt.ylabel("Values")  
 plt.xlabel("Brand")  
 plt.show()  
  
# Regional Analysis (if region/country is available)  
if 'region' in data.columns:  
 print("\nRegional Analysis:")  
 region\_summary = data.groupby('region').agg({  
 'service\_id': 'count',  
 'service\_revenue': 'sum',  
 'tenure\_months': 'mean'  
 }).rename(columns={  
 'service\_id': 'Service Count',  
 'service\_revenue': 'Total Revenue',  
 'tenure\_months': 'Avg Tenure (Months)'  
 })  
 print(region\_summary)  
 region\_summary.plot(kind='bar', figsize=(12, 6))  
 plt.title("Region-Level Summary")  
 plt.ylabel("Values")  
 plt.xlabel("Region")  
 plt.show()  
  
# Exporting Processed Data (Optional)  
processed\_data\_path = 'processed\_data.csv'  
data.to\_csv(processed\_data\_path, index=False)  
print(f"\nProcessed data saved to {processed\_data\_path}")  
  
# Insights Summary  
print("\nKey Insights:")  
print("1. Shape of the dataset indicates dimensions.")  
print("2. Missing values and duplicates should be handled appropriately.")  
print("3. Data type analysis indicates feature engineering opportunities.")  
print("4. Numerical distributions and categorical distributions highlight key trends.")  
print("5. Correlation analysis reveals relationships among numerical features.")  
print("6. Outlier detection suggests data cleaning requirements.")  
print("7. Marketing-specific insights provide actionable outcomes for business strategies.")  
print("8. Regional and brand-specific revenue and tenure analysis offer targeted insights.")  
print("9. Hypotheses on experimental limb bias, conversion effectiveness, revenue impact, tenure influence, region-based conversions, device type impact, and promotion effects offer a foundation for deeper analysis.")

### Jira Ticket for Exploratory Data Analysis (EDA)

#### ****[Task Title]:**** Perform Exploratory Data Analysis (EDA) on CNBA and TNBA Data

#### ****[Current State]:****

The CNBA (Commercial Data) and TNBA (Transactional Data) datasets have been collected but remain unexamined. Key metrics, relationships, and patterns are unknown, limiting the team's ability to design effective strategies for customer segmentation, pricing policies, and conversion optimization.

#### ****[Change Proposed]:****

1. Load and clean the CNBA and TNBA datasets.
2. Perform descriptive statistics to summarize key metrics (e.g., exposures, transactions, conversions).
3. Explore relationships between variables such as pricing tiers, request channels, and household segmentation.
4. Create visualizations (e.g., distribution plots, correlation heatmaps) to identify trends and outliers.
5. Document insights and key findings to inform subsequent tasks like model development and dashboard creation.

#### ****[Benefit/Goal]:****

The EDA will provide a comprehensive understanding of the CNBA and TNBA datasets, uncovering critical patterns and anomalies. This task will establish the foundation for deeper analyses, predictive modeling, and data-driven decision-making to improve commercial and transactional performance.

#### ****[Acceptance Criteria]:****

1. CNBA and TNBA datasets are loaded, cleaned, and validated for consistency.
2. Summary statistics (e.g., means, medians, variances) for key metrics are calculated and documented.
3. At least three visualizations are created to highlight data distributions, correlations, and patterns.
4. A report summarizing findings, trends, and recommendations is shared with stakeholders.
5. Insights generated are aligned with project objectives and downstream tasks.

Let me know if you need additional refinements for the ticket!