Consumer Insights: Investing in Securities

Introduction

This project explores the investment preferences and behaviors of a dataset comprising individuals across different demographics. The dataset includes information on gender, age, various investment avenues (such as mutual funds, equity market, bonds, fixed deposits), reasons for investment in specific avenues, monitoring frequency, and return expectations. The objective is to analyze patterns in investment choices and preferences among the surveyed population.

Project Description

The dataset consists of demographic variables like gender and age, along with categorical data detailing individuals' investment habits and preferences. Key aspects investigated include:

- Investment Avenues: Analysis of preferred investment avenues such as mutual funds, equity markets, bonds, and fixed deposits.
- Reasons for Investment: Exploration of motivations behind choosing specific investment avenues, such as equity, mutual funds, bonds, and fixed deposits.
- Investment Duration: Distribution of investment durations chosen by investors.
- Monitoring Practices: Frequency at which investors monitor their investments.
- Return Expectations: Expected returns from investments, categorized by gender and other demographic factors.

The project employs statistical methods such as one-sample t-tests to compare age demographics against hypothesized means, providing insights into age-related investment behaviors. Visualizations like pie charts and bar plots are used to effectively present findings on investment distribution, reasons for investment choices, and expectations.

Business Problem:

Optimizing Investment Strategies Based on Insights

In today's financial landscape, understanding investor preferences and behaviors is crucial for financial institutions and advisors to tailor their offerings effectively. The dataset contains information on various aspects of investment behavior among individuals, including gender, age, preferred investment avenues, reasons for investment, monitoring practices, and return expectations.

By leveraging the data-driven insights from this analysis, financial institutions can not only optimize their investment strategies but also strengthen client relationships and achieve sustainable growth in the competitive financial services industry.

Analysis

```
import pandas as pd

import numpy as np

df = pd.read_csv("Finance_data.csv")

df
```

- Importing pandas: import pandas as pd imports the pandas library, which provides data structures and data analysis tools.

- Loading the CSV file: pd.read_csv("Finance_data.csv") reads the CSV file named "Finance_data.csv" into a pandas DataFrame df.

- Displaying DataFrame: print(df.head()) prints the first five rows of the DataFrame df. This helps in understanding the structure of the dataset and the types of data it contains.

```
df.info( )
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7280 entries, 0 to 7279
Data columns (total 23 columns):
```

#	Column	Non-Null Count	Dtype
0	gender	7280 non-null	object
1	age	7280 non-null	int64
2	Investment Avenues	7280 non-null	object
3	Mutual_Funds	7280 non-null	int64
4	Equity_Market	7280 non-null	int64
5	Debentures	7280 non-null	int64
6	Government_Bonds	7280 non-null	int64
7	Fixed_Deposits	7280 non-null	int64
8	PPF	7280 non-null	int64
9	Gold	7280 non-null	int64
10	Stock_Marktet	7280 non-null	object
11	Factor	7280 non-null	object
12	Objective	7280 non-null	object
13	Purpose	7280 non-null	object
14	Duration	7280 non-null	object
15	Invest_Monitor	7280 non-null	object
16	Expect	7280 non-null	object
17	Avenue	7280 non-null	object
18	savings objectives	7280 non-null	object
19	Reason_Equity	7280 non-null	object
20	Reason_Mutual	7280 non-null	object
21	Reason_Bonds	7280 non-null	object
22	Reason_FD	7280 non-null	object
23	Source	7280 non-null	object

```
dtypes: int64(8), object(16)
```

The dataset appears to have 23 columns (features), with a mix of integer and object (likely string) data types. Here's a breakdown of what each column represents based on the names provided:

1. gender: Categorical variable indicating the gender of the respondents.
2. age: Numerical variable indicating the age of the respondents.
3. Investment_Avenues: Categorical variable indicating different avenues or types of investments.
4. Mutual_Funds, Equity_Market, Debentures, Government_Bonds, Fixed_Deposits, PPF, Gold: Numerical variables indicating the amount or level of investment in each respective avenue.
5. Stock_Market: Categorical variable related to the stock market, possibly indicating preferences or attitudes.
6. Factor, Objective, Purpose, Duration, Invest_Monitor, Expect, Avenue, What are your savings objectives: Various categorical variables likely capturing different aspects of investment behavior, goals, and preferences.
7. Reason_Equity, Reason_Mutual, Reason_Bonds, Reason_FD: Categorical variables indicating reasons or motivations behind choosing specific investment avenues.
8. Source: Categorical variable indicating the source or channel through which respondents learned about investments.

Based on the information provided, this dataset seems to be focused on understanding the investment behavior, preferences, and motivations of a group of respondents. The numerical columns (Mutual_Funds, Equity_Market, etc.) likely represent either categorical levels or ordinal data related to the amount or intensity of investment in each category.

df.isnull().sum().sum()

The expression df.isnull().sum().sum() calculates the total number of missing values in the DataFrame df. In your case, you mentioned the DataFrame has 7280 rows (observations) and several columns.

When you apply df.isnull().sum(), it gives you a Series where each column name is paired with the number of missing values in that column. .sum () on this Series then adds up these counts across all columns, giving you the total number of missing values in the entire DataFrame.

Given that your DataFrame df has no missing values (Non-Null Count for all columns is 7280), the result of df.isnull().sum().sum() should be 0. This means there are no missing values in any column of your DataFrame.

df.columns

```
Index(['gender', 'age', 'Investment_Avenues', 'Mutual_Funds', 'Equity_Market',
       'Debentures', 'Government_Bonds', 'Fixed_Deposits', 'PPF', 'Gold',
       'Stock_Marktet', 'Factor', 'Objective', 'Purpose', 'Duration',
       'Invest_Monitor', 'Expect', 'Avenue',
       'What are your savings objectives?', 'Reason_Equity', 'Reason_Mutual',
```

'Reason_Bonds', 'Reason_FD', 'Source'],
dtype='object')

The df.columns attribute in pandas returns the names of all columns in the DataFrame df. Based on the structure you described earlier, the columns of your DataFrame are as follows:

1. gender: Categorical variable indicating the gender of the respondents.
2. age: Numerical variable indicating the age of the respondents.
3. Investment_Avenues: Categorical variable indicating different avenues or types of investments.
4. Mutual_Funds: Numerical variable indicating the amount or level of investment in mutual funds.
5. Equity_Market: Numerical variable indicating the amount or level of investment in the equity market.
6. Debentures: Numerical variable indicating the amount or level of investment in debentures.
7. Government_Bonds: Numerical variable indicating the amount or level of investment in government bonds.
8. Fixed_Deposits: Numerical variable indicating the amount or level of investment in fixed deposits.
9. PPF: Numerical variable indicating the amount or level of investment in Public Provident Fund (PPF).
10. Gold: Numerical variable indicating the amount or level of investment in gold.
11. Stock_Market: Categorical variable related to the stock market, possibly indicating preferences or attitudes.
12. Factor: Categorical variable related to factors influencing investment decisions.
13. Objective: Categorical variable related to investment objectives.
14. Purpose: Categorical variable related to the purpose of investments.
15. Duration: Categorical variable indicating the duration of investments.
16. Invest_Monitor: Categorical variable related to monitoring investments.
17. Expect: Categorical variable related to expectations from investments.
18. Avenue: Categorical variable indicating preferred avenues for investments.
19. What are your savings objectives?: Categorical variable indicating savings objectives.
20. Reason_Equity: Categorical variable indicating reasons for choosing equity investments.
21. Reason_Mutual: Categorical variable indicating reasons for choosing mutual funds.
22. Reason_Bonds: Categorical variable indicating reasons for choosing bonds.
23. Reason_FD: Categorical variable indicating reasons for choosing fixed deposits.
24. Source: Categorical variable indicating the source or channel through which respondents learned about investments.

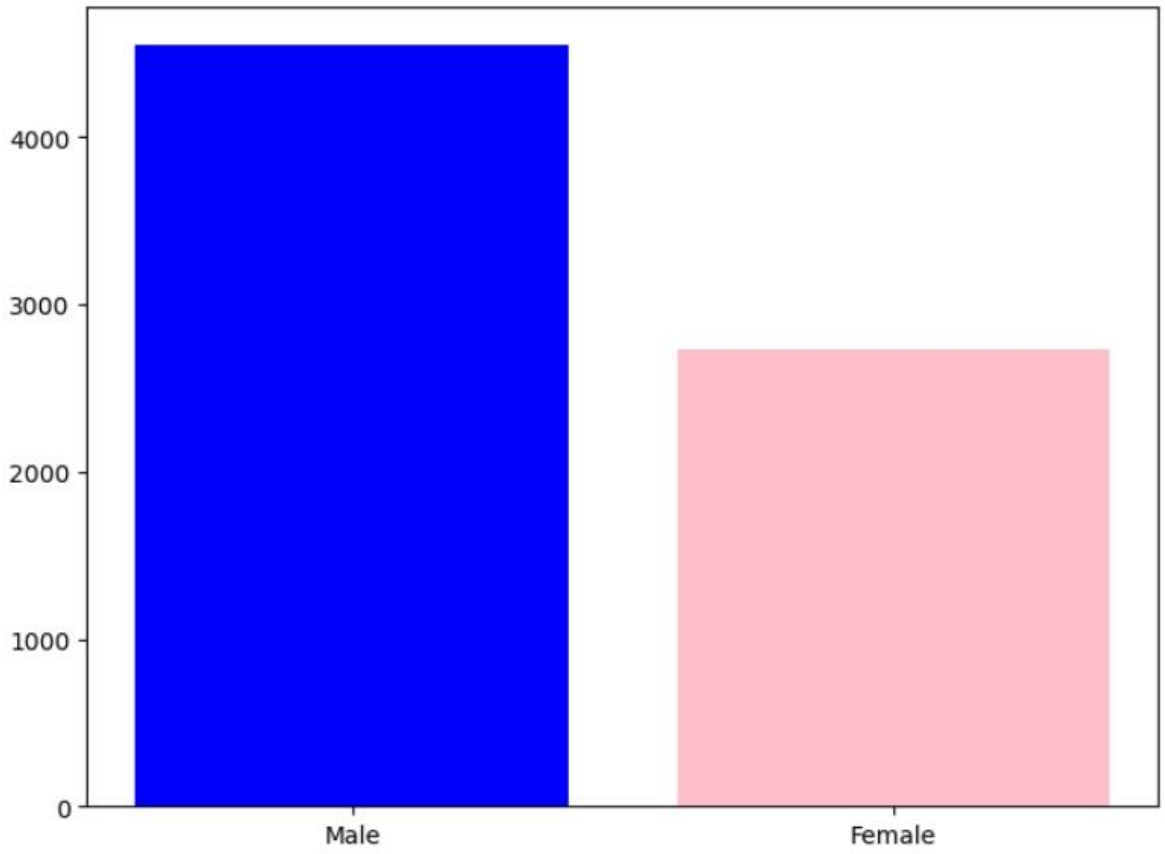
These columns collectively provide a comprehensive view of the characteristics and preferences related to investments among the respondents in the dataset.

```
df_gender = df["gender"]

colors=["blue","pink"]

counts = df_gender.value_counts()

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

plt.bar(counts.index, counts.values, width=0.8, align='center', color=colors)

plt.show()
```

Explanation:

1. Imports: First, import matplotlib.pyplot as plt for plotting functionalities.
2. Data Preparation:
    - df_gender = df["gender"]: This extracts the "gender" column from your DataFrame df assuming it contains categorical values like "Male" and "Female".
3. Define Colors:
    - colors = ["blue", "pink"]: These are chosen as colors for the bars corresponding to different genders.
4. Counting Values:
    - counts = df_gender.value_counts(): This calculates the frequency of each unique gender category in the DataFrame column.
5. Plotting:
    - plt.figure(figsize=(8, 6)): Creates a new figure with a specified size (width=8 inches, height=6 inches).
    - plt.bar(counts.index, counts.values, width=0.8, align='center', color=colors): Plots a bar chart where counts.index (unique genders) are on the x-axis and counts.values (frequency counts) are on the y-axis. width=0.8 ensures the bars are not too thin, and align='center' centers the bars on the x-axis ticks.
    -
6. Labels and Title:
    - plt.xlabel('Gender'): Sets the label for the x-axis.
    - plt.ylabel('Count'): Sets the label for the y-axis.
    - plt.title('Distribution of Gender'): Sets the title for the plot.
7. Display Plot:
    - plt.show(): Finally, displays the plot you've created.

This code snippet should generate a bar plot showing the distribution of genders in your dataset, with "Male" and "Female" (or other categories if present) represented by different colored bars. Adjust the colors or styling as per your preference or dataset characteristics.



Inference

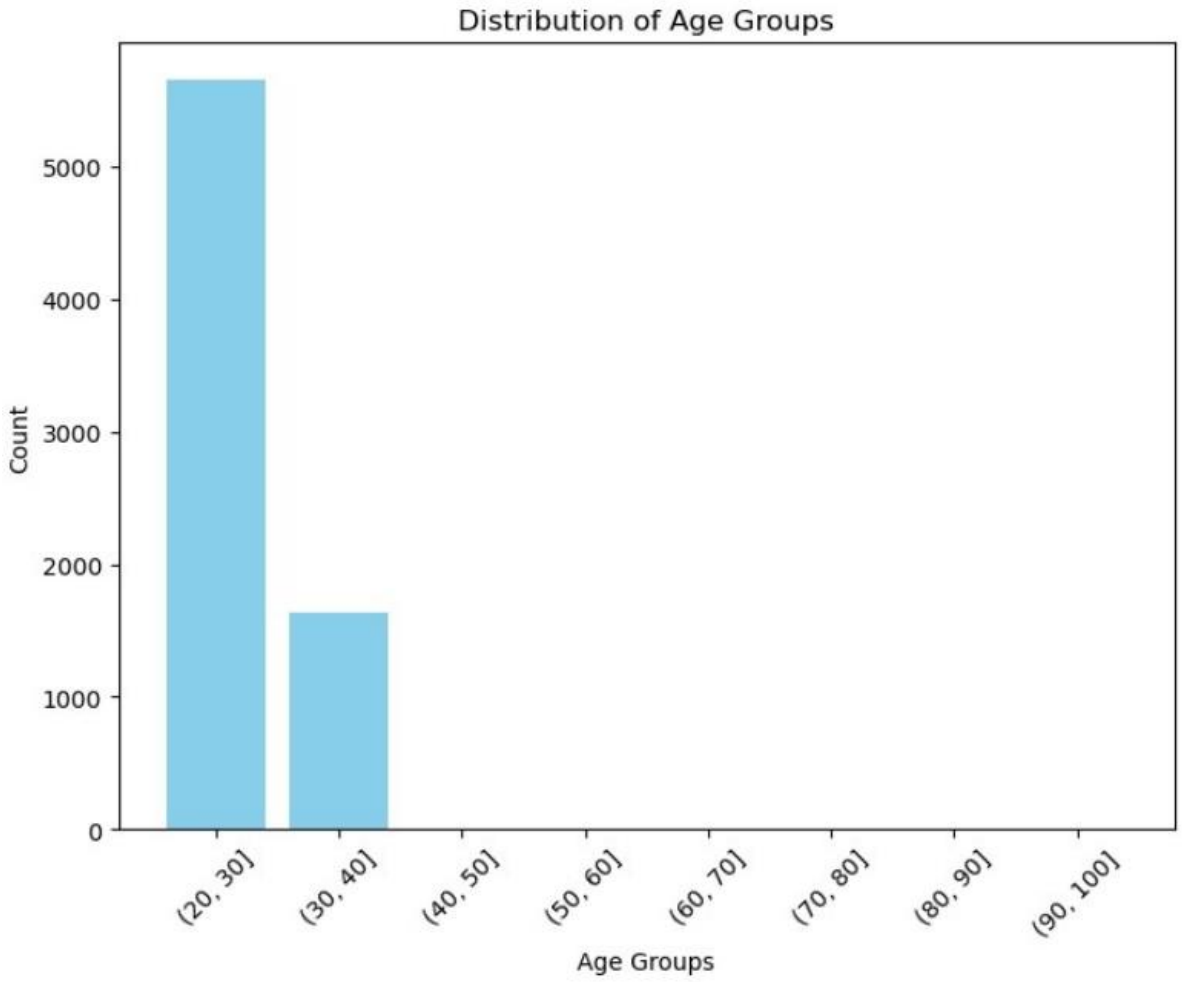
- Male count is above 4000
- Female count is between 2000-3000

```
df_age = df["age"]

age_bins = [20, 30, 40, 50, 60, 70, 80, 90, 100]

age_groups = pd.cut(df_age, bins=age_bins)

age_counts = age_groups.value_counts().sort_index()

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

plt.bar(age_counts.index.astype(str), age_counts.values, width=0.8, color='skyblue')

plt.xlabel('Age Groups')

plt.ylabel('Count')
```

```
plt.title('Distribution of Age Groups')

plt.xticks(rotation=45)

plt.show()
```

Explanation:

1. Imports:
    - import pandas as pd: Import pandas for data manipulation.
    - import matplotlib.pyplot as plt: Import pyplot module from matplotlib for plotting.
2. Data Preparation:
    - df_age = df["age"]: Extracts the "age" column from your DataFrame df.
3. Define Age Bins:
    - age_bins = [20, 30, 40, 50, 60, 70, 80, 90, 100]: Defines bins to categorize ages into groups. Adjust these bins as per your data distribution and analysis requirements.
4. Categorize Ages:
    - age_groups = pd.cut(df_age, bins=age_bins): Cuts the age data into bins defined by age_bins.
5. Count Age Groups:
    - age_counts = age_groups.value_counts().sort_index(): Counts the occurrences of each age group and sorts them based on the index (which are the age group intervals).
6. Plotting:
    - plt.figure(figsize=(8, 6)): Creates a new figure with a specified size (width=8 inches, height=6 inches).
    - plt.bar(age_counts.index.astype(str), age_counts.values, width=0.8, color='skyblue'): Plots a bar chart where age_counts.index.astype(str) (age group intervals converted to strings for better readability) are on the x-axis and age_counts.values (frequency counts) are on the y-axis. width=0.8 ensures the bars are not too thin, and color='skyblue' sets the color of the bars.
7. Labels and Title:
    - plt.xlabel('Age Groups'): Sets the label for the x-axis.
    - plt.ylabel('Count'): Sets the label for the y-axis.
    - plt.title('Distribution of Age Groups'): Sets the title for the plot.
8. Adjust x-axis Labels:
    - plt.xticks(rotation=45): Rotates the x-axis labels by 45 degrees for better readability.
9. Display Plot:
    - plt.show(): Finally, displays the plot you've created.



Inference

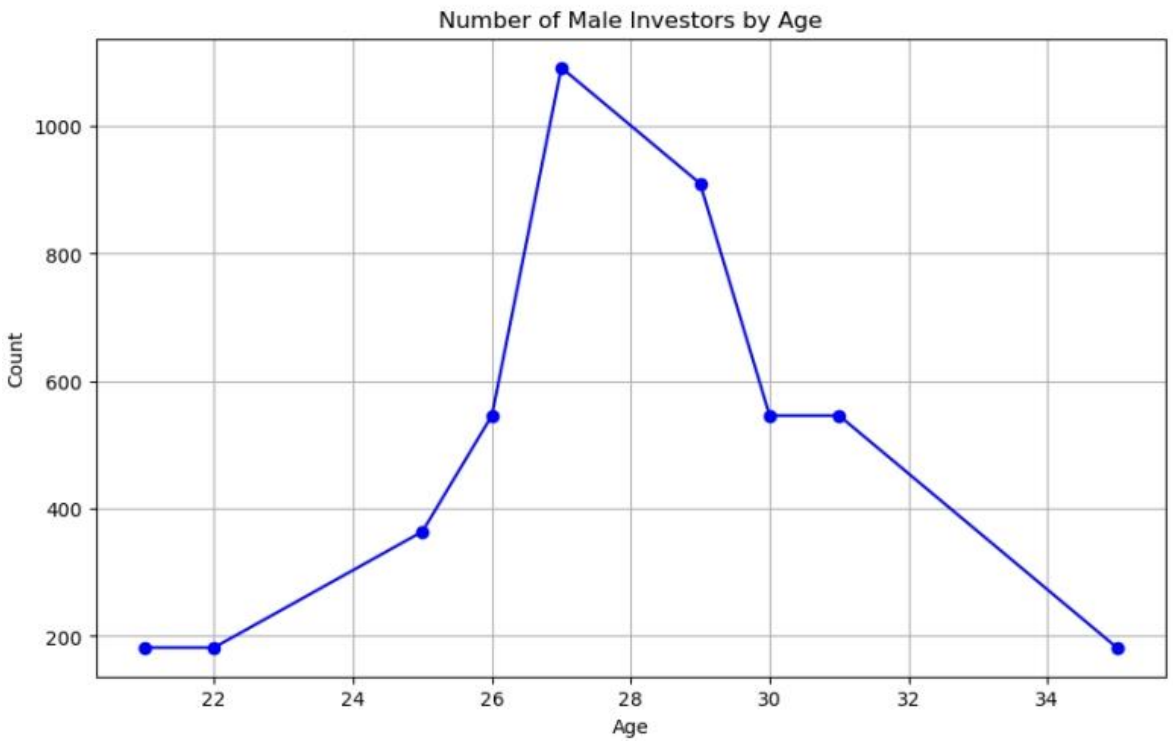
- From dataset, the majority of people investing are in the 20-30 age group.
- The count is above 5000 for the age group of 20-30.
- The count is between 1000 - 2000 for the age group of 30-40

```
# Number of male investors by Age

filtered_df = df[(df['gender'] == 'Male') & (df['Investment_Avenues'] == 'Yes')]

male_investment_age_counts = filtered_df.groupby('age').size().reset_index(name='count')

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

plt.plot(male_investment_age_counts['age'], male_investment_age_counts['count'], marker='o', linestyle='-', color='b')

plt.xlabel('Age')
```

```
plt.ylabel('Count')

plt.title('Number of Male Investors by Age')

plt.grid(True)

plt.show()
```

Explanation:

1. Data Filtering:
    - filtered_df = df[(df['gender'] == 'Male') & (df['Investment_Avenues'] == 'Yes')]: Filters the original DataFrame df to include only male investors who have indicated investment ('Yes' in 'Investment_Avenues').
2. Grouping and Counting:
    - male_investment_age_counts = filtered_df.groupby('age').size().reset_index(name='count'): Groups the filtered data by age (groupby('age')) and counts the number of occurrences (size()), then resets the index (reset_index(name='count')) to create a DataFrame with columns 'age' and 'count'.
3. Plotting:
    - plt.figure(figsize=(10, 6)): Creates a new figure with a specified size (width=10 inches, height=6 inches).
    - plt.plot(male_investment_age_counts['age'], male_investment_age_counts['count'], marker='o', linestyle='-', color='b'): Plots a line graph where x-axis represents age (male_investment_age_counts['age']) and y-axis represents the count of male investors (male_investment_age_counts['count']). marker='o' adds circular markers at data points, linestyle='-' specifies solid lines, and color='b' sets the line color to blue.
4. Labels and Title:
    - plt.xlabel('Age'): Sets the label for the x-axis.
    - plt.ylabel('Count'): Sets the label for the y-axis.
    - plt.title('Number of Male Investors by Age'): Sets the title for the plot.
5. Grid Lines:
    - plt.grid(True): Adds grid lines to the plot for better readability.
6. Display Plot:
    - plt.show(): Finally, displays the plot with all the specified settings.



Inference

- Men are more willing to invest between the ages of 26 and 30.
- The Age category of 27 is investing more when compare to other age categories

Number of Female Investors by Age

female_investment_df = df[(df['gender'] == 'Female') & (df['Investment_Avenues'] == 'Yes')]

female_investment_age_counts =
female_investment_df.groupby('age').size().reset_index(name='count')

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

plt.plot(female_investment_age_counts['age'], female_investment_age_counts['count'], marker='o', linestyle='-', color='red')

plt.xlabel('Age')

plt.ylabel('Count')

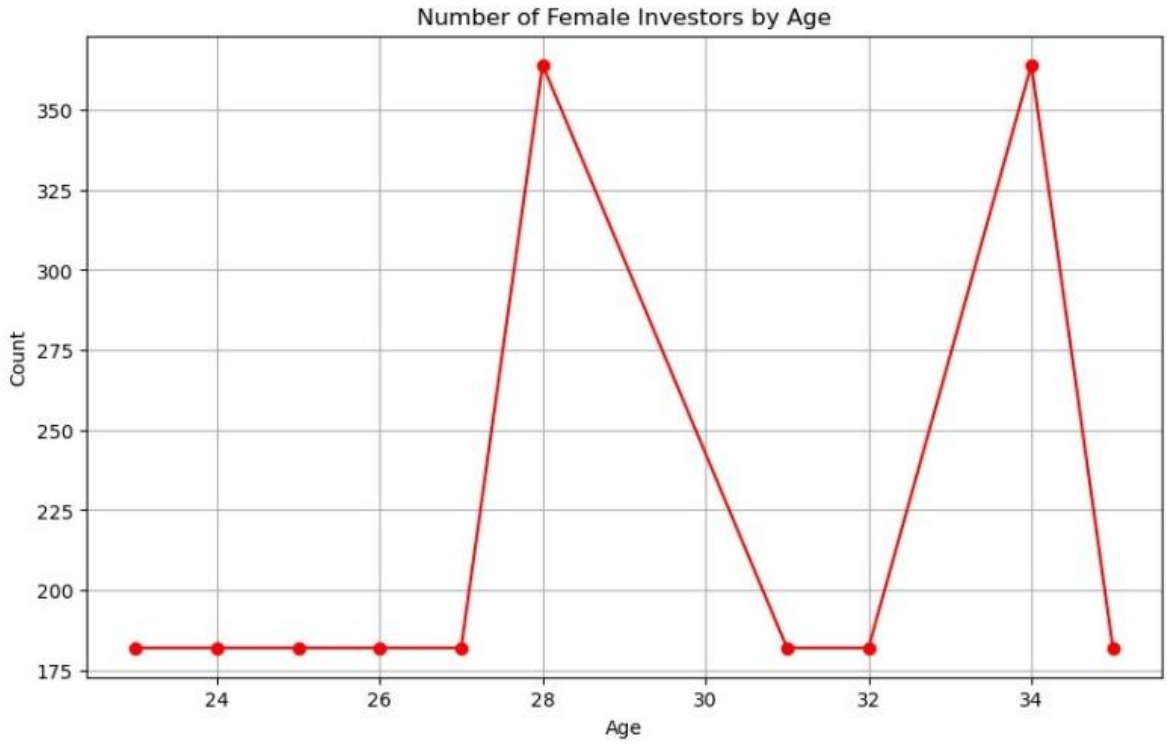
plt.title('Number of Female Investors by Age')

plt.grid(True)

plt.show()

Explanation:

1. Data Filtering:
    - female_investment_df = df[(df['gender'] == 'Female') & (df['Investment_Avenues'] == 'Yes')]: Filters the original DataFrame df to include only female investors who have indicated investment ('Yes' in 'Investment_Avenues').
2. Grouping and Counting:
    - female_investment_age_counts = female_investment_df.groupby('age').size().reset_index(name='count'): Groups the filtered data by age (groupby('age')) and counts the number of occurrences (size()), then resets the index (reset_index(name='count')) to create a DataFrame with columns 'age' and 'count'.
3. Plotting:
    - plt.figure(figsize=(10, 6)): Creates a new figure with a specified size (width=10 inches, height=6 inches).
    - plt.plot(female_investment_age_counts['age'], female_investment_age_counts['count'], marker='o', linestyle='-', color='red'): Plots a line graph where x-axis represents age (female_investment_age_counts['age']) and y-axis represents the count of female investors (female_investment_age_counts['count']). marker='o' adds circular markers at data points, linestyle='-' specifies solid lines, and color='red' sets the line color to red.
4. Labels and Title:
    - plt.xlabel('Age'): Sets the label for the x-axis.
    - plt.ylabel('Count'): Sets the label for the y-axis.
    - plt.title('Number of Female Investors by Age'): Sets the title for the plot.
5. Grid Lines:
    - plt.grid(True): Adds grid lines to the plot for better readability.
6. Display Plot:
    - plt.show(): Finally, displays the plot with all the specified settings.



Inference

• The women starts investing at the age of 23.

• The Age category of 28 and 34 is investing more when compare to other age categories

What sources do investors look at when investing

df_source = df[df["Investment_Avenues"] == "Yes"].groupby("Source").size().reset_index(name="count")

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

plt.pie(df_source["count"], labels=df_source["Source"], autopct='%1.1f%%', startangle=140, colors=["lightgreen","orange","yellow","skyblue"])

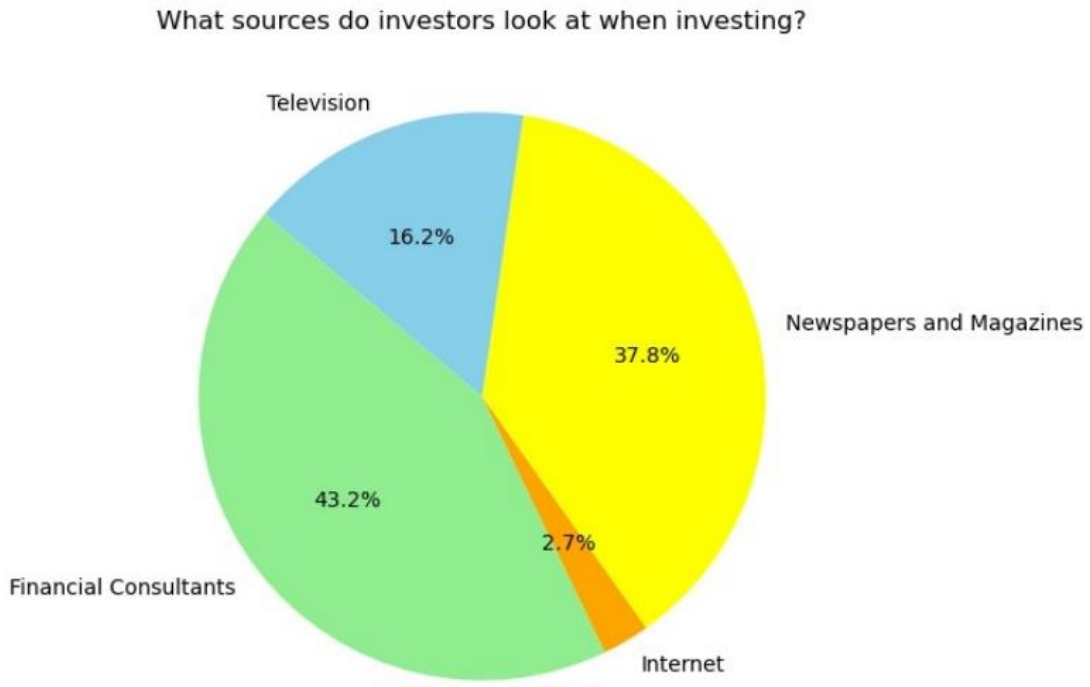
plt.title('What sources do investors look at when investing?')

plt.show()

Explanation:

1. Data Filtering:
    - investors_df = df[df["Investment_Avenues"] == "Yes"]: Filters the original DataFrame df to include only investors who have indicated investment ('Yes' in 'Investment_Avenues').
    -

2. Grouping and Counting:
    - df_source = investors_df.groupby("Source").size().reset_index(name="count"): Groups the filtered data by "Source" and counts the number of occurrences (size()), then resets the index (reset_index(name="count")) to create a DataFrame with columns 'Source' and 'count'.
3. Plotting:
    - plt.figure(figsize=(8, 6)): Creates a new figure with a specified size (width=8 inches, height=6 inches).
    - plt.pie(df_source["count"], labels=df_source["Source"], autopct='%1.1f%%', startangle=140, colors=["lightgreen", "orange", "yellow", "skyblue"]): Plots a pie chart where df_source["count"] provides the sizes of the slices, labels=df_source["Source"] provides the labels for each slice, autopct='%1.1f%%' displays the percentage on each slice, startangle=140 sets the starting angle of the pie chart, and colors specifies the colors for each slice.
4. Title:
    - plt.title('What sources do investors look at when investing?'): Sets the title for the pie chart.
5. Display Plot:
    - plt.show(): Finally, displays the pie chart with all the specified settings.



Inference

• Here we see that investors generally invest through Financial Consultants.

• Financial Consultants - 43.2%

• Television-16.2%

• Newspapers and Magazines- 37.8%

• Internet- 2.7%

Number of Investors by Investment Duration

df_duration = df.groupby("Duration").size().reset_index(name="count")

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

plt.bar(df_duration['Duration'], df_duration['count'], color=['orange',"lightgreen","skyblue","yellow"])

plt.xlabel('Duration')

plt.ylabel('Count')

plt.title('Number of Investors by Investment Duration')

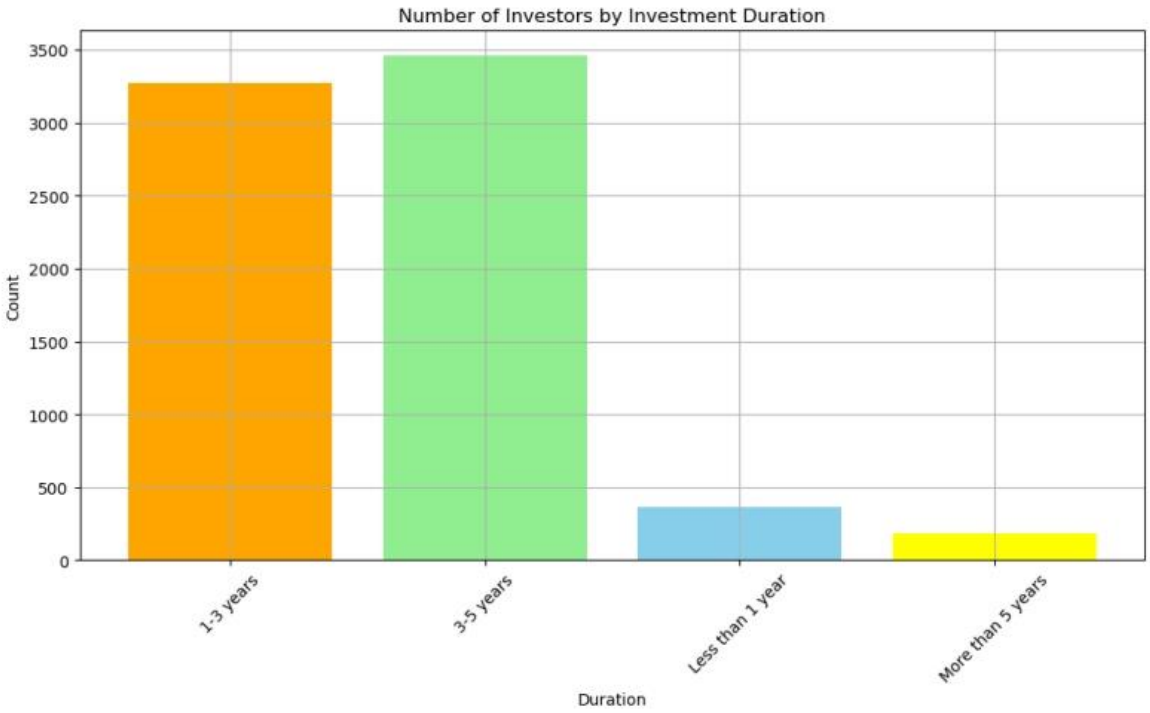
plt.xticks(rotation=45)

plt.grid(True)

plt.show()

Explanation:

1. Grouping and Counting:
    - df_duration = df.groupby("Duration").size().reset_index(name="count"): Groups the data by "Duration" and counts the number of occurrences (size()), then resets the index (reset_index(name="count")) to create a DataFrame with columns 'Duration' and 'count'.
2. Plotting:
    - plt.figure(figsize=(12, 6)): Creates a new figure with a specified size (width=12 inches, height=6 inches).
    - plt.bar(df_duration['Duration'], df_duration['count'], color=['orange', 'lightgreen', 'skyblue', 'yellow']): Plots a bar chart where df_duration['Duration'] represents the x-axis (investment durations) and df_duration['count'] represents the y-axis (number of investors). color parameter specifies different colors for each bar.
3. Labels and Title:
    - plt.xlabel('Duration'): Sets the label for the x-axis.
    - plt.ylabel('Count'): Sets the label for the y-axis.
    - plt.title('Number of Investors by Investment Duration'): Sets the title for the plot.
4. Adjust x-axis Labels:
    - plt.xticks(rotation=45): Rotates the x-axis labels by 45 degrees for better readability.
5. Grid Lines:
    - plt.grid(True): Adds grid lines to the plot for better readability.
6. Display Plot:
    - plt.show(): Finally, displays the bar chart with all the specified settings.



Inference

• The majority of investors aim for the medium term.

• 1-3 years(low term) ranges above 3000

• 3-5 (medium term) ranges above 3300

• Less than 1 year ranges below 500

• more than 5 years ranges below 500

How often do investors check their investments

df_Invest_Monitor = df["Invest_Monitor"].value_counts().reset_index(name="count")

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

plt.bar(df_Invest_Monitor["Invest_Monitor"], df_Invest_Monitor["count"],
color=["skyblue","orange","lightgreen"])

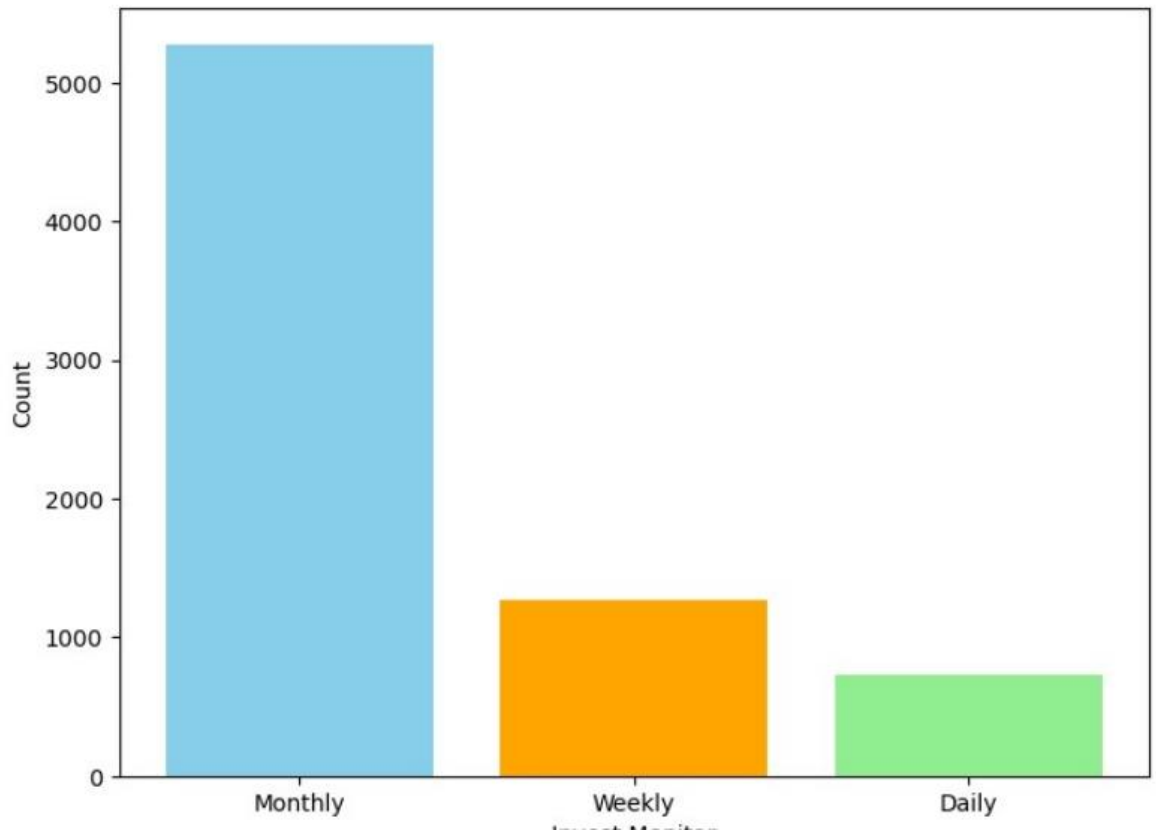
plt.xlabel("Invest Monitor")

plt.ylabel("Count")

plt.show()

Explanation:

1. **Counting Frequencies**:
    - df["Invest_Monitor"].value_counts().reset_index(name="count"): Counts the occurrences of each value in the "Invest_Monitor" column (value_counts()), then resets the index to create a DataFrame with columns 'index' (invest monitor categories) and 'count' (frequency counts).
2. **Plotting**:
    - plt.figure(figsize=(8, 6)): Creates a new figure with a specified size (width=8 inches, height=6 inches).
    - plt.bar(df_Invest_Monitor["index"], df_Invest_Monitor["count"], color=["skyblue", "orange", "lightgreen"]): Plots a bar chart where df_Invest_Monitor["index"] represents the x-axis (invest monitor categories) and df_Invest_Monitor["count"] represents the y-axis (frequency counts). color parameter specifies different colors for each bar.
3. **Labels and Title**:
    - plt.xlabel("Invest Monitor"): Sets the label for the x-axis.
    - plt.ylabel("Count"): Sets the label for the y-axis.
    - plt.title("Frequency of Investors Checking Their Investments"): Sets the title for the plot.
4. **Display Plot**:
    - plt.show(): Finally, displays the bar chart with all the specified settings.



Inference

• The majority of investors prefer to track their investments on a monthly basis.

• Monthly basis tracking ranges above 5000

• Weekly basis tracking ranges between 1000 - 1300

• Daily basis tracking ranges below 1000

How much return do male and female investors expect from their investments

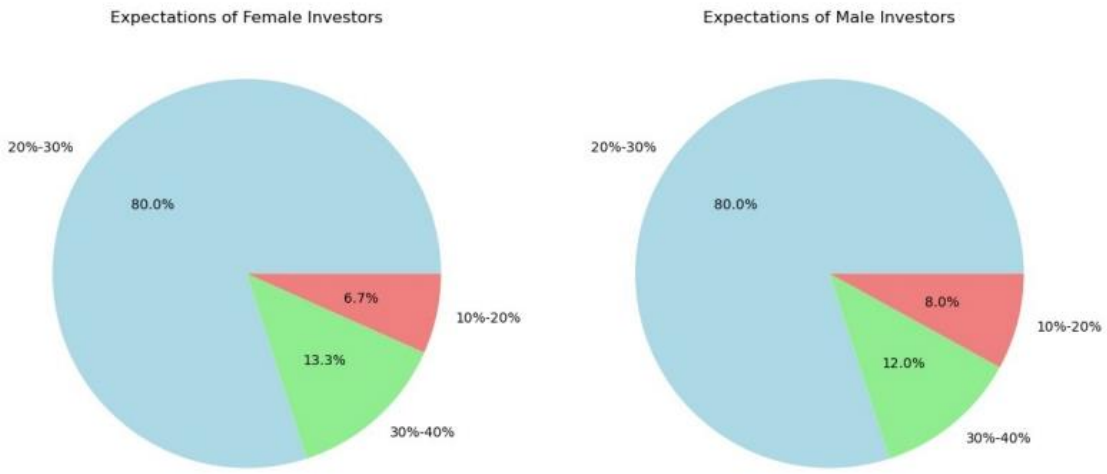
```
female_expectations = df[df['gender'] == 'Female']['Expect'].value_counts()

male_expectations = df[df['gender'] == 'Male']['Expect'].value_counts()

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 7))

ax1.pie(female_expectations, labels=female_expectations.index, autopct='%1.1f%%',
colors=['lightblue', 'lightgreen', 'lightcoral'])

ax1.set_title('Expectations of Female Investors')

ax2.pie(male_expectations, labels=male_expectations.index, autopct='%1.1f%%', colors=['lightblue',
'lightgreen', 'lightcoral'])

ax2.set_title('Expectations of Male Investors')

plt.show()
```

Explanation:

1. Data Preparation:
    - female_expectations = df[df['gender'] == 'Female']['Expect'].value_counts(): Filters the DataFrame to get the count of each expectation category for female investors.
    - male_expectations = df[df['gender'] == 'Male']['Expect'].value_counts(): Filters the DataFrame to get the count of each expectation category for male investors.
2. Subplots:
    - fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 7)): Creates a figure with 1 row and 2 columns of subplots. ax1 and ax2 represent the two subplots where we'll plot expectations for female and male investors, respectively.
3. Pie Charts:
    - ax1.pie(female_expectations, labels=female_expectations.index, autopct='%1.1f%%', colors=['lightblue', 'lightgreen', 'lightcoral']): Creates a pie chart on the first subplot (ax1) for female investors, using female_expectations as data. labels parameter sets the labels, autopct='%1.1f%%' displays percentages, and colors specifies colors for each segment.
    - ax1.set_title('Expectations of Female Investors'): Sets the title for the first subplot.
    - ax2.pie(male_expectations, labels=male_expectations.index, autopct='%1.1f%%', colors=['lightblue', 'lightgreen', 'lightcoral']): Creates a pie chart on the second subplot (ax2) for male investors, using male_expectations as data. Similar parameters as above are used.

- ax2.set_title('Expectations of Male Investors'): Sets the title for the second subplot.

4. Display Plot:
   - plt.show(): Finally, displays the entire figure with both pie charts.



Inference

• 80% of men and female expect a return of 20%-30% from investments.

• Female Investors(6.7 % expects 10%-20% returns), (13.3% expects 30%-40%)

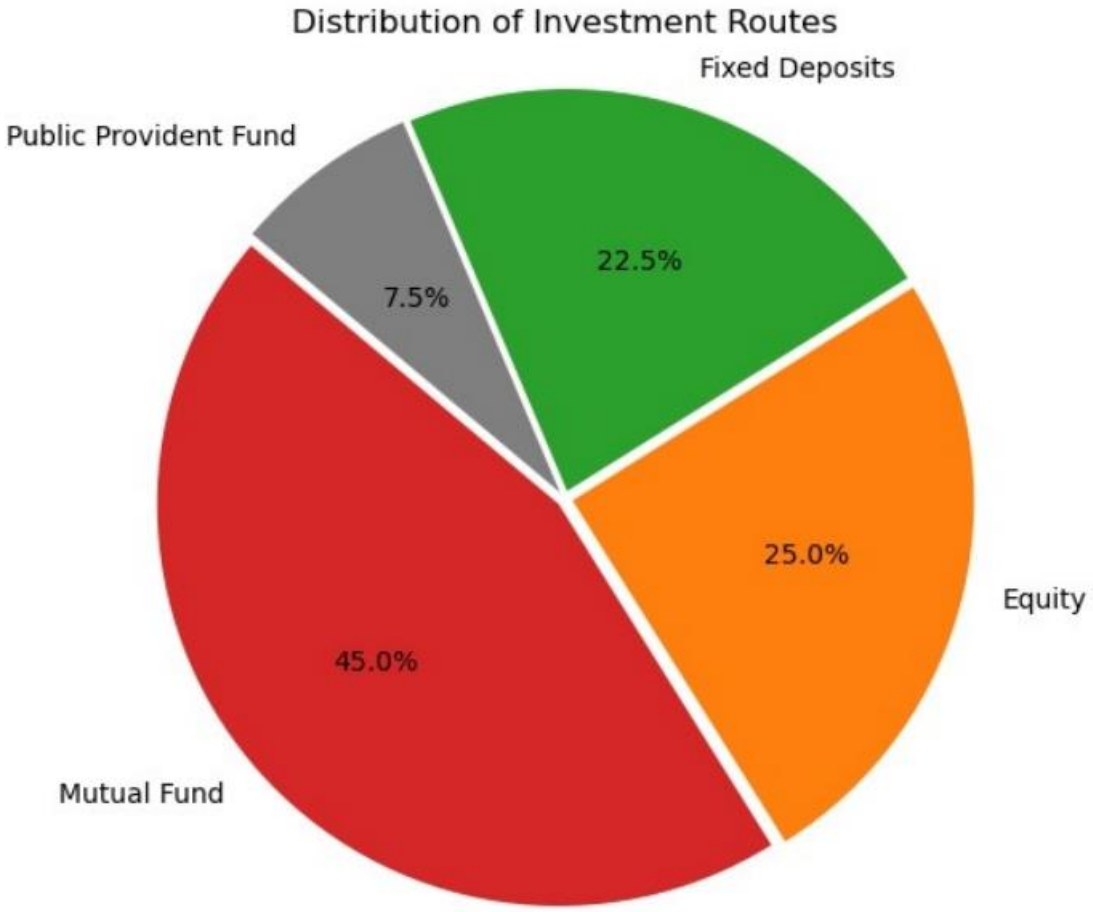
• Male Investors(8.0% expects 10%-20%), (12% expects 30%- 40%)

```
# Distribution of Investment Routes

import random

from matplotlib import colors as mcolors

avenue_counts = df['Avenue'].value_counts()

colors = random.sample(list(mcolors.TABLEAU_COLORS), len(avenue_counts))

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

explode = [0.02] * len(avenue_counts)

plt.pie(avenue_counts, labels=avenue_counts.index, autopct='%1.1f%%', startangle=140,
explode=explode, colors=colors)

plt.title('Distribution of Investment Routes')

plt.axis('equal')
```

plt.show()

Explanation:

1. Counting Frequencies:
    - avenue_counts = df['Avenue'].value_counts(): Counts the occurrences of each investment route ('Avenue') in the DataFrame df.
2. Generating Colors:
    - colors = random.sample(list(mcolors.TABLEAU_COLORS), len(avenue_counts)): Generates a list of random colors from the TABLEAU_COLORS palette provided by matplotlib.colors.
3. Plotting:
    - plt.figure(figsize=(8, 6)): Creates a new figure with a specified size (width=8 inches, height=6 inches).
    - explode = [0.02] * len(avenue_counts): Creates a list where each element is 0.02, which is used to slightly explode each slice of the pie chart for emphasis.
4. Pie Chart:
    - plt.pie(avenue_counts, labels=avenue_counts.index, autopct='%1.1f%%', startangle=140, explode=explode, colors=colors): Plots a pie chart where avenue_counts provides the sizes of the slices, labels=avenue_counts.index provides the labels for each slice, autopct='%1.1f%%' displays the percentage on each slice, startangle=140 sets the starting angle of the pie chart, explode=explode specifies the explode values for each slice, and colors=colors sets the colors of the slices.
5. Title and Axis Aspect:
    - plt.title('Distribution of Investment Routes'): Sets the title for the pie chart.
    - plt.axis('equal'): Ensures the pie chart is drawn as a circle, maintaining an equal aspect ratio.
6. Display Plot:
    - plt.show(): Finally, displays the pie chart with all the specified settings.



Inference

• Investors prefer Mutual Funds than other

• Mutual Fund - 45.0% prefers mutual fund

• Equity - 25.0% prefers Equity

• Fixed Deposits - 22.5% prefers Fixed deposits

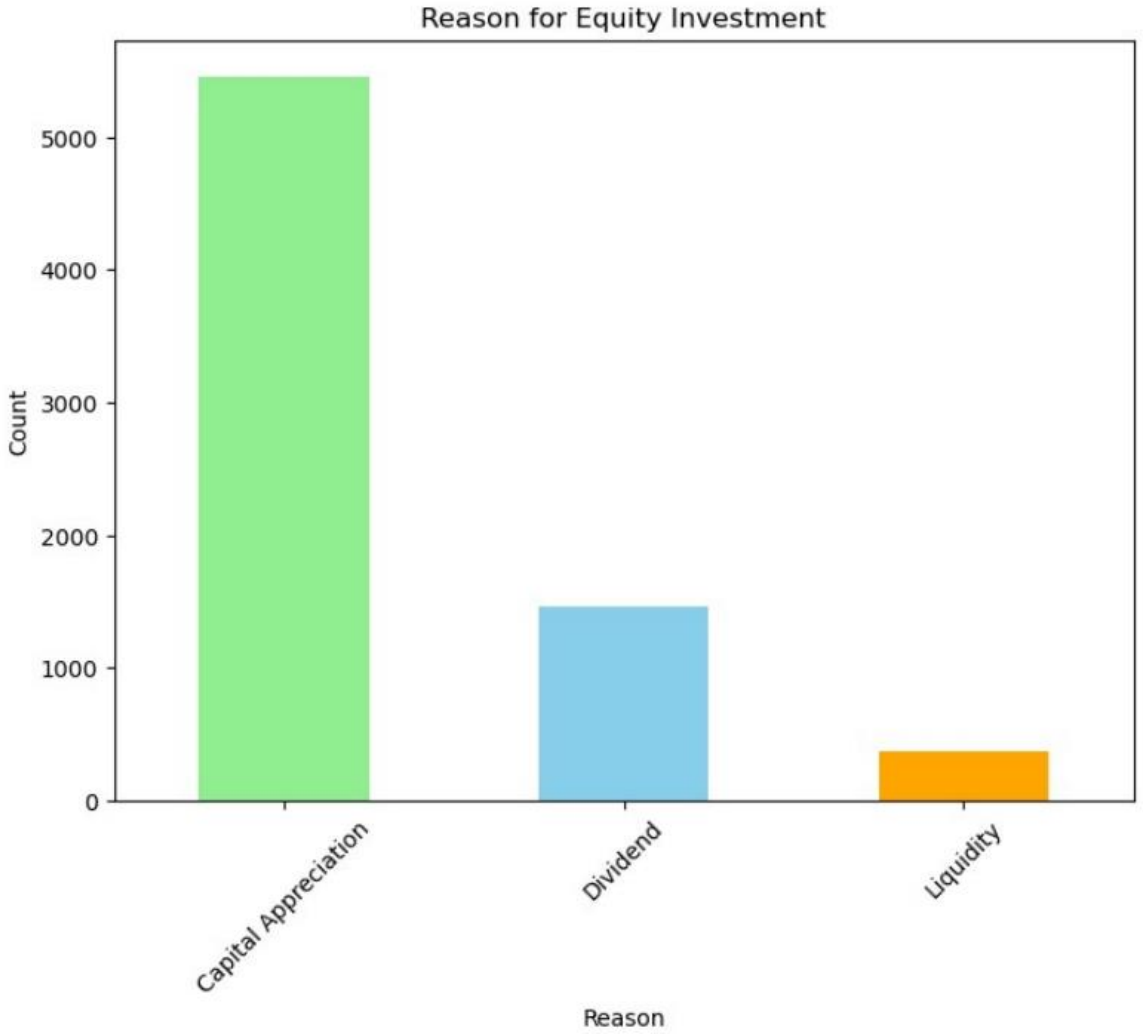
• PPF - 7.5% prefers Public Provident fund

```
# Reason for Equity Investment
reason_equity_counts = df["Reason_Equity"].value_counts()
plt.figure(figsize=(8, 6))
reason_equity_counts.plot(kind='bar', color=["lightgreen","skyblue","orange"])
plt.title('Reason for Equity Investment')
plt.xlabel('Reason')
```

```
plt.ylabel('Count')

plt.xticks(rotation=45)

plt.show()
```

Explanation:

1. Counting Frequencies**:**
    - reason_equity_counts = df["Reason_Equity"].value_counts(): Counts the occurrences of each reason for equity investment ('Reason_Equity') in the DataFrame df.
2. Plotting**:**
    - plt.figure(figsize=(8, 6)): Creates a new figure with a specified size (width=8 inches, height=6 inches).
    - reason_equity_counts.plot(kind='bar', color=["lightgreen", "skyblue", "orange"]): Plots a bar chart where reason_equity_counts provides the data. kind='bar' specifies the type of plot, and color parameter sets the colors for each bar.
3. Labels and Title**:**
    - plt.title('Reason for Equity Investment'): Sets the title for the plot.
    - plt.xlabel('Reason'): Sets the label for the x-axis.
    - plt.ylabel('Count'): Sets the label for the y-axis.
    - plt.xticks(rotation=45): Rotates the x-axis labels by 45 degrees for better readability.
4. Display Plot**:**
    - plt.show(): Finally, displays the bar chart with all the specified settings.



Inference

• Investors who chose the stock said they were investing for Capital Appreciation.

• Capital Appreciation ranges above 5000

• Dividend ranges between 1000 - 2000

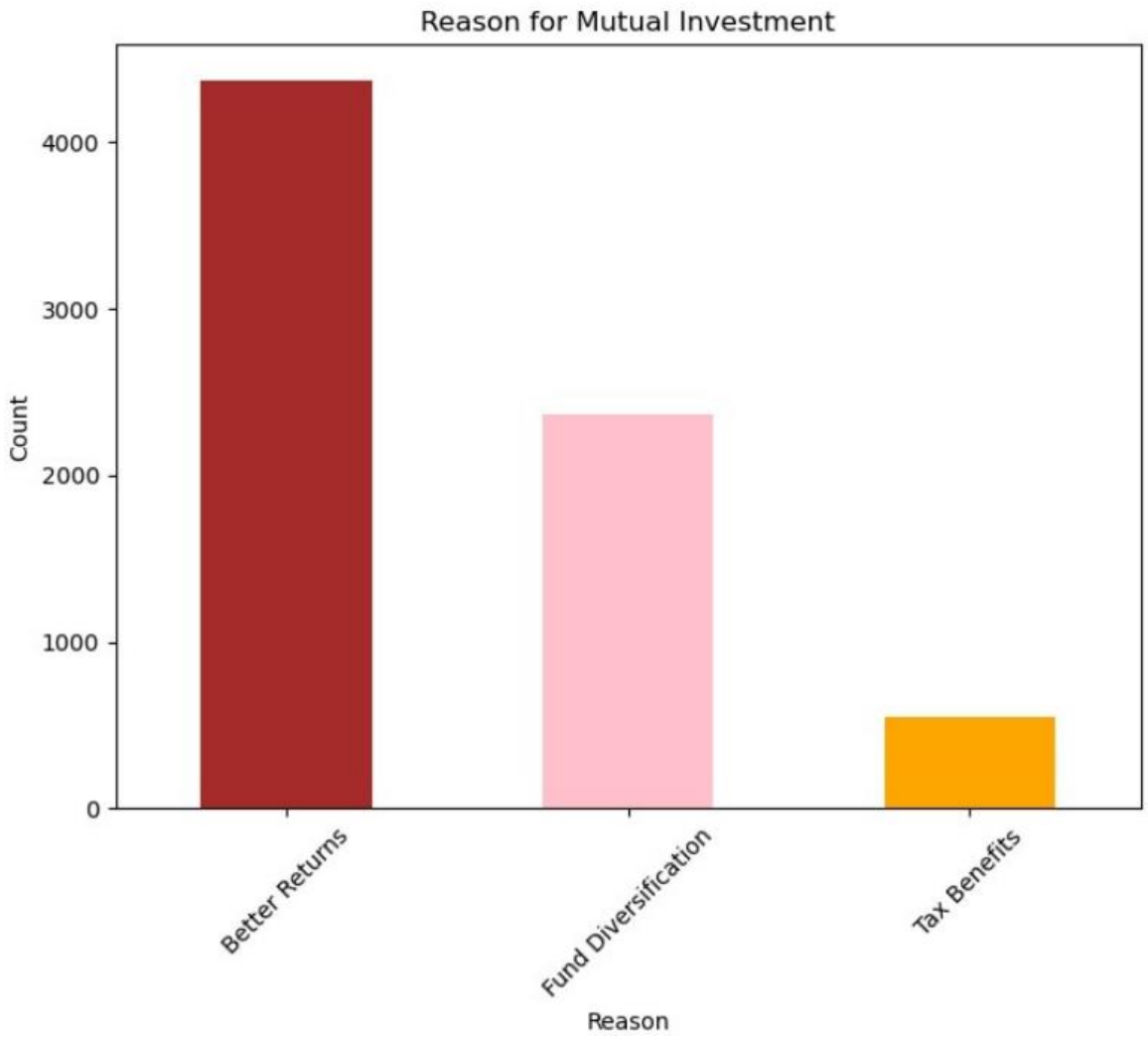
• Liquidity ranges below 1000

```
# Reason for mutual
reason_mutual_counts = df["Reason_Mutual"].value_counts()
plt.figure(figsize=(8, 6))
reason_mutual_counts.plot(kind='bar', color=["brown","pink","orange"])
plt.title('Reason for Mutual Investment')
```

```
plt.xlabel('Reason')

plt.ylabel('Count')

plt.xticks(rotation=45)

plt.show()
```

Explanation:

1. Counting Frequencies**:**
    - reason_mutual_counts = df["Reason_Mutual"].value_counts(): Counts the occurrences of each reason for mutual fund investment ('Reason_Mutual') in the DataFrame df.
2. Plotting**:**
    - plt.figure(figsize=(8, 6)): Creates a new figure with a specified size (width=8 inches, height=6 inches).
    - reason_mutual_counts.plot(kind='bar', color=["brown", "pink", "orange"]): Plots a bar chart where reason_mutual_counts provides the data. kind='bar' specifies the type of plot, and color parameter sets the colors for each bar.
3. Labels and Title**:**
    - plt.title('Reason for Mutual Fund Investment'): Sets the title for the plot.
    - plt.xlabel('Reason'): Sets the label for the x-axis.
    - plt.ylabel('Count'): Sets the label for the y-axis.
    - plt.xticks(rotation=45): Rotates the x-axis labels by 45 degrees for better readability.
4. Display Plot**:**
    - plt.show(): Finally, displays the bar chart with all the specified settings.



Inference

• Those who preferred Mutual Funds said that they chose it because it offered better returns.

• Better Returns ranges above 4000

• Fund Diversification ranges between 2000 – 3000

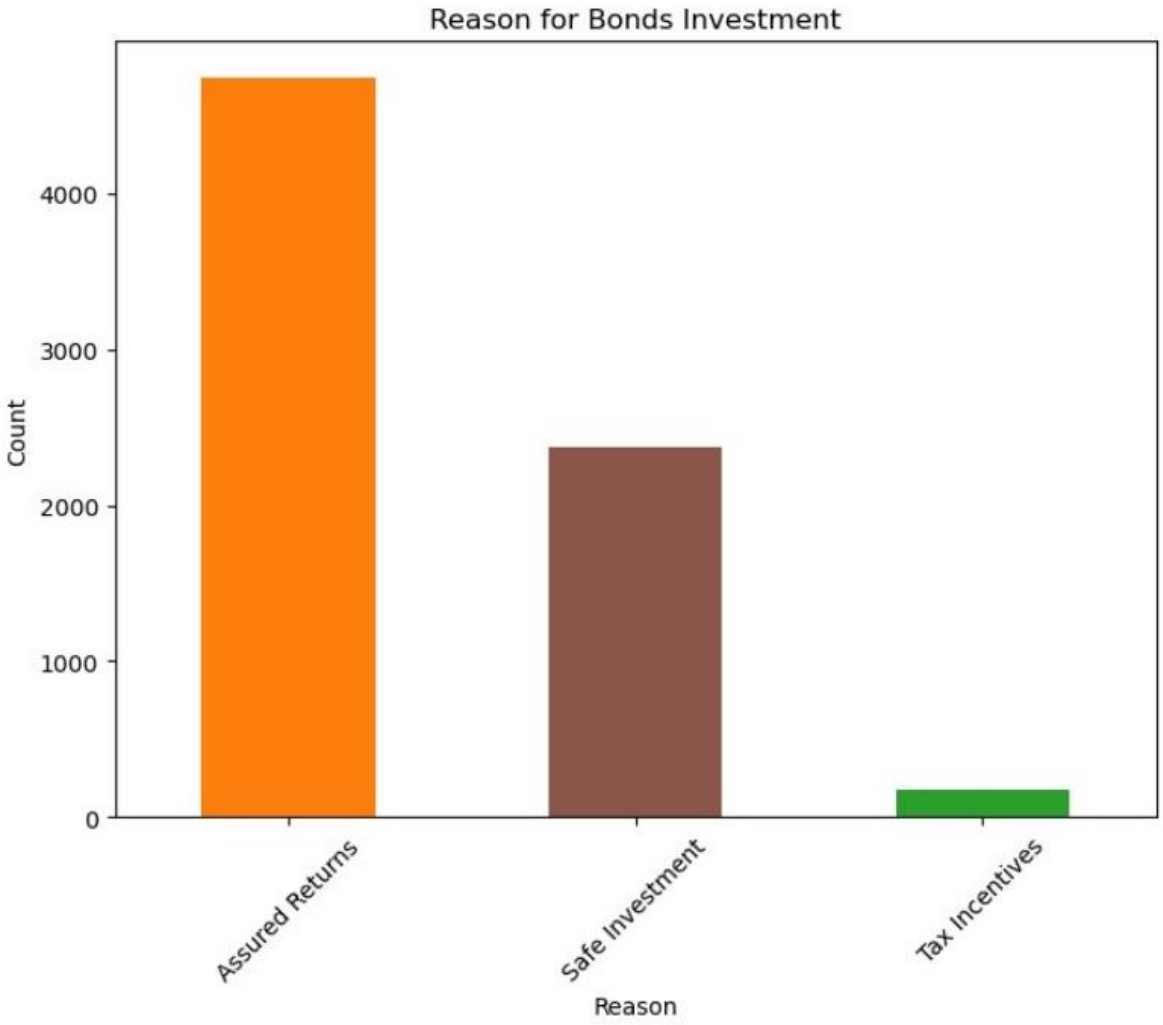
• Tax benefits ranges below 1000

```
# Reason for Bonds Investment
reason_bonds_counts = df["Reason_Bonds"].value_counts()
colors = random.sample(list(mcolors.TABLEAU_COLORS), len(avenue_counts))
plt.figure(figsize=(8, 6))
reason_bonds_counts.plot(kind='bar', color=colors)
plt.title('Reason for Bonds Investment')
```

```
plt.xlabel('Reason')

plt.ylabel('Count')

plt.xticks(rotation=45)

plt.show()
```

Explanation:

1. Counting Frequencies:
    - reason_bonds_counts = df["Reason_Bonds"].value_counts(): Counts the occurrences of each reason for bonds investment ('Reason_Bonds') in the DataFrame df.
2. Generating Colors**:**
    - colors = random.sample(list(mcolors.TABLEAU_COLORS), len(reason_bonds_counts)): Generates a list of random colors from the TABLEAU_COLORS palette provided by matplotlib.colors, based on the number of unique reasons for bonds investment.
3. Plotting:
    - plt.figure(figsize=(8, 6)): Creates a new figure with a specified size (width=8 inches, height=6 inches).
    - reason_bonds_counts.plot(kind='bar', color=colors): Plots a bar chart where reason_bonds_counts provides the data. kind='bar' specifies the type of plot, and color parameter sets the colors for each bar.
4. Labels and Title**:**
    - plt.title('Reason for Bonds Investment'): Sets the title for the plot.
    - plt.xlabel('Reason'): Sets the label for the x-axis.
    - plt.ylabel('Count'): Sets the label for the y-axis.
    - plt.xticks(rotation=45): Rotates the x-axis labels by 45 degrees for better readability.
5. Display Plot**:**
    - plt.show(): Finally, displays the bar chart with all the specified settings.



Inference

• Those who preferred Mutual Funds said that they chose it because it has a Assured Return.

• Assured Returns ranges above 4000

• Safe investments ranges between 2000 – 3000

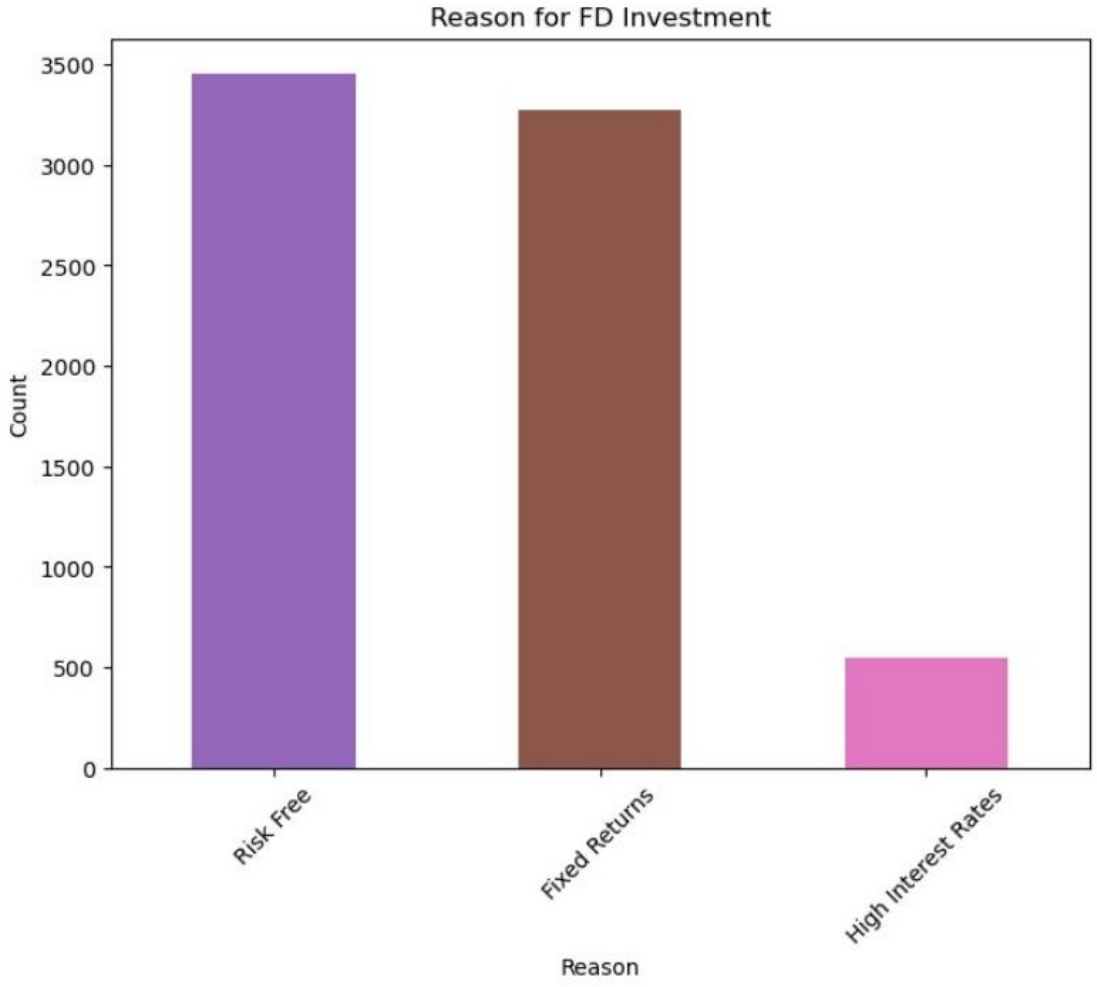
• Tax Incentives ranges below 1000

```
# Reason for Term Deposit
reason_fd_counts = df["Reason_FD"].value_counts()
colors = random.sample(list(mcolors.TABLEAU_COLORS), len(avenue_counts))
plt.figure(figsize=(8, 6))
reason_fd_counts.plot(kind='bar', color=colors)
plt.title('Reason for FD Investment')
```

```
plt.xlabel('Reason')

plt.ylabel('Count')

plt.xticks(rotation=45)

plt.show()
```

Explanation:

1. Counting Frequencies**:**
    - reason_fd_counts = df["Reason_FD"].value_counts(): Counts the occurrences of each reason for term deposit (FD) investment ('Reason_FD') in the DataFrame df.
2. Generating Colors**:**
    - colors = random.sample(list(mcolors.TABLEAU_COLORS), len(reason_fd_counts)): Generates a list of random colors from the TABLEAU_COLORS palette provided by matplotlib.colors, based on the number of unique reasons for FD investment.
3. Plotting**:**
    - plt.figure(figsize=(8, 6)): Creates a new figure with a specified size (width=8 inches, height=6 inches).
    - reason_fd_counts.plot(kind='bar', color=colors): Plots a bar chart where reason_fd_counts provides the data. kind='bar' specifies the type of plot, and color parameter sets the colors for each bar.
4. Labels and Title**:**
    - plt.title('Reason for FD Investment'): Sets the title for the plot.
    - plt.xlabel('Reason'): Sets the label for the x-axis.
    - plt.ylabel('Count'): Sets the label for the y-axis.
    - plt.xticks(rotation=45): Rotates the x-axis labels by 45 degrees for better readability.
5. Display Plot**:**
    - plt.show(): Finally, displays the bar chart with all the specified settings.



Inference

• the reason for turning to this investment instrument is that it is risk-free in Term Deposits and Fixed Income.

• Risk Free ranges between 3000 - 3500

• Fixed Returns ranges between 3000 - 3500 but quite less than Risk Free category

• High rates rated lies between 0 – 500

One - sample T test

from scipy.stats import ttest_1samp

from scipy.stats import ttest_1samp

```
hm = 35 # HYPOTHESISED MEAN

t_stat, p_value = ttest_1samp(df['age'],hm)

alpha = 0.08 # acceptance error percent

print(f"T-statistic:{t_stat}")

print(f"P-value:{p_value}")

if p_value<alpha:

    print(f"Reject null hypothesis at alpha ={alpha}")

else:

    print(f"Failed to reject null hypothesis at alpha = {alpha}")
```

- Importing the ttest_1samp function:
  - from scipy.stats import ttest_1samp: Imports the ttest_1samp function from the scipy.stats module, which is used for conducting a one-sample t-test.
- Setting the Hypothesized Mean (hm):
  - hm = 35: This is the hypothesized mean age of the population.
- Performing the t-test:
  - t_stat, p_value = ttest_1samp(df['age'], hm): Conducts a one-sample t-test on the age data (df['age']) against the hypothesized mean (hm). It returns the t-statistic (t_stat) and the p-value (p_value).
- Setting the Significance Level (Alpha):
  - alpha = 0.08: This is the chosen significance level, often denoted as alpha, which represents the threshold for determining statistical significance.
- Interpreting the Results:
  - print(f"T-statistic:{t_stat}"): Prints the calculated t-statistic.
  - print(f"P-value:{p_value}"): Prints the calculated p-value.
  - The subsequent if-else statement compares the p-value (p_value) to the significance level (alpha):
    - If p_value is less than alpha, it prints "Reject null hypothesis", indicating that there is enough evidence to reject the null hypothesis that the mean age is equal to hm.
    - Otherwise, it prints "Failed to reject null hypothesis", suggesting that there is not enough evidence to reject the null hypothesis.

Conclusion

Through this analysis, several insights into investment behaviors among the surveyed population have been revealed:

- Investment Preferences: Mutual funds and equity markets are popular investment avenues, reflecting investor interest in potential growth opportunities.
- Reasons for Investment: Reasons vary widely, with factors like long-term growth potential, safety, and convenience influencing decisions across different investment options.
- Demographic Differences: Gender-specific differences in investment preferences and return expectations highlight varying risk appetites and financial goals.
- Age and Investment Behavior: Analysis suggests that age may influence investment decisions, with younger investors possibly seeking higher-risk, higher-return options compared to older demographics.

Understanding these patterns can assist financial advisors and policymakers in tailoring investment products and strategies that cater to diverse investor needs and preferences. Additionally, continuous monitoring of these trends can help predict future investment behaviors and adapt financial offerings accordingly.