Financial Investment Product Recommender System

1.0 BUSINESS UNDERSTANDING

BUSINESS PROBLEM STATEMENT:

Our financial services company has a diverse portfolio of investment products, yet the vast majority of our existing customers hold only one product—the Money Market Fund. Despite a broad array of offerings (Balanced Fund, Dollar Fund, Equity Fund, Fixed Income Fund, and Wealth Fund), our product penetration per customer (PPC) remains exceptionally low. This indicates a significant opportunity to cross-sell additional products to our existing customer base, which would increase customer value, loyalty, and the company's overall profitability.

Currently, our customers' data includes key information that could be leveraged to tailor product recommendations. These data points include:

- **Location (town)**
- **Gender**
- **Customer-relationship or beneficiary information** (as customers may have more than one relationship or beneficiary associated with them)
- **Customer age and DOB**
- **Beneficiary age and DOB**

Our goal is to create a user-friendly, intelligent recommender system that can analyze this existing data to suggest additional, relevant financial products to each customer. This system should be able to identify patterns or trends in customer profiles, uncover customer needs, and map those needs to suitable financial products, increasing our PPC in an efficient, cost-effective manner.

OBJECTIVES:

1. **Customer Retention and Loyalty**: By offering personalized recommendations, we aim to build deeper, more personalized relationships with our customers, making them more likely to stay with us long-term.
2. **Increased Revenue per Customer**: A successful cross-selling strategy would increase the average number of products per customer, boosting overall portfolio revenue.
3. **User-Friendly Experience**: Ensuring a straightforward, accessible interface for customers to explore new financial products, particularly given that our target customers may have limited experience with financial diversification.

2.0 DATA UNDERSTANDING

In [288…

```
#importing the necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
import nbformat
import pickle

from sklearn.metrics.pairwise import cosine_similarity
from scipy.spatial.distance import cosine
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from datetime import datetime
from sklearn.neighbors import NearestNeighbors
from sklearn.metrics import accuracy_score
```

First import the neccessary libraries required to help us carry out our project.

In [289…

```
#load the dataset
df = pd.read_csv("DATA\single_member.csv")
df
```

Out[289…

	member_no	reg_date	dob	hse_no	gender	town	relationship	beneficiery_dob
0	99996	2023-10-01 00:00:00.000	1998-04-06 00:00:00	Single Member	Female	NAIROBI	Partner	1998-01-26
1	99996	2023-10-01 00:00:00.000	1998-04-06 00:00:00	Single Member	Female	NAIROBI	Partner	1998-01-26
2	99996	2023-10-01 00:00:00.000	1998-04-06 00:00:00	Single Member	Female	NAIROBI	Partner	1998-01-26
3	99996	2023-10-01 00:00:00.000	1998-04-06 00:00:00	Single Member	Female	NAIROBI	Partner	1998-01-26
4	99996	2023-10-01 00:00:00.000	1998-04-06 00:00:00	Single Member	Female	NAIROBI	Partner	1998-01-26
...	...	...	...	...	...	...	...	..
7532949	2	2011-06-07	1965-05-06	Single Member	Male	NAIROBI	NaN	NaN
7532950	2	2011-06-07	1965-05-06	Single Member	Male	NAIROBI	NaN	NaN
7532951	2	2011-06-07	1965-05-06	Single Member	Male	NAIROBI	NaN	NaN
7532952	2	2011-06-07	1965-05-06	Single Member	Male	NAIROBI	NaN	NaN
7532953	2	2011-06-07	1965-05-06	Single Member	Male	NAIROBI	NaN	NaN

7532954 rows × 9 columns

Our dataset has 9 columns and 7,532,954 rows.

In [290…

```
#inspecting the columns in the dataframe
df.columns
```

Out[290…

```
Index(['member_no', 'reg_date', 'dob', 'hse_no', 'gender', 'town',
       'relationship', 'beneficiery_dob', 'portfolio'],
```

```
      dtype='object')
```

We inspected the various columns within our dataset.

In [291…

```python
#checking the unique values from the member_no column
df['member_no'].unique()
```

Out[291…

```
array([99996, 99994, 99993, ...,    19,     3,     2], dtype=int64)
```

In [292…

```python
#checking the unique values from the gender column
df['gender'].unique()
```

Out[292…

```
array(['Female', 'Male', 'FEMALE', nan, 'MALE', 'F', 'M'], dtype=object)
```

In [293…

```python
#checking the unique values from the town column
df['town'].unique()
```

Out[293…

```
array(['NAIROBI', nan, 'NAIROBI ', 'THIKA', 'NAKURU', 'MOMBASA ',
       'NDANAI ', 'KISUMU', 'KIKUYU', 'JUJA', 'MACHAKOS', 'MERU',
       'KERICHO', 'MUKURWEINI', 'KATANGI', 'THIKA,KIAMBU', 'NANYUKI',
       'RUAI ', 'MALINDI', 'NYAHURURU', 'MUMIAS', 'KAKAMEGA', 'LODWAR',
       'Nairobi ', 'KITENGELA', 'LIMURU', 'THIKA ', 'MIGORI', 'ELDORET',
       'KITUI', 'RUARAKA', 'Ruiru', 'NJORO', 'EMBU', 'SORI', 'NGONG',
       'MASENO', 'GATUNDU', 'KAHURO', 'KILIFI', 'RUIRU', 'KERUGOYA',
       'KARATINA', 'KITALE ', 'MOMBASA', 'BUNGOMA', 'KISII', '-',
       'Nairobi', 'KINAMBA', 'SIAYA', 'KERICHO ', 'KIAMBU', 'KITALE',
       'BUSIA', 'RUNYENJES', 'VILLAGE MARKET', 'Kapsabet', 'KAGWE',
       'NANDI HILLS', 'KUTUS', 'DAGORETTI', 'LITEIN', 'KAJIADO', 'CHUKA',
       'NYERI', 'NAIROBI, NAIROBI', 'Kisumu ', 'KANJUKU', 'EMALI',
       'GILGIL', 'Kiambu ', 'KANGEMA', 'TIRIKI', 'KARATINA ', 'OTHAYA',
       'KARURI ,KIAMBU', 'MURANGA ', 'Mombasa', 'GATUKUYU ', 'MLOLONGO',
       'UTAWALA ', 'KAPENGURIA', 'KAPSABET', 'SUNA', 'BOMET', 'KENOL ',
       'NYERI ', 'BUNGOMA ', 'NDANAI', 'KALIMONI', 'KIGANJO',
       'KAPSOKWONY', 'Nakuru', 'KIAMBU ', 'nairobi', 'CHOGORIA',
       'GITHUNGURI', 'YOANI,SALAMA', 'LIMURU ', 'NAIROBI, KENYA.',
       'UKUNDA', 'MATUU', 'embu', 'NYALI', 'nakuru', 'MBALE', 'EMUHAYA',
       'TELFORD', 'NAIVASHA', 'KAREN', 'KILIFI ', 'HOLA', 'MAKINDU',
       'WOTE', 'DIANI', 'O', 'OLKALAU', 'KAREN ', 'UPLANDS', 'WANGÚRU',
       'KABIYET', 'KAKAMEGA ', 'SOTIK', 'NGONG ', 'CITY SQUARE', 'NKUBU',
       'KANGARI', 'NGONG HILLS ', 'NYAHURURU ', 'KARURI ', 'UTHIRU',
       'MARIMANTI', 'MANCHESTER', 'Kikuyu', '.', 'NAIRO ', 'kiambu',
       'VOI', 'EMBAKASI', 'MOLO', 'Siaya ', 'NGIYA', 'OLKALOU',
       'KANYAGIA', 'WANJOHI', 'kabete', 'Wamunyu', 'MARIAKANI',
       'KISERIAN', 'ARTHI RIVER ', 'SUBUKIA', 'KILGORIS', 'KIMININI',
       'LAGOS', 'SARE', 'ATHI RIVER', 'ngong hills ', 'WERUGHA',
       'NAIROBIP', 'MASINFA', 'kikuyu', 'OL JORO OROK', 'MAUA',
       'GIKUE MURANGA', 'KIMENDE', 'KIMILILI', 'KISUMU ', 'VIHIGA',
       ' NAIROBI ', 'BURUBURU', 'MATATHIA', 'RUAKA', 'UGUNJA', 'LONGISA',
       'ELDORATE ', 'KAPKATET', 'WINNIPEG', 'PIPELINE ', 'KAHAWA WEST ',
       'KANDARA', 'Kimilili ', 'MTWAPA ', 'MARAGUA', 'BANANA ', 'NYAMIRA',
       'AUBURN', 'KARURI', 'MTWAPA', 'ONGATA RONGAI', 'ELDORET ',
       'HOMABAY', 'WANGIGE', 'KENDUBAY', 'KIBWEZI', 'RONGAI', 'RUIRU ',
       'NORTH KINANGOP', 'MAKUYU', 'Nyeri ', 'Eldoret ', 'BOUTIGNY',
       '00506', 'MWINGI', '00100', 'KERUGOYA ', 'TALA', 'OLENGURUONE',
       'MAVINDINI', 'UHURU GARDENS', 'KIKUYU ', 'Ongata Rongai',
       'SOUTH KINANGOP', 'ELDAMA RAVINE', 'MITUNGURU', 'TOM MBOYA',
       'NAKURU ', 'MITABONI', 'NAROMORU', 'MURANGA', 'OLJOROROK',
       'BUTERE', 'OYUGIS', 'SILIBWET', 'EMBU ', 'Doha', 'Eldoret',
       'KANGEMI', 'NYERI-KIGUMO', 'KISERIAN ', 'Athi River', 'KENOL',
       'OL KALOU', 'WESTLANDS', 'Nyahururu ', 'ENDARASHA ', 'GPO/NAIROBI',
       'DAGORETI', 'OGEMBO ', 'mombasa', 'RUAI', 'DONYO SABUK', 'KEROKA',
       'LOWER KABETE', 'MWEIGA', 'Naitobi', 'TOKYO', 'Embu', 'KIRINYAGA',
       'KIANYAGA', 'SIAYA ', 'ADLISWIL', 'BURU BURU', 'MERU ', 'NYAKIO',
       'KWALE', 'AMUKURA', 'NAROK', 'KOBUJOI', "WANG'URU", 'BONDO',
       'Kericho', 'KIKIMA ', 'CHANGAMWE', 'BUTULA', 'NAIROBI KAYOLE',
       'KANGUNDO', 'MARIAKANI ', 'TIMBOROA', 'KIANJAI', 'KALOLENI ',
       'ISINYA', 'IOWA CITY, IOWA', 'Garsen ', 'MUKURWE-INI',
```

'NAIROBI GPO', 'MANYATTA', 'SARE-OWENDO', 'NANYUKI ', 'NDHIWA',
'ELBURGON', 'Orange', 'MAKUENI', 'kisumu ', 'MACHAKOS ',
'KABARNET ', "Murang'a ", 'JUJA, KIAMBU', 'Bungoma', 'MAWEGO',
'Bath', 'ROYSAMBU', 'Nanyuki', 'SAGANA', 'MACH ', 'Nyeri', 'Ruaka',
'KITUI ', 'KAJIADO ', 'NUNGUNI', 'RONGAI ', 'KISII ', 'WANGIGE ',
'Makongeni Jogoo Rd', 'nairobi ', 'Kikuyu ', 'Kisii', 'MSAMBWENI',
'BUJUMBURA', 'GITHONGO', 'KEROKA/KISII', 'BURUBURU, NAIROBI',
'MALAVA', 'KIJABE', 'Woolwich ', 'NDERE-SIAYA', 'AHERO',
'MUHORONI ', 'Nakuru ', 'kahuro', 'Malakisi ', 'MUHORONI',
'ISIOLO', 'SYOKIMAU ', 'ONGATA RONGAI ', 'Maralal ',
'KIBIRICHIA, MERU', 'GRANGER, INDIANA', 'WESTLAND ', 'MUKURWEINI ',
'MASII', 'NRB', 'KWISERO', 'ANKARA', 'SHIMBA HILLS', 'TAVETA',
'WOTE, MAKUENI ', 'WANGURU', 'KISSII', 'BURUBURU NAIROBI ',
'KIRITIRI', 'MESA ARIZONA', 'KAJIADO TOWN', 'KURIA',
'NAIROBI WEST', 'ENDARASHA', 'KIPKAREN ', '0', 'GAMBOGI', 'KOLA',
'Marsabit', 'WEBUYE', 'GAKUNGU', 'NAIROBJ',
'NAIROBI ,KAMITI ROAD RICHLAND', 'NKUBU ', 'MWATATE', 'MALAKISI',
'ITEN', 'KATHIANI', "MURANG'A", 'KAIRIA', 'UPLANDS ', 'RONGO',
'UHURU GARDEN', 'KINANGOP', 'BAHATI NAKURU ', 'MATILIKU', 'meru',
'Limuru', 'MALAA', 'TURBO', 'MARIMA -MERU HIGHWAY ', 'OLJORO OROK',
'homabay ', 'KIKIMA', 'ADAMS ARCADE', 'MUNICH', 'MUMIAS ',
'KALIMONI ', 'CHWELE', 'MARALAL', 'NAMBALE', 'NAIROBI COUNTY',
'Narok ', 'TURBO ', 'KANGARI ', 'Pécs ', 'thika', 'KATSE',
'thika ', 'CHEBUNYO', 'OL JORO OROK ', 'MOBIL PLAZA', 'WAMUNYU ',
'MAGUNGA', 'RORET', 'YALA', 'KAHUHIA', "KENOL MURANG'A",
'Machakos', 'UKWALA', 'KALOLENI', 'JOONDALUP', 'MIGORI ',
'MARAGOLI', 'MARMANET', 'IKANGA', 'KINOO', 'OYUGIS ', 'LANGATA',
'Kiambu', 'Malindi', 'NAIROBI.', 'MAKUENI ', 'Bungoma ',
'GITHUNGURI ', 'MACHOKOS', 'KOMBEWA ', 'UTHIRU ', 'KITENGELA ',
'BANANA', 'CHEPKORIO', 'MASII ', 'Gitugi', '. ', 'RUIRY', 'Kisumu',
'MBUMBUNI-MACHAKOS', 'WAMUYU', 'siaya', 'Kakamega', 'TAITA TAVETA',
'Bumala', 'KIMENDE ', 'BRISTOL', 'MATUNDA', 'KARABA', 'SULTAN',
'GATURA', 'KASARANI ,NAIROBI ', 'NANDI', 'MAGUTUNI', 'CHUKA ',
'NGONG HILL', 'ngong', 'KIANGAI', 'MBITA', 'Mumias', 'KAMBITI',
'MUMBI', 'Juja', 'WODANGA', 'KIRIANI', 'KABARNET', 'MISIKHU',
'Thika', 'NYANSIONGO', 'nyeri ', 'NAIVASHA ', "MURANG'A ",
'KAPCHENO', 'NGARA', 'MADARAKA', 'kitengela ', 'BONDO ', 'WAMUNYU',
'NYANDARUA', 'MOGOGOSIEK', '00', 'NAIROBI/JUJA', 'NA ', 'LUANDA',
'KLEMZIG', 'TEL AVIV', 'ongata rongai ', '30-60101 Manyatta ',
'12410 Nyeri', 'KATHONZWENI ', 'GITHURAI ', 'WESRN AUSTRALIA',
'SABASABA ', 'Gilgil ', 'ISEBANIA', 'KIRIAINI', 'MUTHETHENI ',
'Mombasa ', 'Kitengela', 'CARLIFONIA', 'NIL',
'NAIROBI, NAIROBI AREA, KENYA', 'SOY', 'Sabasaba', 'SABASABA',
'COLUMBIA', 'Homa Bay ', 'MARIMA', 'SEGA', 'KAYOLE', 'BUDD LAKE',
'KITHYOKO', 'KIJABE ', 'Mbale', 'KITHIMANI', 'HANOI', 'chuka',
'KILINDINI', 'MERY', 'CHELMSFORD', 'LOITOKITOK', 'MOSORIOT',
'IKONGE', 'ENGINEER', 'Othaya', 'Ahero', 'MWEA', 'JOSKA ',
'NGAMBWA', 'MASINGA', 'KINANGOP ', 'LONDON', 'KILELESHWA ',
'SARIT CENTRE', 'mgange ', 'ST. MARYS NSW', 'GESIMA', 'Migori ',
'KHWISERO', 'SYOKIMAU', 'MTITO ANDEI ', 'nakuru ', 'NAIROBI CITY',
'ATHIRIVER', 'KABATI', 'Kitale', 'MPEKETONI', 'YOANI',
'KABARTONJO', 'NAMANGA', 'CHESINENDE', 'BAHATI', 'NZEEKA',
'MATATHIA, KAMAE', 'RONALD NGALA ', 'KUBUKUBU', 'THOGOTO',
'GATORA', 'NAI ', 'KUJA,KIAMBU', 'Machakos ', 'AMAGORO', 'Uthiru',
'Naivasha', 'KEROKA ', 'MAZERAS', 'EGERTON', 'KASARANI', 'Kimende',
'WAJIR', 'DUBAI', 'TWO RIVERS', 'NDARAGWA ', 'EASTLEIGH', 'KAGIO',
'HOMA BAY', 'KISII, KENYA', 'Thika ', 'NGECHA', 'Kakamega ',
'KERINGET', 'FORT WAYNE', 'STAREHE', 'GILGIL ', 'KAWANGWARE',
'GPO NAIROBI', 'GITUGI', 'Tala', 'kakamega ', 'TEXAS', 'GACHIE',
'Syokimau', 'Sydney ', 'LESSOS', 'Meru ', 'KIPKAREN RIVER',
'SOSIOT', 'LAKEWOOD SEATTLE', 'Lodwar', 'Kitale ', 'Membley',
'chepkorio', 'Luanda', 'Kijabe ', 'TONGAREN', 'karuri ', 'malindi',
'Ogembo ', 'Maryland', 'Kisii Town ', 'kajiado ', 'LIKONI',
'wangige ', 'Ngong', 'OLKALOU ', '4597-20100 Nakuru', 'GARISSA',
'KIRIAINI ', 'SYOKIMAU L', 'Machakos, Kathiani', 'Kitui',
'AUSTRALIA', 'Bondo', 'Salama ', 'bomet ', 'Naivasha ',
'TOORMINA, NSW', 'Utawala', 'Ankara', 'MOISBRIDGE', 'nbo',
'Kijabe', 'KABAZI', 'P.O BOX 75-30105 SOY', 'kisumu', 'nyeri',
'Voi', 'Kanjuku', "NG'IYA", 'Muranga', 'Bournemouth ',

'SARE-AWENDO', 'narok', 'Ruiru ', 'Kilifi ', 'Meru',
'Ol-joro-orok', '152 30100 eldoret ', 'Mtwapa', 'Karuri',
'Kitengela ', 'Isiolo ', '38061-00623 Nairobi', 'Ngong Hills',
'RARIEDA', ' Nairobi County,', '187 Ngewa', 'NJAMBINI',
'25635 00603 Nairobi', 'GIKOE', 'kakamega', 'ISHIARA ',
'P. O. Box 111 Ruiru', 'Kibwezi', 'Syokimau ', 'kiambu ', 'Bomet ',
'Busia', 'Gilgil', 'LAMU', 'Kajiado ', 'makueni ', 'kisii',
'MERU 2274', 'TURKANA', 'Kilifi', 'nottingham', 'Embakasi',
'Masii - MACHAKOS', 'SEREM', 'Tala, Kangundo. ', 'Ugunja',
'Karatina', 'Migori', 'Berlin', 'KALAMBA', 'UKUNDA ', 'ruiru',
'Sondu', 'BUSIA - PORT VICTORIA ', 'Matuu.', 'Gatundu',
'Seasons Kasarani. ', 'KARUNGU', 'Kahawa Sukari', 'Kangundo ',
'mombasa ', 'St. Georges', 'maseno', 'KINDARUMA ',
'kakuma 1 Nuer community ', 'copenhagen', '549 LIMURU', 'Lamu',
'DRESDEN', 'Ol Kalou ', 'KIRIA-INI ', '123 KIKUYU', 'Kianyaga',
'Kasarani', 'Makueni ', 'KABETE', 'Sagana',
'Imani avenue, Ongata Rongai, Kajiado', 'LOS ANGELES', 'KAKUMA',
'kitengela', '56-10300 Kerugoya', 'Kalimoni P.o Box 178-01001',
'Nairobi, Komarock.', 'kilifi', 'NAIROBI, KITENGELA ',
'14 Lamina Avenue, Mill Park VIC 3082', 'kebirigo ', 'Darwin ',
'Kericho ', 'Nairibi', '782 - 00232 Ruiru', 'eldoret', 'Oyugis',
'Conroe', 'NAIROBI 572 00300', 'Eldama Ravine', 'Kabete ', 'Maua',
'Isiolo', '828-90200', 'Athens', 'Chicago ', 'Hola', 'Abu Dhabi ',
'Nyayo Estate, Court 521', 'Deanside', '452-00200, KIAMBU',
'KOBENHAUN', 'karuri', 'Nanyuki ', 'P.O. Box 2413-60200, Meru',
'Kenol', 'Wote', 'githunguri ', 'NAIROBI. 45095-00100', 'Serem',
'P.o.box 1298-00618 Ruaraka, Nairobi', 'Vienna ', 'karen',
'CHICAGO', 'GENEVA', 'Kiserian ', 'Kabarnet', 'RABAI',
'KIAMBU COUNTY', 'Uthiru Cooperation ', 'Githunguri ',
'79718-00200 NAIROBI', 'stockholm', '42 NDARAGWA',
'P.O. Box 103635-00101', 'Heuchlingen ', 'PORT VICTORIA',
'Doolandella, Australia', 'Grand Oyster Kileleshwa, Nairobi',
'10572-00200 NAIROBI', 'Wangige ', 'Siaya', 'Litein', 'Washington',
'Geneva', 'Winchester ', 'Mariashoni', 'Mwiki', 'Limuru ',
'Nairobi i', 'P O Box 898-00100 NAIROBI', 'KAWNGWARE NAIROBI ',
'rongai', 'Roysambu', 'NAIROBI UMOJA', 'kangema',
'P.O.Box 848-00208 NGONG', 'Braunschweig ', 'Wote MAKUENI',
'814_3 300 KAPSABET ', 'Box 2316 Machakos', 'Ruaraka', 'Nyahururu',
'Kenmore ', 'Melbourne ', '1711 Nyahururu', 'nanyuki',
'kawangware ', 'Narok', 'Githunguri', 'Mahindi ', 'Moncton',
'kERUGOYA', 'Homa Bay', 'Ukunda', 'Mukurweini ', 'Malindi ',
'narumoru', 'Juba', '01000-214 Thika', 'kabete ', 'Kajiado',
'NAROK ', 'Kapenguria ', 'Athi-river ', "murang'a ", 'Gem',
'nyahururu ', 'Gachie ', 'Kirinyaga', ' Nairobi ', 'CALIFORNIA',
'Bomet', 'molo', 'Chuka', 'Nairobi, Kenya.', 'ANTONY',
'wite-makueni', 'ugunja', 'Matunda ', 'KARORI', 'kikuyu ',
'Shinyalu ', 'Molo', 'KIPKAREN', 'KANGUNDO ', 'LUTON', 'FLORIDA',
'NGEWA', 'SUSWA', 'USA', "KENOL MURANG'A ", 'PUYALLUP WA USA',
'LAARE', 'P.O Box 30231 - 00100, Nairobi', 'Runyenjes',
'KENDUBAY ', 'Gigiri, Nairobi', 'GATUKUYU', 'BUNYALA', 'OKIA',
'3704 GPO NAIROBI ', 'RANGWE', 'KITUI KENYA', 'SAUDI ARABIA',
'JUJA KALIMONI', 'MSA', 'GAKUNYU', 'MIHARATI', 'KERUGOYA TOWN',
'RAGENGNI', 'mwingi ', 'NBI', 'IGOJI', 'KIMULOT', 'LONDIANI',
'CAMBRIDGE', 'KHAYEGA', 'NDUNYU NJERU ', '58-10205 Maragua',
'MALINDI ', 'Kasarani, Clay City', 'ATHI RIVER ', 'Sawagongo ',
'HELSINKI', 'GPO', 'BURNTFOREST', 'MATUNDA ', 'HARRISDALE',
'KARANDI', 'DUKHAN ', 'SONDU', 'EMBU, KENYA', 'NAIROBI KENYA',
'NANDI HILLS ', 'HAZELWOOD', 'MGANGE', 'LUCYSUMMER-NAIROBI',
'KUKUYU', 'NGONG ROAD', 'NYAYO STADIUM', 'Karen', 'makuyu',
'LOITIKITOK', 'Kenyenya', 'MUNICH ', 'tawa', 'FUNYULA',
'NOTTIGHAM', 'nanyuki ', 'Kenny Bay', 'MOMBSA', 'SHERWOOD',
'NDARAGWA', 'ongata rongai', 'BUMALA', 'MEITINGEN ', '80100',
'KOMBEWA', 'DOYSELDOR', 'NOTTINGHAM', 'VIENNA', 'KISERIAN KAJIADO',
'NAIROBI, KENYA', 'Mtwapa ', 'KEUMBU', 'Kerugoya Town', 'KAP ',
'UPPER HILL NAIROBI', 'MOI;S BRIDGE', 'ENTERPRISE ROAD',
'THIONGO-NAIROBI', 'KATHONZWENI', 'UTAWALA', 'MIU ATHI-RIVER',
'CHAMAKANGA', 'NYAMACHE', 'NUERI', 'SULTAN HAMUD', 'NAROMORU ',
'NYAMIRA ', 'NAIROBI BURU', 'HAMBURG', 'TOWN ', '14 KADONGO 40223',
'Mois Bridge ', 'RUNYENJES ', 'MARAGWA', 'SURPRISE,USA',

```
       'KALAMBA ', 'Ruai', 'KADONGO', 'TORONGO', 'NJABINI',
       'GIGIRI, VILLAGE MARKET, NAIROBI', 'LAVINGTON', 'SPRING VALLEY',
       'MBUMBUNI', 'NAIROBI MUNICIPALITY', 'NAIROBI,KENYA',
       'Bamburi Mtambo ', 'GITHURAI', 'RUMURUTI',
       'Kenyatta Highway, Nanyuki', 'MUTOMO', 'KENYATTA HOSPITAL',
       'NAIROBI GENERAL POST OFFICE', 'LITIEN', 'LONDIANI ',
       'ONGATA RONGAI NAIROBI', 'GATUNDU ', 'GAMBONI', 'NAIROB',
       'L.KABETE', 'ATHI - RIVER', 'kerugoya ', 'Webuye ', 'Maralal',
       'Andover', 'mombas', 'Kiambu 627-00900', 'Maragua',
       'murang*a 10200-658', 'Syracuse', 'Karen Nairobi ',
       'ontario, California 91764 ', 'Karlsruhe ', 'Nyamira ', 'Olkalau',
       'GILGIL, KENYA', 'Kerugoya ',
       '14 AV FERNAND FENZY92160  ANTONY FRANCE'], dtype=object)
```

In [294…

```python
#checking the unique values from the relationship column
df['relationship'].unique()
```

Out[294…

```
array(['Partner', 'Sister', 'Husband ', 'Husband', 'Daughter', 'Brother ',
       'Son', 'Daughter ', 'Wife', 'Father', 'Mother', 'Brother', 'son',
       'mother', 'father', 'Mother ', 'SPOUSE', 'FRIEND', 'Spouse ',
       'WIfe', 'Father ', 'Spouse', 'child', nan, 'Parent', 'wife',
       'Wife ', 'Son ', 'spouse', 'doughter', 'CHILD', 'ParentChild',
       'partner', 'brother', 'sister', 'MotheR', 'Friend', 'Child',
       'Cousin', 'Dad', 'Mum', 'MOTHER', 'WIFE', 'Sister ', 'SON',
       'BROTHER', 'FATHER', 'aunt ', 'Grandmother', 'HUSBAND', 'self',
       'husband', 'husband ', 'Owner', ' Grand daughter', 'Nephew',
       'SP0USE', 'Self', 'parent', 'SISTER', 'NIECE', 'Niece', 'daughter',
       'Sibling', 'DAUGHTER', 'AUNT', 'brother ', '254701361835',
       'Parent ', 'daughter ', 'FAITHER', 'Family ', 'SPOUSE ',
       'Sibling ', 'PARENT', 'CONFIDANT', 'MUM', 'DAD', 'wife ', 'family',
       'Trustee ', 'son ', 'mother ', 'MOTHER ', 'friend', 'spouse ',
       'S0N', 'SISTER ', 'FATHER ', 'father ', 'FAMILY ', 'WifE',
       'Partner ', 'daugther', '1226805243;130143346', 'relative',
       'BROTHER ', 'Daighter', 'Advisor', 'SON ', 'AUNTIE', 'FIANCEE',
       'Fiancee', 'DOUGHTER', 'parent ', 'HUSBAND ', 'DAUGHTER ', 'SELF',
       'MOther', 'mOTHER', 'Aunt', 'Grandparent', 'Bother ', 'SO',
       'married', 'MARRIED', 'FIANCE', 'Doughter ', 'Guardian ',
       'ASS0CIATE', 'sibling', 'fiance', 'Sistet', 'Grandson', 'Doughter',
       'DAUGTHER', 'wIFE', 'Bro', 'PARTNER', 'B INLAW', 'GUARDIAN',
       'daughte', 'Mom', 'COUSIN', 'GRANDMA', 'aunt', 'Domestic Partner',
       'PATNER', 'me', 'MENTOR', 'UNCLE', 'Sis', 'SiBLING', 'GRAND SON',
       'Fiance ', 'KID', 'SIBLING', 'GRANDSON', 'Nephew ', '-', 'Cousin ',
       'DAUGHETR', 'Person', 's0n', 'na', 'myson', 'PARTNER ', 'Bishop',
       'nephew', 'mwas', 'Mothers', 'Friend ', 'Child ', 'daughter33',
       'CLOSE FRIEND', 'hUSBAND', 'CLOSEFRIEND', 'Boyfriend ',
       'dependant', 'WIFE ', 'MARRIAGE', 'GRAND DAUGHTER', 'DAUGHER',
       'Niece ', 'granddaughter', 'grandson', 'DAUGHTER/CHILD',
       'RELATIVE', 'mum', 'single', 'Self ', 'sister ', 'Mother TO SON',
       'Grandfather', 'DAD ', 'Relative', 'Daughter-in-law ',
       'GRANDMOTHER', 'child ', '15', 'DaD', 'GIRLFRIEND', 'FAMILY',
       'Uncle ', 'SiNGLE', 'SINGLE', 'fiancee', 'FaMILY', 'BENEFICIARY',
       'OWNER', 'Relative ', 'GRANDDAUGHTER', ' Spouse', 'uncle',
       'Married ', 'chil', 'CHILD MINOR', '33', 'NEPHEW', 'JOINT PARTNER',
       'MOTHERS ', 'Fiancé ', 'Others ', 'Grand daughter',
       'DAUGHTER-IN-LAW', 'Na', 'CHILD ', 'SPIRITUAL FATHER', 'COUSN',
       'DEPEDANT', 'Estate', 'Uncle', 'sISTER', 'sisiter', ' BROTHER',
       'Granddaughter ', 'BRother', 'Aunt ', 'cousin', 'Baby', 'Morher',
       'MOM', 'God mother', 'GRANDSON ', 'Husband/spouse', 'SPOUCE',
       'uncle ', 'FRIEND ', 'Single', 'Vhild', 'Patner', 'Spouce',
       'Family', 'SoN', 'SisteR', 'AUNTY', 'FATHER-IN-LAW', 'SonS', '0',
       'MO9', 'Fiance', '33.4', 'CHILF', '254727756559', '0720401466',
       'GRAND FATHER', 'PRINCIPAL', 'Guardian', 'JUNIOR',
       'DGRAND DAUGHTER', 'COWORKER', 'GRAND MOTHER', 'D100', 'NMO',
       'BUSINESS PARTNER', 'GURDIAN', '2021', 'MotH', 'MoM', 'UN,CLE',
       'GRAND CHILD', 'Sister in law', 'Grand Daughter', 'COMAPANION',
       'BOYFRIEND', 'EX-WIFE', 'GRANDDAUGTER', 'PARNT', 'M', 'F', 'Chils',
       ';SON', 'GUARDINA', 'NiecE', 'Brother-IN-LAW', 'Member',
       'Grand mother', 'ADMINISTRATOR', 'VM', 'GRANDFATHER', 'GIRLFRIED',
       'Brestfriend', 'Finance', 'SIS', 'Daughter in law', 'children',
```

```
       'Girlfriend', 'COLLEAGUE', 'TRUSTEE', '254708368426', 'Colleague',
       'Sister-IN-LAW', 'GRAND-DAUGHTER', 'GRANNY', 'GRANDCHILD', "'M",
       'spouce', 'SiLBLING', 'Siblings', 'Step-mother', 'Aunty', 'FREINF',
       '50', '100', '254723993458', 'Cousins', 'Cousins ', 'ASSOCIATE',
       'Fiane', 'guardian', 'Guadian', 'Confidant', 'Auntie', 'GRANDPA',
       'mom', 'Brother in Law', 'BUSINESS PARTNER ', 'Spuose',
       'Business Partner', 'Dating', 'IN LAW', 'STEP-MOM', 'Son/Brother',
       'Chid', 'Granddaughter', 'niece', 'Son/SISTER', 'Grand Son',
       'BrotheR', "Nolan's Guardian", "Xena's Guardian", 'Sister-in law',
       'PARENTS', 'c/o', 'STEP MOTHER', 'NiCE', 'Ciru', 'DEPENDANT',
       'som', 'GUARDIAN ', 'Dau`', 'Kid', 'PARENT ', 'CHILDREN',
       'partner ', 'CLOUSIN', '19112004', 'twinsister', 'MINOR',
       'Sister- Custodian', 'GRANSON', 'COMPANION', 'ENGAGED',
       'WELL WISHER', 'PARNER', 'GRANCHILD', 'Trustee', 'Next of kin',
       'SIIBLING', 'SIBLINGS', 'Foster daughter', 'ADMINSTRATOR',
       'IN-LAW', 'Boyfriend', 'NepheW', 'EMPLOYEE', 'DIRECTORS',
       'RELIGIOUS COMMUNITY', 'CUSTODIAN', 'C/O ARIANNA', 'Sister IN LAW',
       'spose', 'SPAUSE', 'MEMBER', 'DomesticPartner', 'Grand-daughter',
       'Brother-in-law', 'STEP-BROTHER', 'Mother to Evan', 'Sister-Inlaw',
       'BUS. PARTNER', 'Sister(Guardian)', 'CONTACT PERSON', 'FatheR',
       'Brother in law', 'Parwnt', 'M,', 'Sister-GUARDIAN', 'Grand Child',
       'GODMOTHER', 'Sister-Guardian', 'Mother- Guardian', 'Grandchild',
       'Grand son', 'CHLD', 'GRAND.DAUGHTER', 'GRAND-MOTHER', 'Parents',
       'EX WIFE', 'Sister - Guardian', 'Step-Mother', 'FOSTER DAUGHTER',
       'Grand-mother', 'God Mother', 'Mother-in-law', 'MEMBER ',
       'B/S PARTNER', 'LAWYER', 'GRAND-SON', 'parents', 'Stepson',
       'Contingent', 'grandmother', 'G/Daughter', 'Brother and sister',
       'Sibbling', ' ', 'COUSIN ', 'DaughteR', 'SPUOSE', 'CHIILD',
       'DSPOUSE', 'Executor', 'HU', 'EWIFE', 'SPOUDSE', 'SoPOUSE',
       'grandchild', 'ChILD', '9266025', 'PARE NT', '254700442162',
       'BEST FRIEND', 'SISTER IN LAW ', 'mather', 'SAVING PARTNER ',
       'AUNTIE ', 'ParenChild', 'GIRLFRIEND ', 'SINGLE ', 'Parentchild',
       'parentchild', 'SPOUSE20', 'Myself', 'FIACEE', 'SD',
       'Grand mother ', 'Mother of child', 'Childnephew', 'FATH ',
       'MY SON', 'BabyMother', 'CHID', 'GIRL FRIEND', 'chils', 'self ',
       "Mother's ", 'Mum ', 'aunty', 'SPouse', 'Auntie ', 'ChiLD',
       'grandmother ', 'Grand daughter ', 'auntie', 'baby mama ', 'DA',
       'BrOther', 'Fiancee ', 'In law', 'Colleague ', 'WiFE', 'owner',
       'sON', 'Mather', 'Dauhgter', 'myself', 'Hubby', 'Sister in-law',
       'Aunty ', 'Foster parent', 'grandson ', 'Wife/Guardian',
       '21761402', 'GUARDIAN-MOTHER'], dtype=object)
```

In [295…

```python
#checking the unique values from the hse_no column
df['hse_no'].unique()
```

Out[295…

```
array(['Single Member'], dtype=object)
```

In [296…

```python
#checking the unique values from the portfolio column
df['portfolio'].unique()
```

Out[296…

```
array(['Money Market', 'Dollar Fund', 'Wealth Fund', 'Equity Fund',
       'Fixed Income', 'Balanced Fund', 'MoneyMarket', nan], dtype=object)
```

In [297…

```python
#checking the unique values from the town column
df['town'].unique()
```

Out[297…

```
array(['NAIROBI', nan, 'NAIROBI ', 'THIKA', 'NAKURU', 'MOMBASA ',
       'NDANAI ', 'KISUMU', 'KIKUYU', 'JUJA', 'MACHAKOS', 'MERU',
       'KERICHO', 'MUKURWEINI', 'KATANGI', 'THIKA,KIAMBU', 'NANYUKI',
       'RUAI ', 'MALINDI', 'NYAHURURU', 'MUMIAS', 'KAKAMEGA', 'LODWAR',
       'Nairobi ', 'KITENGELA', 'LIMURU', 'THIKA ', 'MIGORI', 'ELDORET',
       'KITUI', 'RUARAKA', 'Ruiru', 'NJORO', 'EMBU', 'SORI', 'NGONG',
       'MASENO', 'GATUNDU', 'KAHURO', 'KILIFI', 'RUIRU', 'KERUGOYA',
       'KARATINA', 'KITALE ', 'MOMBASA', 'BUNGOMA', 'KISII', '-',
       'Nairobi', 'KINAMBA', 'SIAYA', 'KERICHO ', 'KIAMBU', 'KITALE',
       'BUSIA', 'RUNYENJES', 'VILLAGE MARKET', 'Kapsabet', 'KAGWE',
       'NANDI HILLS', 'KUTUS', 'DAGORETTI', 'LITEIN', 'KAJIADO', 'CHUKA',
       'NYERI', 'NAIROBI, NAIROBI', 'Kisumu ', 'KANJUKU', 'EMALI',
```

'GILGIL', 'Kiambu ', 'KANGEMA', 'TIRIKI', 'KARATINA ', 'OTHAYA',
'KARURI ,KIAMBU', 'MURANGA ', 'Mombasa', 'GATUKUYU ', 'MLOLONGO',
'UTAWALA ', 'KAPENGURIA', 'KAPSABET', 'SUNA', 'BOMET', 'KENOL ',
'NYERI ', 'BUNGOMA ', 'NDANAI', 'KALIMONI', 'KIGANJO',
'KAPSOKWONY', 'Nakuru', 'KIAMBU ', 'nairobi', 'CHOGORIA',
'GITHUNGURI', 'YOANI,SALAMA', 'LIMURU ', 'NAIROBI, KENYA.',
'UKUNDA', 'MATUU', 'embu', 'NYALI', 'nakuru', 'MBALE', 'EMUHAYA',
'TELFORD', 'NAIVASHA', 'KAREN', 'KILIFI ', 'HOLA', 'MAKINDU',
'WOTE', 'DIANI', 'O', 'OLKALAU', 'KAREN ', 'UPLANDS', 'WANGÚRU',
'KABIYET', 'KAKAMEGA ', 'SOTIK', 'NGONG ', 'CITY SQUARE', 'NKUBU',
'KANGARI', 'NGONG HILLS ', 'NYAHURURU ', 'KARURI ', 'UTHIRU',
'MARIMANTI', 'MANCHESTER', 'Kikuyu', '.', 'NAIRO ', 'kiambu',
'VOI', 'EMBAKASI', 'MOLO', 'Siaya ', 'NGIYA', 'OLKALOU',
'KANYAGIA', 'WANJOHI', 'kabete', 'Wamunyu', 'MARIAKANI',
'KISERIAN', 'ARTHI RIVER ', 'SUBUKIA', 'KILGORIS', 'KIMININI',
'LAGOS', 'SARE', 'ATHI RIVER', 'ngong hills ', 'WERUGHA',
'NAIROBIP', 'MASINFA', 'kikuyu', 'OL JORO OROK', 'MAUA',
'GIKUE MURANGA', 'KIMENDE', 'KIMILILI', 'KISUMU ', 'VIHIGA',
' NAIROBI ', 'BURUBURU', 'MATATHIA', 'RUAKA', 'UGUNJA', 'LONGISA',
'ELDORATE ', 'KAPKATET', 'WINNIPEG', 'PIPELINE ', 'KAHAWA WEST ',
'KANDARA', 'Kimilili ', 'MTWAPA ', 'MARAGUA', 'BANANA ', 'NYAMIRA',
'AUBURN', 'KARURI', 'MTWAPA', 'ONGATA RONGAI', 'ELDORET ',
'HOMABAY', 'WANGIGE', 'KENDUBAY', 'KIBWEZI', 'RONGAI', 'RUIRU ',
'NORTH KINANGOP', 'MAKUYU', 'Nyeri ', 'Eldoret ', 'BOUTIGNY',
'00506', 'MWINGI', '00100', 'KERUGOYA ', 'TALA', 'OLENGURUONE',
'MAVINDINI', 'UHURU GARDENS', 'KIKUYU ', 'Ongata Rongai',
'SOUTH KINANGOP', 'ELDAMA RAVINE', 'MITUNGURU', 'TOM MBOYA',
'NAKURU ', 'MITABONI', 'NAROMORU', 'MURANGA', 'OLJOROROK',
'BUTERE', 'OYUGIS', 'SILIBWET', 'EMBU ', 'Doha', 'Eldoret',
'KANGEMI', 'NYERI-KIGUMO', 'KISERIAN ', 'Athi River', 'KENOL',
'OL KALOU', 'WESTLANDS', 'Nyahururu ', 'ENDARASHA ', 'GPO/NAIROBI',
'DAGORETI', 'OGEMBO ', 'mombasa', 'RUAI', 'DONYO SABUK', 'KEROKA',
'LOWER KABETE', 'MWEIGA', 'Naitobi', 'TOKYO', 'Embu', 'KIRINYAGA',
'KIANYAGA', 'SIAYA ', 'ADLISWIL', 'BURU BURU', 'MERU ', 'NYAKIO',
'KWALE', 'AMUKURA', 'NAROK', 'KOBUJOI', "WANG'URU", 'BONDO',
'Kericho', 'KIKIMA ', 'CHANGAMWE', 'BUTULA', 'NAIROBI KAYOLE',
'KANGUNDO', 'MARIAKANI ', 'TIMBOROA', 'KIANJAI', 'KALOLENI ',
'ISINYA', 'IOWA CITY, IOWA', 'Garsen ', 'MUKURWE-INI',
'NAIROBI GPO', 'MANYATTA', 'SARE-OWENDO', 'NANYUKI ', 'NDHIWA',
'ELBURGON', 'Orange', 'MAKUENI', 'kisumu ', 'MACHAKOS ',
'KABARNET ', "Murang'a ", 'JUJA, KIAMBU', 'Bungoma', 'MAWEGO',
'Bath', 'ROYSAMBU', 'Nanyuki', 'SAGANA', 'MACH ', 'Nyeri', 'Ruaka',
'KITUI ', 'KAJIADO ', 'NUNGUNI', 'RONGAI ', 'KISII ', 'WANGIGE ',
'Makongeni Jogoo Rd', 'nairobi ', 'Kikuyu ', 'Kisii', 'MSAMBWENI',
'BUJUMBURA', 'GITHONGO', 'KEROKA/KISII', 'BURUBURU, NAIROBI',
'MALAVA', 'KIJABE', 'Woolwich ', 'NDERE-SIAYA', 'AHERO',
'MUHORONI ', 'Nakuru ', 'kahuro', 'Malakisi ', 'MUHORONI',
'ISIOLO', 'SYOKIMAU ', 'ONGATA RONGAI ', 'Maralal ',
'KIBIRICHIA, MERU', 'GRANGER, INDIANA', 'WESTLAND ', 'MUKURWEINI ',
'MASII', 'NRB', 'KWISERO', 'ANKARA', 'SHIMBA HILLS', 'TAVETA',
'WOTE, MAKUENI ', 'WANGURU', 'KISSII', 'BURUBURU NAIROBI ',
'KIRITIRI', 'MESA ARIZONA', 'KAJIADO TOWN', 'KURIA',
'NAIROBI WEST', 'ENDARASHA', 'KIPKAREN ', '0', 'GAMBOGI', 'KOLA',
'Marsabit', 'WEBUYE', 'GAKUNGU', 'NAIROBJ',
'NAIROBI ,KAMITI ROAD RICHLAND', 'NKUBU ', 'MWATATE', 'MALAKISI',
'ITEN', 'KATHIANI', "MURANG'A", 'KAIRIA', 'UPLANDS ', 'RONGO',
'UHURU GARDEN', 'KINANGOP', 'BAHATI NAKURU ', 'MATILIKU', 'meru',
'Limuru', 'MALAA', 'TURBO', 'MARIMA -MERU HIGHWAY ', 'OLJORO OROK',
'homabay ', 'KIKIMA', 'ADAMS ARCADE', 'MUNICH', 'MUMIAS ',
'KALIMONI ', 'CHWELE', 'MARALAL', 'NAMBALE', 'NAIROBI COUNTY',
'Narok ', 'TURBO ', 'KANGARI ', 'Pécs ', 'thika', 'KATSE',
'thika ', 'CHEBUNYO', 'OL JORO OROK ', 'MOBIL PLAZA', 'WAMUNYU ',
'MAGUNGA', 'RORET', 'YALA', 'KAHUHIA', "KENOL MURANG'A",
'Machakos', 'UKWALA', 'KALOLENI', 'JOONDALUP', 'MIGORI ',
'MARAGOLI', 'MARMANET', 'IKANGA', 'KINOO', 'OYUGIS ', 'LANGATA',
'Kiambu', 'Malindi', 'NAIROBI.', 'MAKUENI ', 'Bungoma ',
'GITHUNGURI ', 'MACHOKOS', 'KOMBEWA ', 'UTHIRU ', 'KITENGELA ',
'BANANA', 'CHEPKORIO', 'MASII ', 'Gitugi', '. ', 'RUIRY', 'Kisumu',
'MBUMBUNI-MACHAKOS', 'WAMUYU', 'siaya', 'Kakamega', 'TAITA TAVETA',

'Bumala', 'KIMENDE ', 'BRISTOL', 'MATUNDA', 'KARABA', 'SULTAN',
'GATURA', 'KASARANI ,NAIROBI ', 'NANDI', 'MAGUTUNI', 'CHUKA ',
'NGONG HILL', 'ngong', 'KIANGAI', 'MBITA', 'Mumias', 'KAMBITI',
'MUMBI', 'Juja', 'WODANGA', 'KIRIANI', 'KABARNET', 'MISIKHU',
'Thika', 'NYANSIONGO', 'nyeri ', 'NAIVASHA ', "MURANG'A ",
'KAPCHENO', 'NGARA', 'MADARAKA', 'kitengela ', 'BONDO ', 'WAMUNYU',
'NYANDARUA', 'MOGOGOSIEK', '00', 'NAIROBI/JUJA', 'NA ', 'LUANDA',
'KLEMZIG', 'TEL AVIV', 'ongata rongai ', '30-60101 Manyatta ',
'12410 Nyeri', 'KATHONZWENI ', 'GITHURAI ', 'WESRN AUSTRALIA',
'SABASABA ', 'Gilgil ', 'ISEBANIA', 'KIRIAINI', 'MUTHETHENI ',
'Mombasa ', 'Kitengela', 'CARLIFONIA', 'NIL',
'NAIROBI, NAIROBI AREA, KENYA', 'SOY', 'Sabasaba', 'SABASABA',
'COLUMBIA', 'Homa Bay ', 'MARIMA', 'SEGA', 'KAYOLE', 'BUDD LAKE',
'KITHYOKO', 'KIJABE ', 'Mbale', 'KITHIMANI', 'HANOI', 'chuka',
'KILINDINI', 'MERY', 'CHELMSFORD', 'LOITOKITOK', 'MOSORIOT',
'IKONGE', 'ENGINEER', 'Othaya', 'Ahero', 'MWEA', 'JOSKA ',
'NGAMBWA', 'MASINGA', 'KINANGOP ', 'LONDON', 'KILELESHWA ',
'SARIT CENTRE', 'mgange ', 'ST. MARYS NSW', 'GESIMA', 'Migori ',
'KHWISERO', 'SYOKIMAU', 'MTITO ANDEI ', 'nakuru ', 'NAIROBI CITY',
'ATHIRIVER', 'KABATI', 'Kitale', 'MPEKETONI', 'YOANI',
'KABARTONJO', 'NAMANGA', 'CHESINENDE', 'BAHATI', 'NZEEKA',
'MATATHIA, KAMAE', 'RONALD NGALA ', 'KUBUKUBU', 'THOGOTO',
'GATORA', 'NAI ', 'KUJA,KIAMBU', 'Machakos ', 'AMAGORO', 'Uthiru',
'Naivasha', 'KEROKA ', 'MAZERAS', 'EGERTON', 'KASARANI', 'Kimende',
'WAJIR', 'DUBAI', 'TWO RIVERS', 'NDARAGWA ', 'EASTLEIGH', 'KAGIO',
'HOMA BAY', 'KISII, KENYA', 'Thika ', 'NGECHA', 'Kakamega ',
'KERINGET', 'FORT WAYNE', 'STAREHE', 'GILGIL ', 'KAWANGWARE',
'GPO NAIROBI', 'GITUGI', 'Tala', 'kakamega ', 'TEXAS', 'GACHIE',
'Syokimau', 'Sydney ', 'LESSOS', 'Meru ', 'KIPKAREN RIVER',
'SOSIOT', 'LAKEWOOD SEATTLE', 'Lodwar', 'Kitale ', 'Membley',
'chepkorio', 'Luanda', 'Kijabe ', 'TONGAREN', 'karuri ', 'malindi',
'Ogembo ', 'Maryland', 'Kisii Town ', 'kajiado ', 'LIKONI',
'wangige ', 'Ngong', 'OLKALOU ', '4597-20100 Nakuru', 'GARISSA',
'KIRIAINI ', 'SYOKIMAU L', 'Machakos, Kathiani', 'Kitui',
'AUSTRALIA', 'Bondo', 'Salama ', 'bomet ', 'Naivasha ',
'TOORMINA, NSW', 'Utawala', 'Ankara', 'MOISBRIDGE', 'nbo',
'Kijabe', 'KABAZI', 'P.O BOX 75-30105 SOY', 'kisumu', 'nyeri',
'Voi', 'Kanjuku', "NG'IYA", 'Muranga', 'Bournemouth ',
'SARE-AWENDO', 'narok', 'Ruiru ', 'Kilifi ', 'Meru',
'Ol-joro-orok', '152 30100 eldoret ', 'Mtwapa', 'Karuri',
'Kitengela ', 'Isiolo ', '38061-00623 Nairobi', 'Ngong Hills',
'RARIEDA', ' Nairobi County,', '187 Ngewa', 'NJAMBINI',
'25635 00603 Nairobi', 'GIKOE', 'kakamega', 'ISHIARA ',
'P. O. Box 111 Ruiru', 'Kibwezi', 'Syokimau ', 'kiambu ', 'Bomet ',
'Busia', 'Gilgil', 'LAMU', 'Kajiado ', 'makueni ', 'kisii',
'MERU 2274', 'TURKANA', 'Kilifi', 'nottingham', 'Embakasi',
'Masii - MACHAKOS', 'SEREM', 'Tala, Kangundo. ', 'Ugunja',
'Karatina', 'Migori', 'Berlin', 'KALAMBA', 'UKUNDA ', 'ruiru',
'Sondu', 'BUSIA - PORT VICTORIA ', 'Matuu.', 'Gatundu',
'Seasons Kasarani. ', 'KARUNGU', 'Kahawa Sukari', 'Kangundo ',
'mombasa ', 'St. Georges', 'maseno', 'KINDARUMA ',
'kakuma 1 Nuer community ', 'copenhagen', '549 LIMURU', 'Lamu',
'DRESDEN', 'Ol Kalou ', 'KIRIA-INI ', '123 KIKUYU', 'Kianyaga',
'Kasarani', 'Makueni ', 'KABETE', 'Sagana',
'Imani avenue, Ongata Rongai, Kajiado', 'LOS ANGELES', 'KAKUMA',
'kitengela', '56-10300 Kerugoya', 'Kalimoni P.o Box 178-01001',
'Nairobi, Komarock.', 'kilifi', 'NAIROBI, KITENGELA ',
'14 Lamina Avenue, Mill Park VIC 3082', 'kebirigo ', 'Darwin ',
'Kericho ', 'Nairibi', '782 - 00232 Ruiru', 'eldoret', 'Oyugis',
'Conroe', 'NAIROBI 572 00300', 'Eldama Ravine', 'Kabete ', 'Maua',
'Isiolo', '828-90200', 'Athens', 'Chicago ', 'Hola', 'Abu Dhabi ',
'Nyayo Estate, Court 521', 'Deanside', '452-00200, KIAMBU',
'KOBENHAUN', 'karuri', 'Nanyuki ', 'P.O. Box 2413-60200, Meru',
'Kenol', 'Wote', 'githunguri ', 'NAIROBI. 45095-00100', 'Serem',
'P.o.box 1298-00618 Ruaraka, Nairobi', 'Vienna ', 'karen',
'CHICAGO', 'GENEVA', 'Kiserian ', 'Kabarnet', 'RABAI',
'KIAMBU COUNTY', 'Uthiru Cooperation ', 'Githunguri ',
'79718-00200 NAIROBI', 'stockholm', '42 NDARAGWA',
'P.O. Box 103635-00101', 'Heuchlingen ', 'PORT VICTORIA',

```
'Doolandella, Australia', 'Grand Oyster Kileleshwa, Nairobi',
'10572-00200 NAIROBI', 'Wangige ', 'Siaya', 'Litein', 'Washington',
'Geneva', 'Winchester ', 'Mariashoni', 'Mwiki', 'Limuru ',
'Nairobi i', 'P O Box 898-00100 NAIROBI', 'KAWNGWARE NAIROBI ',
'rongai', 'Roysambu', 'NAIROBI UMOJA', 'kangema',
'P.O.Box 848-00208 NGONG', 'Braunschweig ', 'Wote MAKUENI',
'814_3 300 KAPSABET ', 'Box 2316 Machakos', 'Ruaraka', 'Nyahururu',
'Kenmore ', 'Melbourne ', '1711 Nyahururu', 'nanyuki',
'kawangware ', 'Narok', 'Githunguri', 'Mahindi ', 'Moncton',
'kERUGOYA', 'Homa Bay', 'Ukunda', 'Mukurweini ', 'Malindi ',
'narumoru', 'Juba', '01000-214 Thika', 'kabete ', 'Kajiado',
'NAROK ', 'Kapenguria ', 'Athi-river ', "murang'a ", 'Gem',
'nyahururu ', 'Gachie ', 'Kirinyaga', ' Nairobi ', 'CALIFORNIA',
'Bomet', 'molo', 'Chuka', 'Nairobi, Kenya.', 'ANTONY',
'wite-makueni', 'ugunja', 'Matunda ', 'KARORI', 'kikuyu ',
'Shinyalu ', 'Molo', 'KIPKAREN', 'KANGUNDO ', 'LUTON', 'FLORIDA',
'NGEWA', 'SUSWA', 'USA', "KENOL MURANG'A ", 'PUYALLUP WA USA',
'LAARE', 'P.O Box 30231 - 00100, Nairobi', 'Runyenjes',
'KENDUBAY ', 'Gigiri, Nairobi', 'GATUKUYU', 'BUNYALA', 'OKIA',
'3704 GPO NAIROBI ', 'RANGWE', 'KITUI KENYA', 'SAUDI ARABIA',
'JUJA KALIMONI', 'MSA', 'GAKUNYU', 'MIHARATI', 'KERUGOYA TOWN',
'RAGENGNI', 'mwingi ', 'NBI', 'IGOJI', 'KIMULOT', 'LONDIANI',
'CAMBRIDGE', 'KHAYEGA', 'NDUNYU NJERU ', '58-10205 Maragua',
'MALINDI ', 'Kasarani, Clay City', 'ATHI RIVER ', 'Sawagongo ',
'HELSINKI', 'GPO', 'BURNTFOREST', 'MATUNDA ', 'HARRISDALE',
'KARANDI', 'DUKHAN ', 'SONDU', 'EMBU, KENYA', 'NAIROBI KENYA',
'NANDI HILLS ', 'HAZELWOOD', 'MGANGE', 'LUCYSUMMER-NAIROBI',
'KUKUYU', 'NGONG ROAD', 'NYAYO STADIUM', 'Karen', 'makuyu',
'LOITIKITOK', 'Kenyenya', 'MUNICH ', 'tawa', 'FUNYULA',
'NOTTIGHAM', 'nanyuki ', 'Kenny Bay', 'MOMBSA', 'SHERWOOD',
'NDARAGWA', 'ongata rongai', 'BUMALA', 'MEITINGEN ', '80100',
'KOMBEWA', 'DOYSELDOR', 'NOTTINGHAM', 'VIENNA', 'KISERIAN KAJIADO',
'NAIROBI, KENYA', 'Mtwapa ', 'KEUMBU', 'Kerugoya Town', 'KAP ',
'UPPER HILL NAIROBI', 'MOI;S BRIDGE', 'ENTERPRISE  ROAD',
'THIONGO-NAIROBI', 'KATHONZWENI', 'UTAWALA', 'MIU ATHI-RIVER',
'CHAMAKANGA', 'NYAMACHE', 'NUERI', 'SULTAN HAMUD', 'NAROMORU ',
'NYAMIRA ', 'NAIROBI BURU', 'HAMBURG', 'TOWN ', '14 KADONGO 40223',
'Mois Bridge ', 'RUNYENJES ', 'MARAGWA', 'SURPRISE,USA',
'KALAMBA ', 'Ruai', 'KADONGO', 'TORONGO', 'NJABINI',
'GIGIRI, VILLAGE MARKET, NAIROBI', 'LAVINGTON', 'SPRING VALLEY',
'MBUMBUNI', 'NAIROBI MUNICIPALITY', 'NAIROBI,KENYA',
'Bamburi Mtambo ', 'GITHURAI', 'RUMURUTI',
'Kenyatta Highway, Nanyuki', 'MUTOMO', 'KENYATTA HOSPITAL',
'NAIROBI GENERAL POST OFFICE', 'LITIEN', 'LONDIANI ',
'ONGATA RONGAI NAIROBI', 'GATUNDU ', 'GAMBONI', 'NAIROB',
'L.KABETE', 'ATHI - RIVER', 'kerugoya ', 'Webuye ', 'Maralal',
'Andover', 'mombas', 'Kiambu 627-00900', 'Maragua',
'murang*a 10200-658', 'Syracuse', 'Karen Nairobi ',
'ontario, California 91764 ', 'Karlsruhe ', 'Nyamira ', 'Olkalau',
'GILGIL, KENYA', 'Kerugoya ',
'14 AV FERNAND FENZY92160  ANTONY FRANCE'], dtype=object)
```

In [298…

```
#inspecting the dataframe for null values
df.isna().sum()
```

Out[298…

```
member_no                 0
reg_date               1863
dob                   20038
hse_no                    0
gender                 4667
town                1048216
relationship        1632933
beneficiery_dob     1086598
portfolio                 9
dtype: int64
```

3.0 DATA CLEANING AND PREPARATION

In [299…

```
#inspecting the dataframe duplicate values
df.duplicated().sum()
```

Out[299…

```
7407066
```

We have 7407066 duplicated rows in our dataset.

In the below code block, we drop duplicated values.

In [300…

```
df = df.drop_duplicates()
```

From our problem statement, we aim to cross sell investment products depending on Age, Gender and Location. Therefore, we are going to drop some columns which are irrelevant. e.g. hse_no, beneficiary_dob,reg_date.

In [301…

```
df=df.drop(columns=['reg_date','hse_no'])
```

In [302…

```
#inspecting the null vaues after dropping columns
df.isna().sum()
```

Out[302…

```
member_no              0
dob                  238
gender                82
town               15026
relationship       14308
beneficiery_dob    13937
portfolio              4
dtype: int64
```

In [303…

```
#checking for unique values in relationship column
df.relationship.unique()
```

Out[303…

```
array(['Partner', 'Sister', 'Husband ', 'Husband', 'Daughter', 'Brother ',
       'Son', 'Daughter ', 'Wife', 'Father', 'Mother', 'Brother', 'son',
       'mother', 'father', 'Mother ', 'SPOUSE', 'FRIEND', 'Spouse ',
       'WIfe', 'Father ', 'Spouse', 'child', nan, 'Parent', 'wife',
       'Wife ', 'Son ', 'spouse', 'doughter', 'CHILD', 'ParentChild',
       'partner', 'brother', 'sister', 'MotheR', 'Friend', 'Child',
       'Cousin', 'Dad', 'Mum', 'MOTHER', 'WIFE', 'Sister ', 'SON',
       'BROTHER', 'FATHER', 'aunt ', 'Grandmother', 'HUSBAND', 'self',
       'husband', 'husband ', 'Owner', ' Grand daughter', 'Nephew',
       'SP0USE', 'Self', 'parent', 'SISTER', 'NIECE', 'Niece', 'daughter',
       'Sibling', 'DAUGHTER', 'AUNT', 'brother ', '254701361835',
       'Parent ', 'daughter ', 'FAITHER', 'Family ', 'SPOUSE ',
       'Sibling ', 'PARENT', 'CONFIDANT', 'MUM', 'DAD', 'wife ', 'family',
       'Trustee ', 'son ', 'mother ', 'MOTHER ', 'friend', 'spouse ',
       'S0N', 'SISTER ', 'FATHER ', 'father ', 'FAMILY ', 'WiFE',
       'Partner ', 'daugther', '1226805243;130143346', 'relative',
       'BROTHER ', 'Daighter', 'Advisor', 'SON ', 'AUNTIE', 'FIANCEE',
       'Fiancee', 'DOUGHTER', 'parent ', 'HUSBAND ', 'DAUGHTER ', 'SELF',
       'MOther', 'mOTHER', 'Aunt', 'Grandparent', 'Bother ', 'SO',
       'married', 'MARRIED', 'FIANCE', 'Doughter ', 'Guardian ',
       'ASS0CIATE', 'sibling', 'fiance', 'Sistet', 'Grandson', 'Doughter',
       'DAUGTHER', 'wIFE', 'Bro', 'PARTNER', 'B INLAW', 'GUARDIAN',
       'daughte', 'Mom', 'COUSIN', 'GRANDMA', 'aunt', 'Domestic Partner',
       'PATNER', 'me', 'MENTOR', 'UNCLE', 'Sis', 'SiBLING', 'GRAND SON',
       'Fiance ', 'KID', 'SIBLING', 'GRANDSON', 'Nephew ', '-', 'Cousin ',
       'DAUGHETR', 'Person', 's0n', 'na', 'myson', 'PARTNER ', 'Bishop',
       'nephew', 'mwas', 'Mothers', 'Friend ', 'Child ', 'daughter33',
       'CLOSE FRIEND', 'hUSBAND', 'CLOSEFRIEND', 'Boyfriend ',
       'dependant', 'WIFE ', 'MARRIAGE', 'GRAND DAUGHTER', 'DAUGHER',
       'Niece ', 'granddaughter', 'grandson', 'DAUGHTER/CHILD',
       'RELATIVE', 'mum', 'single', 'Self ', 'sister ', 'Mother TO SON',
       'Grandfather', 'DAD ', 'Relative', 'Daughter-in-law ',
       'GRANDMOTHER', 'child ', '15', 'DaD', 'GIRLFRIEND', 'FAMILY',
```

```
       'Uncle ', 'SiNGLE', 'SINGLE', 'fiancee', 'FaMILY', 'BENEFICIARY',
       'OWNER', 'Relative ', 'GRANDDAUGHTER', ' Spouse', 'uncle',
       'Married ', 'chil', 'CHILD MINOR', '33', 'NEPHEW', 'JOINT PARTNER',
       'MOTHERS ', 'Fiancé ', 'Others ', 'Grand daughter',
       'DAUGHTER-IN-LAW', 'Na', 'CHILD ', 'SPIRITUAL FATHER', 'COUSN',
       'DEPEDANT', 'Estate', 'Uncle', 'sISTER', 'sisiter', ' BROTHER',
       'Granddaughter ', 'BRother', 'Aunt ', 'cousin', 'Baby', 'Morher',
       'MOM', 'God mother', 'GRANDSON ', 'Husband/spouse', 'SPOUCE',
       'uncle ', 'FRIEND ', 'Single', 'Vhild', 'Patner', 'Spouce',
       'Family', 'SoN', 'SisteR', 'AUNTY', 'FATHER-IN-LAW', 'SonS', '0',
       'MO9', 'Fiance', '33.4', 'CHILF', '254727756559', '0720401466',
       'GRAND FATHER', 'PRINCIPAL', 'Guardian', 'JUNIOR',
       'DGRAND DAUGHTER', 'COWORKER', 'GRAND MOTHER', 'D100', 'NMO',
       'BUSINESS PARTNER', 'GURDIAN', '2021', 'MotH', 'MoM', 'UN,CLE',
       'GRAND CHILD', 'Sister in law', 'Grand Daughter', 'COMAPANION',
       'BOYFRIEND', 'EX-WIFE', 'GRANDDAUGTER', 'PARNT', 'M', 'F', 'Chils',
       ';SON', 'GUARDINA', 'NiecE', 'Brother-IN-LAW', 'Member',
       'Grand mother', 'ADMINISTRATOR', 'VM', 'GRANDFATHER', 'GIRLFRIED',
       'Brestfriend', 'Finance', 'SIS', 'Daughter in law', 'children',
       'Girlfriend', 'COLLEAGUE', 'TRUSTEE', '254708368426', 'Colleague',
       'Sister-IN-LAW', 'GRAND-DAUGHTER', 'GRANNY', 'GRANDCHILD', "'M",
       'spouce', 'SiLBLING', 'Siblings', 'Step-mother', 'Aunty', 'FREINF',
       '50', '100', '254723993458', 'Cousins', 'Cousins ', 'ASSOCIATE',
       'Fiane', 'guardian', 'Guadian', 'Confidant', 'Auntie', 'GRANDPA',
       'mom', 'Brother in Law', 'BUSINESS PARTNER ', 'Spuose',
       'Business Partner', 'Dating', 'IN LAW', 'STEP-MOM', 'Son/Brother',
       'Chid', 'Granddaughter', 'niece', 'Son/SISTER', 'Grand Son',
       'BrotheR', "Nolan's Guardian", "Xena's Guardian", 'Sister-in law',
       'PARENTS', 'c/o', 'STEP MOTHER', 'NiCE', 'Ciru', 'DEPENDANT',
       'som', 'GUARDIAN ', 'Dau`', 'Kid', 'PARENT ', 'CHILDREN',
       'partner ', 'CLOUSIN', '19112004', 'twinsister', 'MINOR',
       'Sister- Custodian', 'GRANSON', 'COMPANION', 'ENGAGED',
       'WELL WISHER', 'PARNER', 'GRANCHILD', 'Trustee', 'Next of kin',
       'SIIBLING', 'SIBLINGS', 'Foster daughter', 'ADMINSTRATOR',
       'IN-LAW', 'Boyfriend', 'NepheW', 'EMPLOYEE', 'DIRECTORS',
       'RELIGIOUS COMMUNITY', 'CUSTODIAN', 'C/O ARIANNA', 'Sister IN LAW',
       'spose', 'SPAUSE', 'MEMBER', 'DomesticPartner', 'Grand-daughter',
       'Brother-in-law', 'STEP-BROTHER', 'Mother to Evan', 'Sister-Inlaw',
       'BUS. PARTNER', 'Sister(Guardian)', 'CONTACT PERSON', 'FatheR',
       'Brother in law', 'Parwnt', 'M,', 'Sister-GUARDIAN', 'Grand Child',
       'GODMOTHER', 'Sister-Guardian', 'Mother- Guardian', 'Grandchild',
       'Grand son', 'CHLD', 'GRAND.DAUGHTER', 'GRAND-MOTHER', 'Parents',
       'EX WIFE', 'Sister - Guardian', 'Step-Mother', 'FOSTER DAUGHTER',
       'Grand-mother', 'God Mother', 'Mother-in-law', 'MEMBER ',
       'B/S PARTNER', 'LAWYER', 'GRAND-SON', 'parents', 'Stepson',
       'Contingent', 'grandmother', 'G/Daughter', 'Brother and sister',
       'Sibbling', ' ', 'COUSIN ', 'DaughteR', 'SPUOSE', 'CHIILD',
       'DSPOUSE', 'Executor', 'HU', 'EWIFE', 'SPOUDSE', 'SoPOUSE',
       'grandchild', 'ChILD', '9266025', 'PARE NT', '254700442162',
       'BEST FRIEND', 'SISTER IN LAW ', 'mather', 'SAVING PARTNER ',
       'AUNTIE ', 'ParenChild', 'GIRLFRIEND ', 'SINGLE ', 'Parentchild',
       'parentchild', 'SPOUSE20', 'Myself', 'FIACEE', 'SD',
       'Grand mother ', 'Mother of child', 'Childnephew', 'FATH ',
       'MY SON', 'BabyMother', 'CHID', 'GIRL FRIEND', 'chils', 'self ',
       "Mother's ", 'Mum ', 'aunty', 'SPouse', 'Auntie ', 'ChiLD',
       'grandmother ', 'Grand daughter ', 'auntie', 'baby mama ', 'DA',
       'BrOther', 'Fiancee ', 'In law', 'Colleague ', 'WiFE', 'owner',
       'sON', 'Mather', 'Dauhgter', 'myself', 'Hubby', 'Sister in-law',
       'Aunty ', 'Foster parent', 'grandson ', 'Wife/Guardian',
       '21761402', 'GUARDIAN-MOTHER'], dtype=object)
```

In [304…

```
# Convert values in the 'relationship' column to lowercase and remove any extra spac
df['relationship'] = df['relationship'].str.lower().str.strip()

# Get the counts of each unique value in the 'relationship' column,
# then format the output as a single line string
df['relationship'].value_counts().to_string().replace("\n", " ")
```

Out[304…

```
"mother                 18900 son                 18207 daughter
18010 sister                12234 wife                 11190 husband
9563 brother                7768 father                6738 spouse
3311 child                  1567 parent                 627 friend
393 niece                   369 nephew                 313 cousin
309 aunt                    220 partner                203 sibling
199 fiance                  137 guardian               117 fiancee
100 uncle                    85 grandson                74 grandmother
61 family                   51 self                    42 -
40 estate                   39 grand daughter          31 dad
29 granddaughter            29 mum                     27 girlfriend
26 doughter                 25 relative                18 boyfriend
17 aunty                    16 grandchild              16 member
15 business partner         15 auntie                  14 sister in law
12 grandfather              11 mom                     11 gurdian
11 kid                      10 so                      10 grand-daughter
9 trustee                   8 grand mother             8 s0n
8 single                    7 colleague                7 grand child
7 patner                    6 spouce                   6 100
5 parents                   5 owner                    4 dependant
4 married                   4 grand son                4 0
4 daughetr                  4 grand-mother             4 foster daughter
4 daugther                  3 bus. partner             3 minor
3 in law                    3 grand father             3 na
3 daighter                  3 parentchild              3 my son
3 spause                    3 50                       2 girl friend
2 siblings                  2 daugher                  2 myself
2 baby mama                 2 depedant                 2 sister-guardian
2 mothers                   2 god mother               2 son/brother
2 chils                     2 chid                     2 sister in-law
2 daughter-in-law           2 granddaugter             2 cousins
2 associate                 2 brother in law           2 guardina
2 vhild                     2 ex-wife                  2 b/s partner
2 vm                        2 c/o arianna              2 children
2 spuose                    2 in-law                   2 mother-in-law
2 parner                    2 confidant                2 close friend
2 sister-in-law             2 contact person           2 step-mother
2 m                         2 grandma                  2 sis
2 brother-in-law            2 mather                   2 beneficiary
2 1226805243;130143346      2 babymother               1 mother to evan
1 joint partner             1 2021                     1 dauhgter
1 'm                        1 mother- guardian         1 d100
1 sibbling                  1 guadian                  1 15
1 best friend               1 brother and sister       1 daughter/child
1 ass0ciate                 1 sd                       1 xena's guardian
1 sons                      1 brestfriend              1 step mother
1 daughter in law           1 parnt                    1 chiild
1 ciru                      1 sister-in law            1 dspouse
1 fath                      1 others                   1 fiancé
1 grand-son                 1 religious community      1 hubby
1 254700442162              1 son/sister               1 silbling
1 employee                  1 grand.daughter           1 un,cle
1 som                       1 fiane                    1 husband/spouse
1 bother                    1 sistet                   1 finance
1 granny                    1 spiritual father         1 morher
1 granson                   1 19112004                 1 ewife
1 33.4                      1 freinf                   1 sister - guardian
1 domesticpartner           1 254708368426             1 executor
1 childnephew               1 mentor                   1 daughte
1 stepson                   1 twinsister               1 moth
1 spoudse                   1 junior                   1 me
1 sister-inlaw              1 sister- custodian        1 siibling
1 father-in-law             1 guardian-mother          1 granchild
1 wife/guardian             1 254727756559             1 nice
1 closefriend               1 advisor                  1 hu
1 directors                 1 c/o                      1 next of kin
1 foster parent             1 myson                    1 clousin
1 da                        1 step-mom                 1 pare nt
1 chil                      1 33                       1 sisiter
```

```
1 saving partner                1 custodian                   1 nolan's guardian
1 cousn                         1 companion                   1 sister(guardian)
1 m,                            1 fiacee                      1 ;son
1 254723993458                  1 sp0use                      1 dating
1 bishop                        1 daughter33                  1 254701361835
1 bro                           1 step-brother                1 g/daughter
1 well wisher                   1 grandpa                     1 child minor
1 mother of child               1 lawyer                      1 mother to son
1 mo9                           1 parenchild                  1 dgrand daughter
1 coworker                      1 domestic partner            1 chilf
1 spose                         1 marriage                    1 parwnt
1 person                        1 21761402                    1 faither
1 comapanion                    1 0720401466                  1 dau`
1 b inlaw                       1 f                           1 chld
1 sopouse                       1 principal                   1 nmo
1 ex wife                       1 baby                        1 administrator
1 grandparent                   1 godmother                   1 adminstrator
1 mwas                          1 spouse20                    1 mother's
1 girlfried                     1 9266025                     1 contingent
1 engaged                       1                             1"
```

In [305…

```python
# Replace any missing values in the 'relationship' column with 'son'
df['relationship'] = df['relationship'].fillna('son')

# Check if there are any remaining missing values in the 'relationship' column
df['relationship'].isna().sum()
```

Out[305…

```
0
```

3.1 Mapping Columns

To organize and standardize relationship values in the dataset, we define categories for mapping different variations of relationship terms. This approach creates a consistent structure by grouping terms like "spouse," "child," "sibling," etc., into predefined categories. By applying this mapping, we ensure that any variations in spelling, formatting, or synonymous terms are aligned under a single label.

The following code accomplishes this by:

Defining a dictionary (relationship_map) that assigns potential variations of relationship terms to specific categories. Creating a function (map_relationship) that checks each entry in the relationship column against the keywords in relationship_map. Applying this function to the column to classify each entry according to the defined categories. Finally, we display the count of values in each category to confirm the classification distribution.

In [306…

```python
# Convert all entries to lower case and strip whitespace for uniformity
df['relationship'] = df['relationship'].str.lower().str.strip()

# Mapping dictionary
relationship_map = {
    'partner': [
        'partner', 'spouse', 'sp0use','husband', 'wife', 'fiancee', 'ex-wife', 'spou
        'husband ', 'sponse', 'spouse ', 'wiFe', 'wifE', 'wife ', 'spouse', 'married
        'husband/spouse', 'ex-wife', 'fiance ', 'fiance', 'married ', 'myson', 'baby
        'fiancee ', 'souse', 'hubby', 'dating', 'boyfriend', 'girlfriend', 'ex wife'
        'girlfriend ', 'wife/guardian', 'ex husband', 'girl friend', 'fiancée', 'sop
        'soPOUSE', 'spause', 'sposue', 'ex husband', 'love', 'b/f', 'g/f', 'divorcee
        'bf', 'gf', 'b/f', 'g/f', 'fiancé ', 'domesticpartner', 'domestic partner',
        'exhusband', 'wive', 'husb', 'exspouse', 'partner ', 'significant other', 'f
        'Husband ', 'Husband',  'Wife',   'SPOUSE' , 'Spouse ', 'WIfe', 'Spouse',  '
        'HUSBAND', 'husband', 'husband ',  'SP0USE' , 'SPOUSE ',  'wife ',
```

```
        'spouse ', 'Partner ' , 'FIANCEE', 'Fiancee',  'HUSBAND ',  'FIANCE',
        'wIFE', 'PARTNER',  'Domestic Partner', 'PATNER', 'Fiance ', 'PARTNER ',  'h
        'WIFE ', 'fiancee', 'JOINT PARTNER', 'Fiancé ' , 'Husband/spouse', 'SPOUCE',
        'Fiance', 'BOYFRIEND', 'EX-WIFE',  'PARNT',  'GIRLFRIED'
    ],
    'child': [
        'child', 'kid', 'daughter', 'son', 'children', 'daughter33', 'doughter', 'da
        'daugther', 'son ', 'daughter ', 'child minor', 'kids', 'baby', 'infant', 't
        'minor', 'child ', 'son/sister', 'chid', 'newborn', 'son/brother', 'childnep
        'chils', 'chiLD', 'kID', 'dau`', 'daugher', 'childnephew', 'my son','child',
        'daughter', 'daughter33', 'my son', 'infant', 'childnephew', 'kids', 'baby',
        'sON', 'cHILS', 'CHID', 'CHIILD', 'KID', 'FOSTER' 'DAUGHTER', 'CHILD MINOR',
        'SOM', 'CHILDREN', 'JUNIOR', 'BABY',  'CHIL', 'DAUGHTER/CHILDDAUGHER', 'CHIL
        'DAUGHTE', 'DAUGHETR', 'S0N','MYSON','child','CHILD','Child ','DAUGHTER/CHIL
        'CHILD ','children','CHILDREN','ChILD','ChiLD'
    ],
    'parent': [
        'mother', 'mom', 'mum', 'father', 'dad', 'parent', 'parents', 'mother ', 'fa
        'dad ', 'mum ', 'mom ', 'mummy', 'daddy', 'papa', 'mama', 'mother in law', '
        'mothers', 'fathers', 'moms', 'mums', 'father-in-law', 'mother-in-law', 'mom
        'parent ', 'mother to son', 'mother of child', 'parents', 'mother- guardian'
        'parentchild', 'parental', 'parenthood', 'parenting', 'mother IN LAW', 'mom
        'mum in law', 'parenting', 'fatherhood', 'motherhood', 'parent ', 'grandpare
        'step father', 'step-mother', 'step-father','Parent', 'Parents', 'Mother', '
        'Father', 'DAD', 'Dad', 'Guardian', 'Grandmother', 'Grandfather', 'Step-moth
        'Mother-in-law', 'Father-in-law', 'ParentChild', 'Mama', 'Mummy', 'Dadi', 'M
    ],
    'self': [
        'self', 'owner', 'me', 'myself', 'self ', 'owner ', 'i', 'myself ', 'persona
        'own', 'my own', 'self-employed', 'proprietor', 'me ', 'self employed', 'sel
        'my account', 'own account'
    ],
    'guardian': [
        'guardian', 'custodian', 'trustee', 'guard', 'guardian ', 'custodian ', 'tru
        'legal guardian', 'guardianship', 'custodianship', 'trusteeship', 'protector
        'conservator','conservator', 'foster parent', 'foster guardian', 'legal cust
    ],
    'sibling': [
        'siblings', 'brother', 'sister', 'sibling', 'brother ', 'sister ', 'bro', 's
        'sibling ', 'brother-in-law', 'sister-in-law', 'sibling in law', 'brother an
        'siBLING', 'sibblings', 'sister in law', 'brother in law', 'sister/guardian'
        'sister-guardian', 'bros', 'sisses', 'step-sister', 'step-brother', 'half-si
       'Sister ', 'SISTER','Sibling', 'brother ','Sibling ','SISTER ','Sistet', 'Bro
        'SIBLING',  's0n', 'sister ''sISTER', 'sisiter', ' BROTHER','BRother', 'Sist
       'som',
       'Sister- Custodian',
       'SIIBLING', 'SIBLINGS','Brother and sister',
       'Sibbling'
    ],
    'relative': [
        'cousin', 'nephew', 'grand child', 'niece', 'grandmother', 'granddaughter',
        'aunt', 'uncle', 'granny', 'grandparent', 'grand child', 'relative', 'relati
        'aunties', 'aunts', 'grandparents', 'great grandmother', 'great grandfather'
        'grandmother ', 'grandfather ', 'nephew ', 'niece ', 'cousin ', 'aunt ', 'un
        'grandson ', 'grandchild', 'grandchild ', 'grandchildren', 'grandkids', 'gre
        'kin', 'kinship', 'next of kin', 'in-laws', 'inlaw', 'in laws', 'extended re
'NEPHEW', 'Grand daughter', 'DAUGHTER-IN-LAW', 'COUSN', 'Uncle', 'Granddaughter ', '
'GRANDCHILD', 'Aunty', 'Cousins', 'Cousins ', 'Auntie', 'GRANDPA', 'Brother in Law',
    ],
    'friends': ['friend','closefriend', 'confidant', 'friend ',
        'peers', 'acquaintance', 'comrade', 'pal', 'buddy', 'mate', 'fellow', 'ally'
        'confidante', 'friend of the family', 'family friend',
        'peer', 'companion', 'companions'
    ],
```

```
    'professional':['colleague', 'coworker', 'partner in law',
        'associate', 'advisor', 'mentor', 'colleague ', 'coworker ',
        'associate ', 'colleague', 'professional', 'mentor', 'adviser', 'counselor',
        'co-worker', 'workmate', 'teammate',
        'partner in business', 'business associate', 'collaborator', 'collegue','est
    ],
    'other': ['spiritual advisor','sponsor'
    ]
}

# Function to apply the mapping
def map_relationship(value):
    for category, keywords in relationship_map.items():
        if any(keyword == value for keyword in keywords):
            return category
    return 'other'  # Default category if no matches found

# Apply the mapping function to the 'relationship' column
df['relationship'] = df['relationship'].apply(map_relationship)

# Show the value counts of each category
print(df['relationship'].value_counts())
```

```
child           52152
parent          26352
partner         24578
sibling         20240
relative         1534
friends           396
other             392
guardian          128
professional       67
self               49
Name: relationship, dtype: int64
```

In [307…

```
#checking the columns in the dataframe
df.columns
```

Out[307…

```
Index(['member_no', 'dob', 'gender', 'town', 'relationship', 'beneficiery_dob',
       'portfolio'],
      dtype='object')
```

In [308…

```
#inspecting the value_counts in the gender column
df.gender.value_counts()
```

Out[308…

```
Female    83178
Male      42480
F            92
M            48
FEMALE        4
MALE          4
Name: gender, dtype: int64
```

In [309…

```
#checking for null values
df.gender.isna().sum()
```

Out[309…

```
82
```

In [310…

```
#fill the null values with the 'Female' gender
df['gender']=df.gender.fillna('Female')
df['gender'].isna().sum()
```

Out[310…

```
0
```

To standardize entries in the gender column, we create a dictionary to map various formats of "Male" and "Female" into consistent labels. This step ensures uniformity across the dataset, which is essential for accurate analysis.

The process involves:

Defining a gender_map dictionary where different versions of "Male" and "Female" are assigned to a single, standardized label. Using the replace function to apply this mapping to the gender column and store the cleaned values in a new column, gender_mapped. This streamlined approach produces a clean and uniform gender_mapped column ready for analysis.

In [311…
```
gender_map = {
    'Female':'Female','F':'Female','FEMALE':'Female',
    'Male':'Male','M':'Male','MALE':'Male',
}

df['gender_mapped'] = df['gender'].replace(gender_map)
```

In [312…
```
#inspecting the cleaned df
df.head()
```

Out[312…

	member_no	dob	gender	town	relationship	beneficiery_dob	portfolio	gender_mapped
0	99996	1998-04-06 00:00:00	Female	NAIROBI	partner	1998-01-26	Money Market	Female
14	99996	1998-04-06 00:00:00	Female	NAIROBI	sibling	2001-03-04	Money Market	Female
28	99994	1966-01-01 00:00:00	Female	NaN	partner	1962-01-01	Money Market	Female
40	99993	1974-01-01 00:00:00	Female	NAIROBI	partner	1970-09-29	Money Market	Female
70	99993	1974-01-01 00:00:00	Female	NAIROBI	child	1994-01-01	Money Market	Female

In [313…
```
#age distribution based on dob(date of birth)
df.dob.value_counts()
```

Out[313…
```
1962-01-01             541
1960-01-01             360
1968-01-01             320
1974-01-01             311
1970-01-01             311
                      ...
1956-11-30 00:00:00      1
1957-03-22               1
1971-03-03               1
1961-07-23               1
1992-08-03 00:00:00      1
Name: dob, Length: 19092, dtype: int64
```

In [314…
```
#age distribution based on beneficiary_dob(date of birth)
df.beneficiery_dob.value_counts()
```

Out[314…
```
1899-12-30             1854
1962-01-01              584
1970-01-01              562
1960-01-01              488
1963-01-01              475
                       ...
1953-03-15                1
2005-10-25 00:00:00       1
1982-04-24 00:00:00       1
1961-07-04                1
1986-11-09                1
Name: beneficiery_dob, Length: 27488, dtype: int64
```

In [315…
```
#fill with the modal age
df['dob'] = df.dob.fillna('1962-01-01')
df.dob.isna().sum()
```

Out[315…
```
0
```

To compute ages from the dates of birth in our dataset, we:

1. **Convert Dates of Birth:** The `dob` and `beneficiery_dob` columns are first converted to datetime format to ensure proper date handling. If a date cannot be converted, it is set to `NaT` (missing).

2. **Define Age Calculation Function:** A helper function, `calculate_age`, computes the age by subtracting the birth year from the current year and adjusting for birthdays that have not yet occurred this year.

3. **Apply Function to Columns:** The function is applied to both `dob` and `beneficiery_dob` to create two new columns: `member_age` and `beneficiery_age`, which store the calculated ages.

Finally, the resulting columns are previewed to verify successful age computation.

In [316…
```
# Convert the dob column to datetime format
df['dob'] = pd.to_datetime(df['dob'], errors='coerce')

# Function to calculate age
def calculate_age(birth_date):
    if pd.isnull(birth_date):
        return None
    today = datetime.today()
    return today.year - birth_date.year - ((today.month, today.day) < (birth_date.mo

# Apply the function to calculate member age
df['member_age'] = df['dob'].apply(calculate_age)

# Convert beneficiery_dob column to datetime and calculate beneficiary age similarly
df['beneficiery_dob'] = pd.to_datetime(df['beneficiery_dob'], errors='coerce')
df['beneficiery_age'] = df['beneficiery_dob'].apply(calculate_age)

# Check results
print(df[['dob', 'member_age', 'beneficiery_dob', 'beneficiery_age']].head())
```

```
          dob  member_age beneficiery_dob  beneficiery_age
0  1998-04-06          26      1998-01-26             26.0
14 1998-04-06          26      2001-03-04             23.0
28 1966-01-01          58      1962-01-01             62.0
```

```
40 1974-01-01          50      1970-09-29             54.0
70 1974-01-01          50      1994-01-01             30.0
```

In [317…

```
# Display summary statistics for the 'beneficiery_age' column to understand the age
df.beneficiery_age.describe()
```

Out[317…

```
count    111951.000000
mean         37.980795
std          22.103286
min        -178.000000
25%          24.000000
50%          36.000000
75%          53.000000
max         329.000000
Name: beneficiery_age, dtype: float64
```

In [318…

```
#display the portfolio allocation
df.portfolio.value_counts()
```

Out[318…

```
Money Market     119710
Equity Fund        2014
Dollar Fund        1480
Balanced Fund      1180
Fixed Income       1101
Wealth Fund         398
MoneyMarket           1
Name: portfolio, dtype: int64
```

In [319…

```
#fill with the mode
df['portfolio']=df.portfolio.fillna('Money Market')
df.portfolio.isna().sum()
```

Out[319…

```
0
```

---

1. **Portfolio Mapping:** It defines a dictionary (`portfolio_map`) that standardizes various portfolio names in the `portfolio` column. For instance, entries like "Money Mrket" and "MoneyMarket" are all mapped to "Money Market," ensuring consistency across the dataset.
2. **Replace Portfolio Values:** The `replace` function is then applied to the `portfolio` column, replacing any of the portfolio variations with their standardized labels as defined in the `portfolio_map` dictionary.
3. **Create New Column:** The standardized portfolio names are saved in a new column, `portfolio_map` , in the dataframe.

In [320…

```
#clean up the portfolio column
portfolio_map ={
    'Money Mrket':'Money Market','MoneyMarket':'Money Market',
    'Equity Fund':'Equity Fund',
    'Dollar Fund':'Dollar Fund',
    'Balanced Fund':'Balanced Fund',
    'Fixed Income':'Fixed Income',
    'Wealth Fund':'Wealth Fund',
}
df['portfolio_map'] = df['portfolio'].replace(portfolio_map)
```

In [321…

```
#check the new_df
df.head()
```

Out[321…

	member_no	dob	gender	town	relationship	beneficiery_dob	portfolio	gender_mapped
0	99996	1998-04-06	Female	NAIROBI	partner	1998-01-26	Money Market	Female
14	99996	1998-04-06	Female	NAIROBI	sibling	2001-03-04	Money Market	Female
28	99994	1966-01-01	Female	NaN	partner	1962-01-01	Money Market	Female
40	99993	1974-01-01	Female	NAIROBI	partner	1970-09-29	Money Market	Female
70	99993	1974-01-01	Female	NAIROBI	child	1994-01-01	Money Market	Female

In [322…

```
#inspecting the null values in the town column
df.town.isna().sum()
```

Out[322… 15026

In [323…

```
#inspecting the value_counts in the town column
df.town.value_counts()
```

Out[323…

```
NAIROBI                   56997
Nairobi                   10261
THIKA                      3000
NAKURU                     2608
MOMBASA                    2440
                          ...
P.O.Box 848-00208 NGONG       1
Geneva                        1
OKIA                          1
NYAKIO                        1
NYAMIRA                       1
Name: town, Length: 973, dtype: int64
```

In [324…

```
#fill the null values with unknown
df['town'] = df.town.fillna('Unknown')
df.town.isna().sum()
```

Out[324… 0

In [325…

```
df.head()
```

Out[325…

	member_no	dob	gender	town	relationship	beneficiery_dob	portfolio	gender_mapped
0	99996	1998-04-06	Female	NAIROBI	partner	1998-01-26	Money Market	Female
14	99996	1998-04-06	Female	NAIROBI	sibling	2001-03-04	Money Market	Female
28	99994	1966-01-01	Female	Unknown	partner	1962-01-01	Money Market	Female
40	99993	1974-01-01	Female	NAIROBI	partner	1970-09-29	Money Market	Female

	member_no	dob	gender	town	relationship	beneficiery_dob	portfolio	gender_mapped
70	99993	1974-01-01	Female	NAIROBI	child	1994-01-01	Money Market	Female

In [326…
```
# Drop unnecessary columns ('dob', 'gender', 'portfolio', 'beneficiery_dob') from th
new_df=df.drop(columns=['dob','gender','portfolio','beneficiery_dob'])
new_df.head()
```

Out[326…

	member_no	town	relationship	gender_mapped	member_age	beneficiery_age	portfolio_ma
0	99996	NAIROBI	partner	Female	26	26.0	Money Marke
14	99996	NAIROBI	sibling	Female	26	23.0	Money Marke
28	99994	Unknown	partner	Female	58	62.0	Money Marke
40	99993	NAIROBI	partner	Female	50	54.0	Money Marke
70	99993	NAIROBI	child	Female	50	30.0	Money Marke

In [327…
```
#fill null values with the mode
new_df['member_age'].fillna(new_df['member_age'].mode()[0], inplace=True)
new_df['beneficiery_age'].fillna(new_df['beneficiery_age'].mode()[0], inplace=True)
```

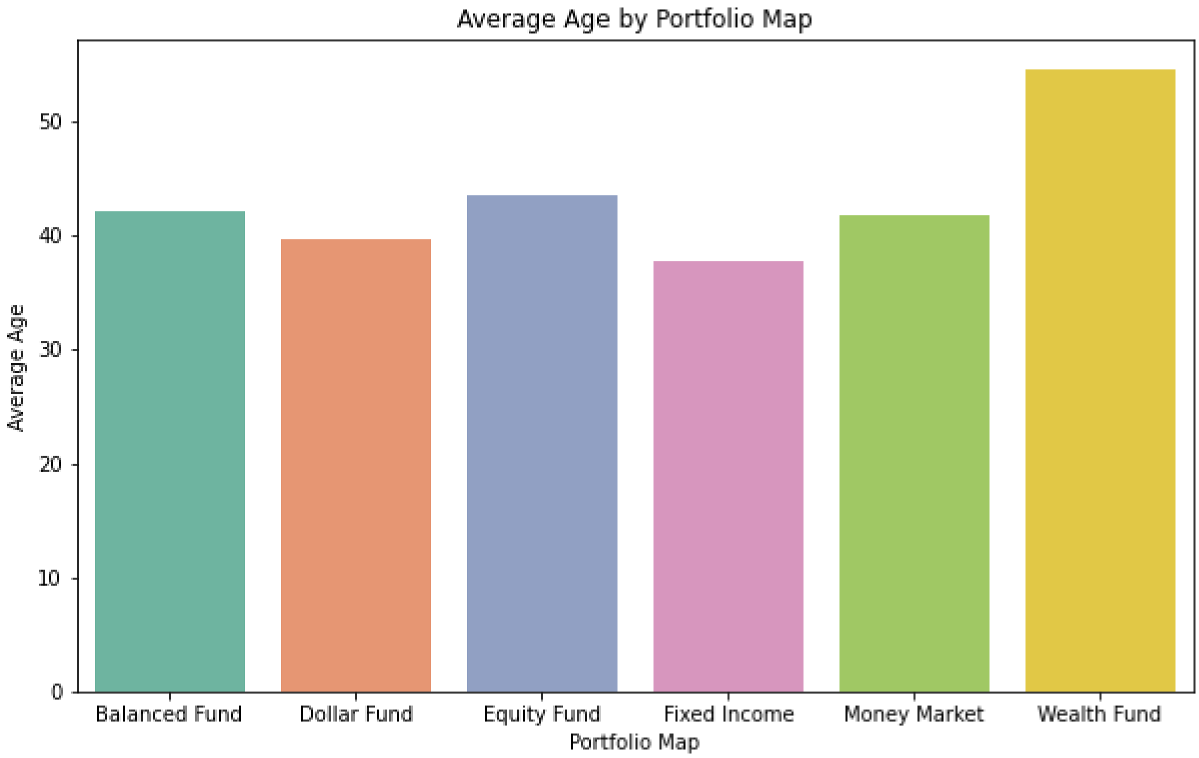
In [328…
```
# Set reasonable age bounds
lower_bound = 0
upper_bound = 100

# Create a copy of original ages for comparison if needed
original_ages = new_df['beneficiery_age'].copy()

# Replace outliers by clipping to bounds
new_df['beneficiery_age'] = new_df['beneficiery_age'].clip(lower=lower_bound, upper=
```

3.2 DATA VISUALIZATION

Here we visualize the various columns within our new data set to see how they relate.

3.2.1 A Barplot showing Average Age by Portfolio Map

In [329…
```
average_age = new_df.groupby('portfolio_map')['member_age'].mean().reset_index()
plt.figure(figsize=(10, 6))
sns.barplot(
    x='portfolio_map',
    y='member_age',
    data=average_age,
    hue='portfolio_map',
    palette="Set2",
    dodge=False
).legend([], [], frameon=False)
plt.title('Average Age by Portfolio Map')
plt.xlabel('Portfolio Map')
plt.ylabel('Average Age')
plt.show()
```



The output is a bar plot showing the average age for each portfolio type. Each bar represents the average age of members who belong to that specific portfolio, allowing you to quickly compare the average ages across different portfolio categories. The resulting plot gives insights into the age distribution for each portfolio, highlighting which portfolios are associated with older or younger members.

In [330…

```
#check columns
new_df.columns
```

Out[330…

```
Index(['member_no', 'town', 'relationship', 'gender_mapped', 'member_age',
       'beneficiery_age', 'portfolio_map'],
      dtype='object')
```

In [331…

```
#convert the cleaned data to csv
new_df.to_csv('cleaned_data.csv', index=False)
```

In [332…

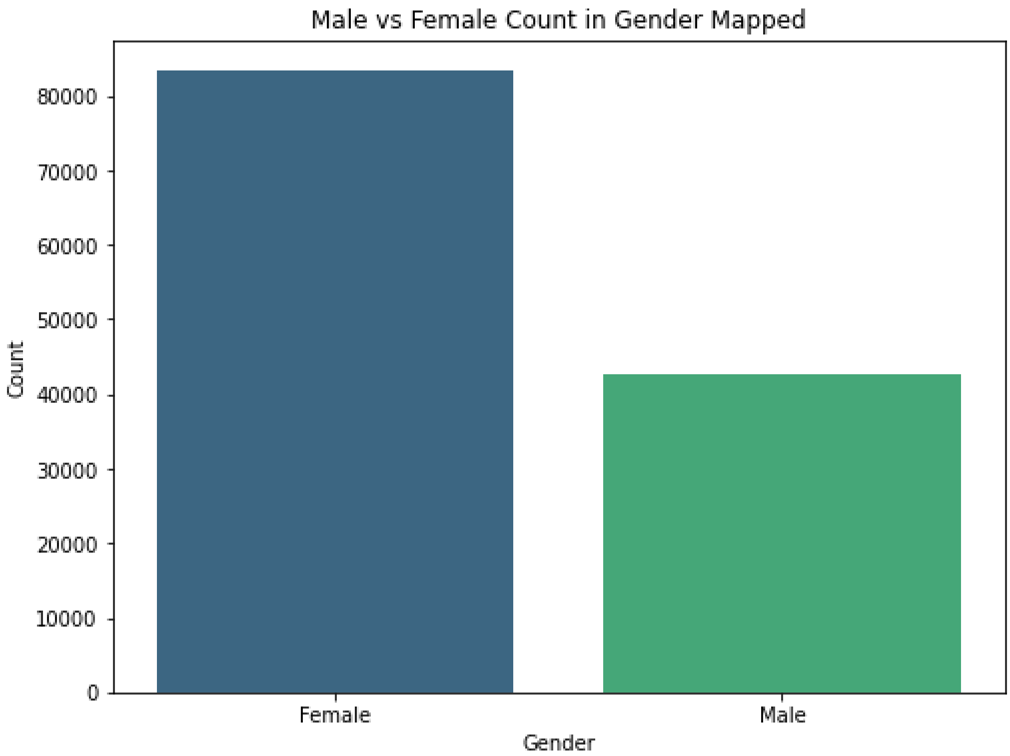
```
#inspecting the preprocessed dataframe
new_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 125888 entries, 0 to 7532842
Data columns (total 7 columns):
 #   Column           Non-Null Count   Dtype  
---  ------           --------------   -----  
 0   member_no        125888 non-null  int64  
 1   town             125888 non-null  object 
 2   relationship     125888 non-null  object 
 3   gender_mapped    125888 non-null  object 
 4   member_age       125888 non-null  int64  
 5   beneficiery_age  125888 non-null  float64
 6   portfolio_map    125888 non-null  object 
dtypes: float64(1), int64(2), object(4)
memory usage: 7.7+ MB
```

3.2.2 A Barplot showing Male vs Female Count in Gender Mapped.

In [333…

```
plt.figure(figsize=(8, 6))
sns.countplot(
    data=new_df,
    x='gender_mapped',
    hue='gender_mapped',  # Assign `x` variable to `hue`
    palette='viridis'
).legend([], [], frameon=False)  # Remove the legend
plt.title('Male vs Female Count in Gender Mapped')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.show()
```

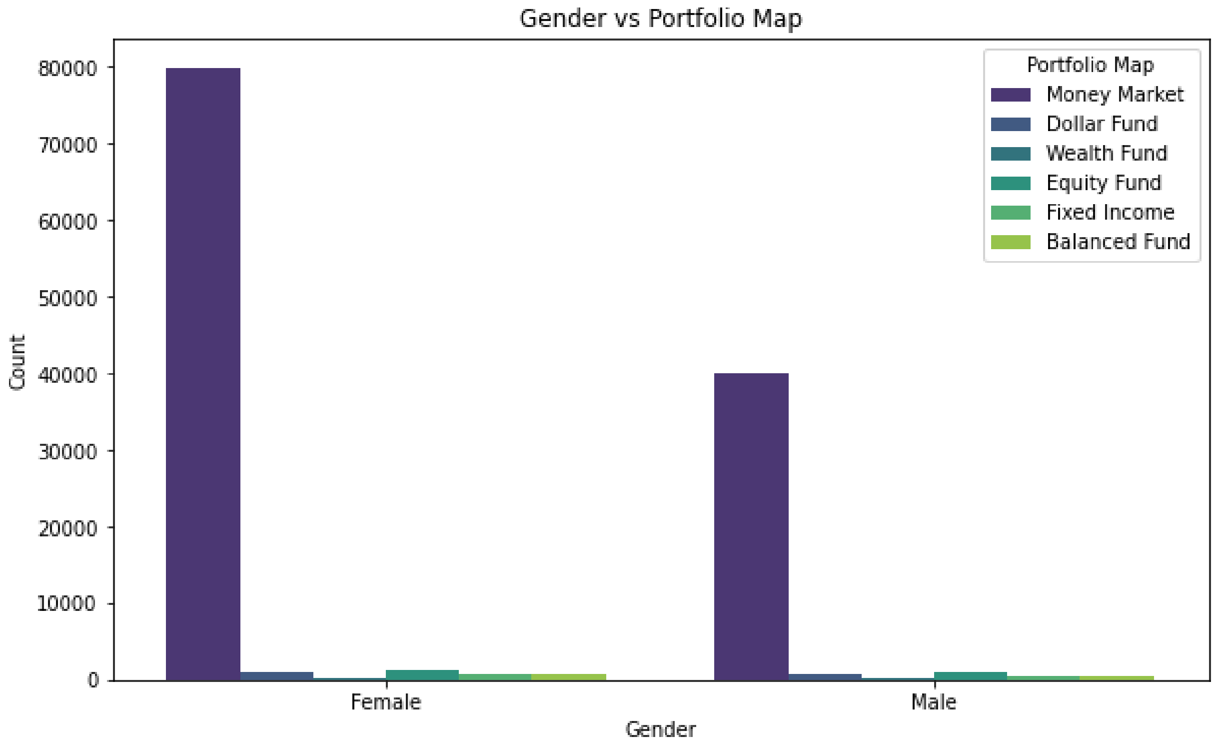


This plot gives a clear visual comparison of how many male and female entries are present in the dataset after the gender mapping process. The resulting plot allows for an easy comparison of the number of male versus female entries in the dataset, making it visually straightforward to assess gender distribution.

3.2.3 A Barplot showing the Relationship between Gender and Portfolio Map

In [334…

```
plt.figure(figsize=(10, 6))
sns.countplot(data=new_df, x='gender_mapped', hue='portfolio_map', palette='viridis'
plt.title('Gender vs Portfolio Map')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.legend(title='Portfolio Map')
plt.show()
```

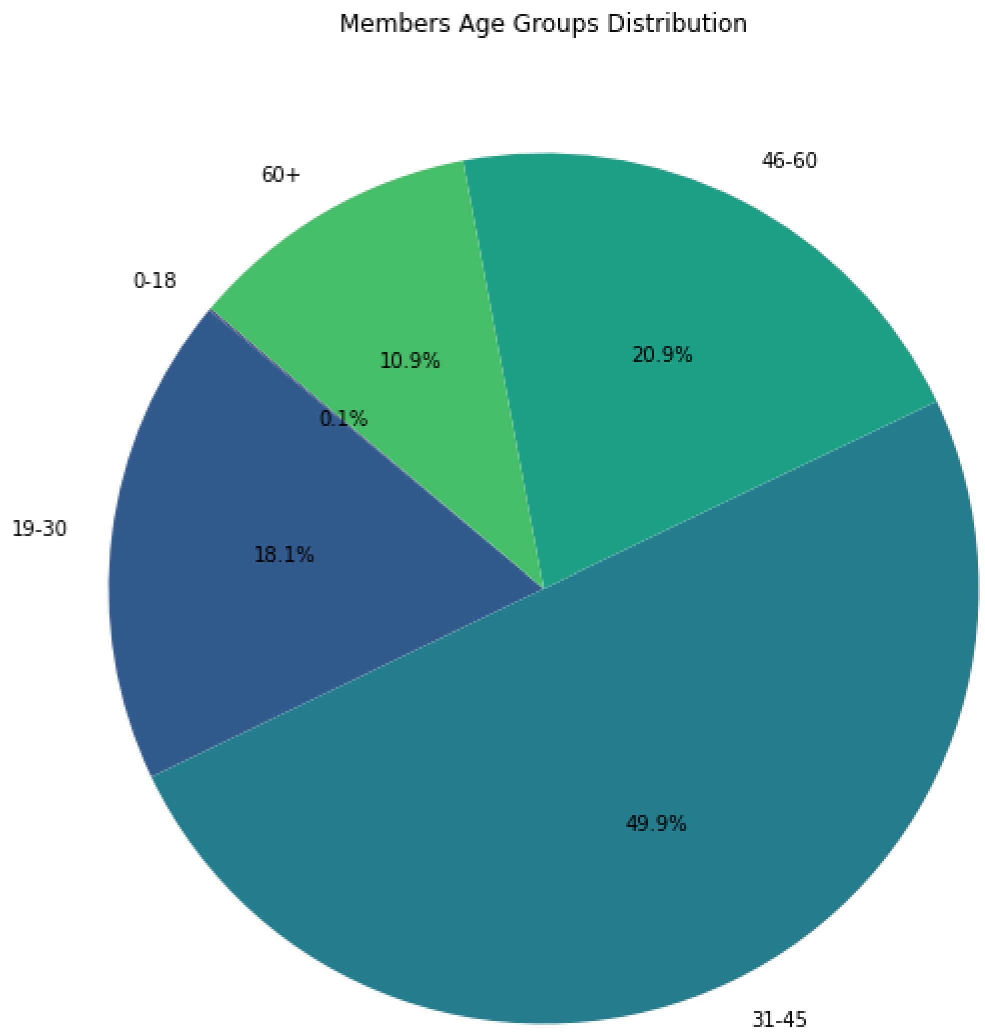


This plot provides a clear visual comparison of how genders are distributed across different portfolio categories, helping to understand the intersection of gender and portfolio preferences. It allows you to see if certain portfolio types are more favored by one gender over the other or if the distribution is relatively balanced.

3.2.4 A Piechart showing Members Age Groups Distribution.

In [335…

```
age_bins = [0, 18, 30, 45, 60, 100]
age_labels = ['0-18', '19-30', '31-45', '46-60', '60+']
new_df['age_group'] = pd.cut(new_df['member_age'], bins=age_bins, labels=age_labels)
age_counts = new_df['age_group'].value_counts().sort_index()
plt.figure(figsize=(10, 10))
plt.pie(age_counts, labels=age_counts.index, autopct='%1.1f%%', startangle=140, colo
plt.title('Members Age Groups Distribution')
plt.show()
```

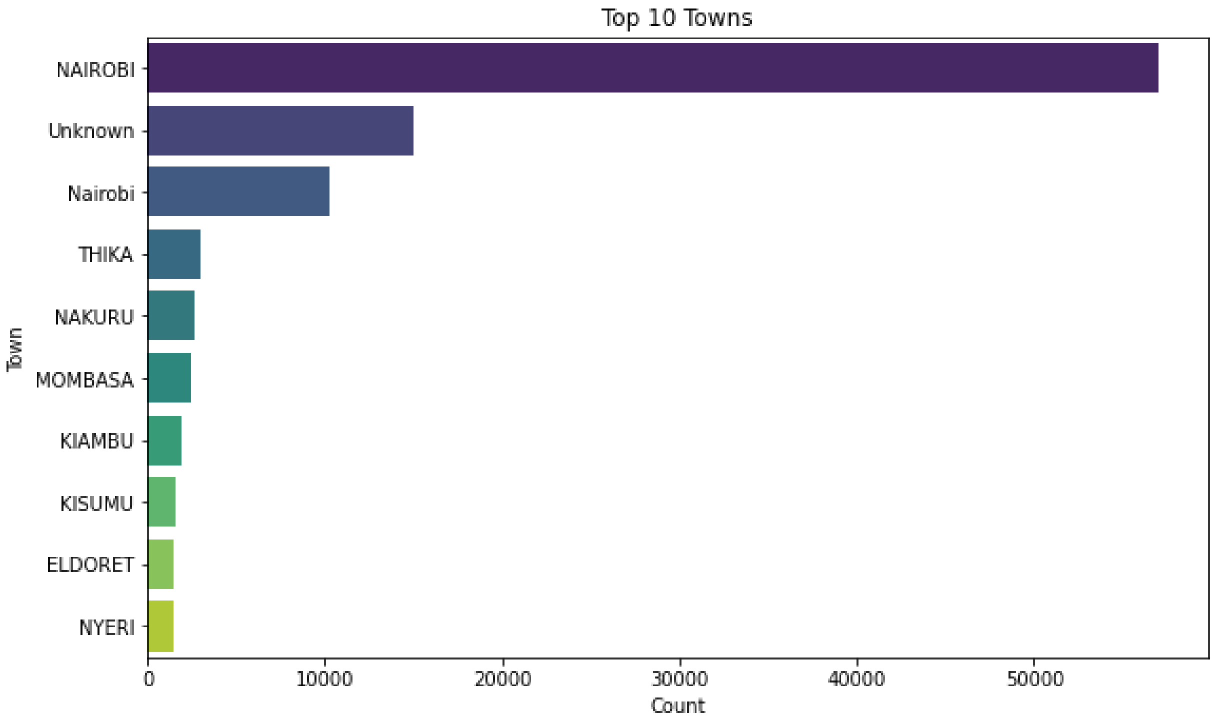


Each segment of the pie chart represents one of these age groups. The size of each segment corresponds to the percentage of members that fall into each age group. The color palette viridis provides visually distinguishable colors for each segment. The autopct option displays the percentage of total members in each age group directly on the chart, making it easy to compare the distribution across groups. This pie chart provides a clear visual of the age distribution among the members, helping to identify which age groups are the most and least represented in the dataset.

3.2.5 A barplot Showing the Top 10 Towns In terms of Investment.

In [336…

```
plt.figure(figsize=(10, 6))
sns.barplot(
    x=top_10_towns.values,
    y=top_10_towns.index,
    palette='viridis',
    hue=top_10_towns.index,
    dodge=False,
    legend=False
)
plt.title('Top 10 Towns')
plt.xlabel('Count')
plt.ylabel('Town')
plt.show()
```



Overview of the most common towns in the dataset, highlighting their relative frequencies.

4.0 MODELING

In [337…

```
#inspect the new_df
new_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 125888 entries, 0 to 7532842
Data columns (total 8 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   member_no         125888 non-null  int64
 1   town              125888 non-null  object
 2   relationship      125888 non-null  object
 3   gender_mapped     125888 non-null  object
 4   member_age        125888 non-null  int64
 5   beneficiery_age   125888 non-null  float64
 6   portfolio_map     125888 non-null  object
 7   age_group         125808 non-null  category
dtypes: category(1), float64(1), int64(2), object(4)
memory usage: 12.8+ MB
```

In [338…

```
#check the first entries in the dataframe
new_df.head()
```

Out[338…

	member_no	town	relationship	gender_mapped	member_age	beneficiery_age	portfolio_ma
0	99996	NAIROBI	partner	Female	26	26.0	Money Marke
14	99996	NAIROBI	sibling	Female	26	23.0	Money Marke
28	99994	Unknown	partner	Female	58	62.0	Money Marke
40	99993	NAIROBI	partner	Female	50	54.0	Money Marke
70	99993	NAIROBI	child	Female	50	30.0	Money Marke

In [339…

```
# Create a feature matrix combining the relevant columns
X = new_df[['member_age', 'beneficiery_age', 'age_group', 'gender_mapped']].copy()
```

```
# Convert the feature columns to a single string for each row
X['features'] = X.astype(str).sum(axis=1)

# Create a TF-IDF matrix from the features
tfidf = TfidfVectorizer(max_features=1000)
X_tfidf = tfidf.fit_transform(X['features'])

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X_tfidf, new_df['portfolio_map']
```

In [340…

```
# Train the Nearest Neighbors model
model = NearestNeighbors(metric='cosine', algorithm='brute')
model.fit(X_train)
```

Out[340…

```
NearestNeighbors(algorithm='brute', metric='cosine')
```

In [341…

```
def get_recommendations(member_features, n=5):
    # Get distances and indices of the nearest neighbors
    distances, indices = model.kneighbors(member_features, n_neighbors=n+1)

    # Retrieve the original portfolio information from `new_df` using indices
    # Exclude the first column of `indices` since it's the item itself
    recommended_portfolios = new_df.iloc[indices[0, 1:]]['portfolio_map'].tolist()

    return recommended_portfolios
```

In [342…

```
# Make predictions on the test set
y_pred = []
for i in range(X_test.shape[0]):
    member = X_test[i].toarray()
    recommendations = get_recommendations(member.reshape(1, -1), n=1)
    y_pred.append(recommendations[0])  # Get the top recommendation

# Calculate accuracy by comparing the predicted product to the actual product
accuracy = accuracy_score(y_test, y_pred)

print(f"Accuracy: {accuracy:.2f}")
```

```
Accuracy: 0.79
```

In [343…

```
import pickle

# Save the NearestNeighbors model
with open('model.pkl', 'wb') as file:
    pickle.dump(model, file)

# Save the TF-IDF vectorizer
with open('tfidf.pkl', 'wb') as file:
    pickle.dump(tfidf, file)
```

In [344…

```
# Save the reference dataset for rule-based recommendations
new_df.to_csv('investment_member.csv', index=False)
```

RULE BASED ADDED

In [345…

```
def get_recommendations(member_features, member_no, n=5):
    # Fetch member's details
    member_row = new_df.loc[new_df['member_no'] == member_no].iloc[0]
    member_beneficiary_age = member_row['beneficiery_age']
    member_age_group = member_row['age_group']
    member_town = member_row['town']
    member_gender = member_row['gender_mapped']
```

```
    member_current_products = set(new_df.loc[new_df['member_no'] == member_no, 'port

    recommended_products = []

    # Rule 1: Recommend based on beneficiary age (Student and Junior Accounts)
    #if 18 <= member_beneficiary_age <= 25:
    if  member_beneficiary_age >= 18 and  member_beneficiary_age <= 25:
        recommended_products.append("Student Account")
    elif member_beneficiary_age < 18:
        recommended_products.append("Junior Account")

    # Rule 2: Recommend popular products within the same age group
    age_group_products = (
        new_df[new_df['age_group'] == member_age_group]
        .portfolio_map.value_counts()
        .index
        .tolist()
    )
    for product in age_group_products:
        if product not in member_current_products and product not in recommended_pro
            recommended_products.append(product)
        if len(recommended_products) >= n:
            return recommended_products[:n]

    # Rule 3: Recommend popular products in the same town
    town_products = (
        new_df[new_df['town'] == member_town]
        .portfolio_map.value_counts()
        .index
        .tolist()
    )
    for product in town_products:
        if product not in member_current_products and product not in recommended_pro
            recommended_products.append(product)
        if len(recommended_products) >= n:
            return recommended_products[:n]

    gender_products = (
        new_df[new_df['gender_mapped']== member_gender]
        .portfolio_map.value_counts()
        .index
        .tolist()
    )
    for product in gender_products:
        if product not in member_current_products and product not in recommended_pro
            recommended_products.append(product)
        if len(recommended_products) >= n:
            return recommended_products[:n]

    # Collaborative Filtering for Additional Recommendations
    distances, indices = model.kneighbors(member_features, n_neighbors=n + len(recom
    for index in indices[0]:
        product = new_df.iloc[index]['portfolio_map']
        if product not in member_current_products and product not in recommended_pro
            recommended_products.append(product)
        if len(recommended_products) >= n:
            break

    return recommended_products[:n]
```

The `get_recommendations` function generates personalized recommendations for a member by leveraging rule-based logic and collaborative filtering. It begins by fetching the member's details, such as their age group, town, gender, and current products. The function first applies

Rule 1, recommending products based on the beneficiary's age (e.g., "Student Account" or "Junior Account"). **Rule 2** suggests popular products within the same age group that the member does not already own. Similarly, **Rule 3** expands recommendations to products popular in the member's town, and another rule considers gender-based product preferences. Finally, a collaborative filtering model is employed to find additional products based on feature similarity to other members, ensuring a well-rounded and diverse recommendation list. The recommendations are constrained to a maximum of `n` products and prioritize relevance to the member's profile and preferences.

In [346…

```
new_df.columns
```

Out[346…

```
Index(['member_no', 'town', 'relationship', 'gender_mapped', 'member_age',
       'beneficiery_age', 'portfolio_map', 'age_group'],
      dtype='object')
```

5.0 EVALUATION

In [347…

```
# Test the recommendations for a sample member
sample_index = 0  # Adjust the sample index for testing
sample_member = X_test[sample_index].toarray()
sample_member_id = new_df.iloc[sample_index]['member_no']  # Ensure 'member_id' is i

# Get recommendations for the sample member
recommendations = get_recommendations(sample_member.reshape(1, -1), sample_member_id

# Display the results
print("Recommendations for sample member profile:")
print(recommendations)
```

```
Recommendations for sample member profile:
['Fixed Income', 'Equity Fund', 'Dollar Fund']
```

In [348…

```
# Test the recommendations for a sample member
sample_index = 20
sample_member = X_test[sample_index].toarray()
sample_member_id = new_df.iloc[sample_index]['member_no']

# Get recommendations for the sample member
recommendations = get_recommendations(sample_member.reshape(1, -1), sample_member_id

# Display the results
print("Recommendations for sample member profile:")
print(recommendations)
```

```
Recommendations for sample member profile:
['Equity Fund', 'Dollar Fund', 'Fixed Income']
```

In [349…

```
def get_recommendations_with_messages(member_features, member_no, n=5):
    # Fetch member's details
    member_row = new_df.loc[new_df['member_no'] == member_no].iloc[0]
    member_beneficiary_age = member_row['beneficiery_age']
    member_age_group = member_row['age_group']
    member_town = member_row['town']
    member_gender = member_row['gender_mapped']
    member_current_products = set(new_df.loc[new_df['member_no'] == member_no, 'port

    recommended_products = []
    messages = []

    # Rule 1: Recommend based on beneficiary age (Student and Junior Accounts)
    age_recommendations = []
```

```
if  member_beneficiary_age >= 18 and  member_beneficiary_age <= 25:
    age_recommendations.append("Student Account")
    messages.append(
        f"Planning for your child's future? Our Student Account is perfect for y
        f"Start securing their educational journey today!"
    )
elif  member_beneficiary_age < 18:
    age_recommendations.append("Junior Account")
    messages.append(
        f"Give your child a head start with our Junior Account! It's specially d
        f"to help them develop good financial habits early."
    )
recommended_products.extend(age_recommendations)

# Rule 2: Recommend popular products within the same age group
age_group_recommendations = []
age_group_products = (
    new_df[new_df['age_group'] == member_age_group]
    .portfolio_map.value_counts()
    .index
    .tolist()
)
for product in age_group_products:
    if product not in member_current_products and product not in recommended_pro
        age_group_recommendations.append(product)
    if len(recommended_products) + len(age_group_recommendations) >= n:
        break

if age_group_recommendations:
    messages.append(
        f"Members in your age group are enjoying these popular products: {', '.j
        f"Join them in making smart financial choices!"
    )
recommended_products.extend(age_group_recommendations)

# Rule 3: Recommend popular products in the same town
location_recommendations = []
town_products = (
    new_df[new_df['town'] == member_town]
    .portfolio_map.value_counts()
    .index
    .tolist()
)
for product in town_products:
    if product not in member_current_products and product not in recommended_pro
        location_recommendations.append(product)
    if len(recommended_products) + len(location_recommendations) >= n:
        break

if location_recommendations:
    messages.append(
        f"Trending in {member_town}! Your neighbors are choosing {', '.join(loca
        f"Discover why these products are popular in your community!"
    )
recommended_products.extend(location_recommendations)

# Rule 4: Gender-based recommendations
gender_recommendations = []
gender_products = (
    new_df[new_df['gender_mapped'] == member_gender]
    .portfolio_map.value_counts()
    .index
    .tolist()
)
```

```python
    for product in gender_products:
        if product not in member_current_products and product not in recommended_pro
            gender_recommendations.append(product)
        if len(recommended_products) + len(gender_recommendations) >= n:
            break

    if gender_recommendations:
        gender_message = (
            "Specially curated for you! " if member_gender == 'Female' else
            "Join other members like you! "
        )
        messages.append(
            f"{gender_message}Discover {', '.join(gender_recommendations)} - "
            f"products that match your financial goals."
        )
    recommended_products.extend(gender_recommendations)
    # Collaborative Filtering for Additional Recommendations
    if len(recommended_products) < n:
        collab_recommendations = []
        distances, indices = model.kneighbors(member_features, n_neighbors=n + len(r
        for index in indices[0]:
            product = new_df.iloc[index]['portfolio_map']
            if product not in member_current_products and product not in recommended
                collab_recommendations.append(product)
            if len(recommended_products) + len(collab_recommendations) >= n:
                break

        if collab_recommendations:
            messages.append(
                f"Based on your profile, we think you'll love {', '.join(collab_reco
                f"These products align perfectly with your financial journey!"
            )
        recommended_products.extend(collab_recommendations)

    # Final personalized message
    if len(recommended_products) > 0:
        messages.append(
            f"💡 Pro tip: Adding {', '.join(recommended_products[:n])} to your portf
            f"could help you achieve your financial goals faster!"
        )

    return recommended_products[:n], messages
```

This function, `get_recommendations_with_messages`, generates personalized product recommendations for a member based on their profile. Here's a breakdown:

1. **Fetch Member's Profile**: It pulls details like the member's age group, town, gender, and current products.

2. **Rule-Based Recommendations**:

    - **Beneficiary's Age**: If the beneficiary is between 18-25, it suggests a "Student Account" with a personalized message. If under 18, it recommends a "Junior Account" with a suitable message.
    - **Age Group**: It suggests popular products among other members in the same age group, excluding products the member already owns.
    - **Town**: It recommends popular products in the member's town, again excluding owned products.

- **Gender**: It suggests popular products among other members of the same gender with a gender-specific message.

3. **Collaborative Filtering**: If fewer than n recommendations are generated, it uses collaborative filtering to find additional suggestions based on members with similar profiles.

4. **Final Message**: It compiles a final message to encourage the member to consider the recommended products.

In [350…

```python
# Test the recommendations for a sample member
sample_index = 3464  # Adjust the sample index for testing
sample_member = X_test[sample_index].toarray()
sample_member_id = new_df.iloc[sample_index]['member_no']  # Ensure 'member_id' is i

# Get recommendations and messages for the sample member
recommendations, messages = get_recommendations_with_messages(sample_member.reshape(

# Display the results
print("\n🏦 Member Profile Analysis")
print("-" * 50)
print(f"Member ID: {sample_member_id}")
member_details = new_df.loc[new_df['member_no'] == sample_member_id].iloc[0]
print(f"Age Group: {member_details['age_group']}")
print(f"Town: {member_details['town']}")
print(f"Gender: {member_details['gender_mapped']}")
if not pd.isna(member_details['beneficiery_age']):
    print(f"Beneficiary Age: {member_details['beneficiery_age']}")

print("\n📊 Recommended Products")
print("-" * 50)
for i, product in enumerate(recommendations, 1):
    print(f"{i}. {product}")

print("\n💌 Personalized Messages")
print("-" * 50)
for i, message in enumerate(messages, 1):
    print(f"Message {i}:")
    print(f"{message}")
    print()

# Optional: Display current products for comparison
current_products = set(new_df.loc[new_df['member_no'] == sample_member_id, 'portfoli
if current_products:
    print("\n📝 Current Portfolio")
    print("-" * 50)
    for product in current_products:
        print(f"• {product}")
```

```
🏦 Member Profile Analysis
--------------------------------------------------
Member ID: 96336
Age Group: 31-45
Town: Unknown
Gender: Female
Beneficiary Age: 64.0

📊 Recommended Products
--------------------------------------------------
1. Equity Fund
2. Dollar Fund
3. Fixed Income

💌 Personalized Messages
--------------------------------------------------
```

```
Message 1:
Members in your age group are enjoying these popular products: Equity Fund, Dollar Fu
nd, Fixed Income. Join them in making smart financial choices!

Message 2:
💡 Pro tip: Adding Equity Fund, Dollar Fund, Fixed Income to your portfolio could he
lp you achieve your financial goals faster!


📝 Current Portfolio
--------------------------------------------------
• Money Market
```

In [351…

```python
# Test the recommendations for a sample member
sample_index = 9  # Adjust the sample index for testing
sample_member = X_test[sample_index].toarray()
sample_member_id = new_df.iloc[sample_index]['member_no']  # Ensure 'member_id' is i

# Get recommendations and messages for the sample member
recommendations, messages = get_recommendations_with_messages(sample_member.reshape(

# Display the results
print("\n🏦 Member Profile Analysis")
print("-" * 50)
print(f"Member ID: {sample_member_id}")
member_details = new_df.loc[new_df['member_no'] == sample_member_id].iloc[0]
print(f"Age Group: {member_details['age_group']}")
print(f"Town: {member_details['town']}")
print(f"Gender: {member_details['gender_mapped']}")
if not pd.isna(member_details['beneficiery_age']):
    print(f"Beneficiary Age: {member_details['beneficiery_age']}")

print("\n📊 Recommended Products")
print("-" * 50)
for i, product in enumerate(recommendations, 1):
    print(f"{i}. {product}")

print("\n💌 Personalized Messages")
print("-" * 50)
for i, message in enumerate(messages, 1):
    print(f"Message {i}:")
    print(f"{message}")
    print()

# Optional: Display current products for comparison
current_products = set(new_df.loc[new_df['member_no'] == sample_member_id, 'portfoli
if current_products:
    print("\n📝 Current Portfolio")
    print("-" * 50)
    for product in current_products:
        print(f"• {product}")
```

```
🏦 Member Profile Analysis
--------------------------------------------------
Member ID: 99988
Age Group: 31-45
Town: NAIROBI
Gender: Female
Beneficiary Age: 3.0

📊 Recommended Products
--------------------------------------------------
1. Junior Account
2. Equity Fund
3. Dollar Fund
```

```
💌 Personalized Messages
--------------------------------------------------
Message 1:
Give your child a head start with our Junior Account! It's specially designed for chi
ldren under 18 to help them develop good financial habits early.

Message 2:
Members in your age group are enjoying these popular products: Equity Fund, Dollar Fu
nd. Join them in making smart financial choices!

Message 3:
💡 Pro tip: Adding Junior Account, Equity Fund, Dollar Fund to your portfolio could
help you achieve your financial goals faster!


📝 Current Portfolio
--------------------------------------------------
• Money Market
```

In [352…

```
new_df.head()
```

Out[352…

	member_no	town	relationship	gender_mapped	member_age	beneficiery_age	portfolio_ma
0	99996	NAIROBI	partner	Female	26	26.0	Money Marke
14	99996	NAIROBI	sibling	Female	26	23.0	Money Marke
28	99994	Unknown	partner	Female	58	62.0	Money Marke
40	99993	NAIROBI	partner	Female	50	54.0	Money Marke
70	99993	NAIROBI	child	Female	50	30.0	Money Marke

5.1 Recommendations

1. **Segmented Product Recommendations by Age Group**:
   - **Younger Investors** (e.g., under 30): Consider promoting high-growth or equity-based products, as younger investors often have a higher risk tolerance and a longer investment horizon.
   - **Middle-aged Investors** (e.g., 30-55): Emphasize balanced portfolios with a mix of equity and Fixed Deposit products. This group might prioritize growth with some level of stability.
   - **Older Investors** (e.g., 55+): Highlight low-risk, income-generating products, such as Money Market, which provide regular income and capital preservation for retirement.
2. **Targeted Marketing Based on Location**:
   - Leverage regional or town-based trends identified in the data to tailor marketing campaigns. For instance, if certain towns show a preference for specific investment types, create location-based advertising to match these preferences.

5.2 Conclusions

1. **Enhanced Customer Retention and Loyalty**:
   - Personalized product recommendations, tailored to each customer's profile, help foster a sense of individual attention and care. This personal touch is likely to improve

customer satisfaction, deepening loyalty and encouraging long-term relationships with our services.

2. **Increased Revenue through Effective Cross-Selling**:

    - By successfully cross-selling products that align with customer demographics and preferences, we can expand each customer's product portfolio. This not only improves the individual customer’s experience but also increases the revenue per customer, contributing to stronger overall financial performance.

3. **Streamlined, User-Centric Experience**:

    - A user-friendly recommendation interface makes it easier for customers, particularly those unfamiliar with diverse financial products, to explore and understand new investment options. This approach empowers customers to make informed choices, driving engagement and potentially increasing the adoption of recommended products.

4.**Patterns in Age and Investment Preferences**:

- Age is a significant factor in determining investment preferences. Younger individuals show interest in high-growth products, while older investors lean towards stability-focused options. Recognizing this can aid in making age-appropriate recommendations.

1. **Impact of Geographic Location**:
    - Investment preferences appear to have geographic patterns, possibly due to regional income levels, financial awareness, or cultural attitudes towards risk. Utilizing location data can enhance the relevance of recommendations.

6.0 DEPLOYMENT

In [353…

```python
#saving the notebook as a pickle file
import nbformat
import pickle

# Define the notebook filename
notebook_filename = 'Investment Product Cross Selling (1).ipynb'

# Read the notebook content using nbformat with UTF-8 encoding
with open(notebook_filename, 'r', encoding='utf-8') as f:
    notebook_content = nbformat.read(f, as_version=4)

# Pickle the notebook content
with open('Investment_Product_Cross_Selling_notebook.pkl', 'wb') as f:
    pickle.dump(notebook_content, f)

print("Notebook content has been pickled successfully.")
```

```
Notebook content has been pickled successfully.
```