Name: Kiran Songire

```
import pandas as pd
import numpy as np
import seaborn as sns
```

```
import matplotlib.pyplot as plt
import datetime as dt
import math
```

```
data = pd.ExcelFile('KPMG_VI_New_raw_data_update_final.xlsx')
Transactions_data = pd.read_excel(data, 'Transactions', header=1)
NewCustomerList_data = pd.read_excel(data, 'NewCustomerList', header=1)
CustomerDemographic_data = pd.read_excel(data, 'CustomerDemographic', header=1)
CustomerAddress_data = pd.read_excel(data, 'CustomerAddress', header=1)
```

```
<ipython-input-540-66946817764b>:3: FutureWarning: Inferring datetime64[ns] from data containing strings is deprecated and will be remo
  NewCustomerList_data = pd.read_excel(data, 'NewCustomerList', header=1)
<ipython-input-540-66946817764b>:4: FutureWarning: Inferring datetime64[ns] from data containing strings is deprecated and will be remo
  CustomerDemographic_data = pd.read_excel(data, 'CustomerDemographic', header=1)
```

```
Transactions_data.head(5)
```

	transaction_id	product_id	customer_id	transaction_date	online_order	order_status	brand	product_line	product_class	product_s
0	1	2	2950	2017-02-25	0.0	Approved	Solex	Standard	medium	med
1	2	3	3120	2017-05-21	1.0	Approved	Trek Bicycles	Standard	medium	la
2	3	37	402	2017-10-16	0.0	Approved	OHM Cycles	Standard	low	med
3	4	88	3135	2017-08-31	0.0	Approved	Norco Bicycles	Standard	medium	med
4	5	78	787	2017-10-01	1.0	Approved	Giant Bicycles	Standard	medium	la

```
Transactions_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20000 entries, 0 to 19999
Data columns (total 13 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   transaction_id           20000 non-null  int64
 1   product_id               20000 non-null  int64
 2   customer_id              20000 non-null  int64
 3   transaction_date         20000 non-null  datetime64[ns]
 4   online_order             19640 non-null  float64
 5   order_status             20000 non-null  object
 6   brand                    19803 non-null  object
 7   product_line             19803 non-null  object
 8   product_class            19803 non-null  object
 9   product_size             19803 non-null  object
 10  list_price               20000 non-null  float64
 11  standard_cost            19803 non-null  float64
 12  product_first_sold_date  19803 non-null  float64
dtypes: datetime64[ns](1), float64(4), int64(3), object(5)
memory usage: 2.0+ MB
```

```
Transactions_data.shape
```

```
(20000, 13)
```

```
Transactions_data.isnull().sum()
```

```
transaction_id              0
product_id                  0
customer_id                 0
transaction_date            0
online_order              360
order_status                0
brand                     197
product_line              197
product_class             197
product_size              197
list_price                  0
standard_cost             197
product_first_sold_date   197
dtype: int64
```

```
Transactions_data.duplicated().sum()
```

```
0
```

```
Transactions_data.nunique()
```

```
transaction_id            20000
product_id                  101
customer_id                3494
transaction_date            364
online_order                  2
order_status                  2
brand                         6
product_line                  4
product_class                 3
product_size                  3
list_price                  296
standard_cost               103
product_first_sold_date     100
dtype: int64
```

```
Transactions_data['order_status'].value_counts()
```

```
Approved     19821
Cancelled      179
Name: order_status, dtype: int64
```

```
Transactions_data['brand'].value_counts()
```

```
Solex             4253
Giant Bicycles    3312
WeareA2B          3295
OHM Cycles        3043
Trek Bicycles     2990
Norco Bicycles    2910
Name: brand, dtype: int64
```

```
Transactions_data['product_line'].value_counts()
```

```
Standard    14176
Road         3970
Touring      1234
Mountain      423
Name: product_line, dtype: int64
```

```
Transactions_data['product_class'].value_counts()
```

```
medium    13826
high       3013
low        2964
Name: product_class, dtype: int64
```

```
Transactions_data['product_size'].value_counts()
```

```
medium    12990
large      3976
small      2837
Name: product_size, dtype: int64
```

```
Transactions_data['product_first_sold_date']
```

```
0        41245.0
1        41701.0
2        36361.0
3        36145.0
4        42226.0
          ...
19995    37823.0
19996    35560.0
19997    40410.0
19998    38216.0
19999    36334.0
Name: product_first_sold_date, Length: 20000, dtype: float64
```

```
#convert date column from integer to datetime
Transactions_data['product_first_sold_date'] = pd.to_datetime(Transactions_data['product_first_sold_date'], unit='s')
Transactions_data['product_first_sold_date'].head()
```

```
0   1970-01-01 11:27:25
1   1970-01-01 11:35:01
2   1970-01-01 10:06:01
3   1970-01-01 10:02:25
4   1970-01-01 11:43:46
Name: product_first_sold_date, dtype: datetime64[ns]
```

```
Transactions_data['product_first_sold_date'].head(20)
```

```
0    1970-01-01 11:27:25
1    1970-01-01 11:35:01
2    1970-01-01 10:06:01
3    1970-01-01 10:02:25
4    1970-01-01 11:43:46
5    1970-01-01 10:50:31
6    1970-01-01 09:29:25
7    1970-01-01 11:05:15
8    1970-01-01 09:17:35
9    1970-01-01 10:36:56
10   1970-01-01 11:19:44
11   1970-01-01 11:42:52
12   1970-01-01 09:35:27
13   1970-01-01 09:36:26
14   1970-01-01 10:36:33
15   1970-01-01 10:31:13
16   1970-01-01 10:36:46
17   1970-01-01 09:24:48
18   1970-01-01 11:05:15
19   1970-01-01 10:22:17
Name: product_first_sold_date, dtype: datetime64[ns]
```

Exploring New Customer List Data Set

```
NewCustomerList_data.head(5)
```

	first_name	last_name	gender	past_3_years_bike_related_purchases	DOB	job_title	job_industry_category	wealth_segment	decea

```
NewCustomerList_data.head(5)
```

	first_name	last_name	gender	past_3_years_bike_related_purchases	DOB	job_title	job_industry_category	wealth_segment	decea
0	Chickie	Brister	Male	86	1957-07-12	General Manager	Manufacturing	Mass Customer	
1	Morly	Genery	Male	69	1970-03-22	Structural Engineer	Property	Mass Customer	
2	Ardelis	Forrester	Female	10	1974-08-28	Senior Cost Accountant	Financial Services	Affluent Customer	
3	Lucine	Stutt	Female	64	1979-01-28	Account Representative III	Manufacturing	Affluent Customer	
4	Melinda	Hadlee	Female	34	1965-09-21	Financial Analyst	Financial Services	Affluent Customer	

5 rows × 23 columns

```
NewCustomerList_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 23 columns):
 #   Column                               Non-Null Count  Dtype
---  ------                               --------------  -----
 0   first_name                           1000 non-null   object
 1   last_name                            971 non-null    object
 2   gender                               1000 non-null   object
 3   past_3_years_bike_related_purchases  1000 non-null   int64
 4   DOB                                  983 non-null    datetime64[ns]
 5   job_title                            894 non-null    object
 6   job_industry_category                835 non-null    object
 7   wealth_segment                       1000 non-null   object
 8   deceased_indicator                   1000 non-null   object
 9   owns_car                             1000 non-null   object
 10  tenure                               1000 non-null   int64
 11  address                              1000 non-null   object
 12  postcode                             1000 non-null   int64
 13  state                                1000 non-null   object
 14  country                              1000 non-null   object
 15  property_valuation                   1000 non-null   int64
 16  Unnamed: 16                          1000 non-null   float64
 17  Unnamed: 17                          1000 non-null   float64
 18  Unnamed: 18                          1000 non-null   float64
 19  Unnamed: 19                          1000 non-null   float64
 20  Unnamed: 20                          1000 non-null   int64
 21  Rank                                 1000 non-null   int64
 22  Value                                1000 non-null   float64
dtypes: datetime64[ns](1), float64(5), int64(6), object(11)
memory usage: 179.8+ KB
```

```
NewCustomerList_data.drop(['Unnamed: 16', 'Unnamed: 17', 'Unnamed: 18',
       'Unnamed: 19', 'Unnamed: 20'], axis=1, inplace=True)
```

```
NewCustomerList_data.shape
```

```
(1000, 18)
```

```
NewCustomerList_data.isnull().sum()
```

```
first_name                             0
last_name                             29
gender                                 0
past_3_years_bike_related_purchases    0
DOB                                   17
job_title                            106
```

```
job_industry_category                165
wealth_segment                         0
deceased_indicator                     0
owns_car                               0
tenure                                 0
address                                0
postcode                               0
state                                  0
country                                0
property_valuation                     0
Rank                                   0
Value                                  0
dtype: int64
```

```
NewCustomerList_data.duplicated().sum()
```

```
0
```

```
NewCustomerList_data.nunique()
```

```
first_name                           940
last_name                            961
gender                                 3
past_3_years_bike_related_purchases  100
DOB                                  958
job_title                            184
job_industry_category                  9
wealth_segment                         3
deceased_indicator                     1
owns_car                               2
tenure                                23
address                             1000
postcode                             522
state                                  3
country                                1
property_valuation                    12
Rank                                 324
Value                                324
dtype: int64
```

```
NewCustomerList_data.columns
```

```
Index(['first_name', 'last_name', 'gender',
       'past_3_years_bike_related_purchases', 'DOB', 'job_title',
       'job_industry_category', 'wealth_segment', 'deceased_indicator',
       'owns_car', 'tenure', 'address', 'postcode', 'state', 'country',
       'property_valuation', 'Rank', 'Value'],
      dtype='object')
```

```
NewCustomerList_data['gender'].value_counts()
```

```
Female    513
Male      470
U          17
Name: gender, dtype: int64
```

```
NewCustomerList_data['job_industry_category'].value_counts()
```

```
Financial Services    203
Manufacturing         199
Health                152
Retail                 78
Property               64
IT                     51
Entertainment          37
Argiculture            26
Telecommunications     25
Name: job_industry_category, dtype: int64
```

```
NewCustomerList_data['wealth_segment'].value_counts()
```

```
Mass Customer        508
High Net Worth       251
```

```
Affluent Customer    241
Name: wealth_segment, dtype: int64
```

```
NewCustomerList_data['state'].value_counts()
```

```
NSW    506
VIC    266
QLD    228
Name: state, dtype: int64
```

```
NewCustomerList_data['owns_car'].value_counts()
```

```
No     507
Yes    493
Name: owns_car, dtype: int64
```

```
NewCustomerList_data['deceased_indicator'].value_counts()
```

```
N    1000
Name: deceased_indicator, dtype: int64
```

```
# Check the entries for each column
for col in CustomerDemographic_data.columns:
    print('{} : {}'.format(col, CustomerDemographic_data[col].unique()))
```

```
 'ð\xa0ð\xa0±ð\xa0¹ð\xa0±ð\xa0±¸ð\xa0²ð\xa0³']
owns_car : ['Yes' 'No']
tenure : [11. 16. 15.  7.  8. 13. 20.  9.  6.  1. 18. 21. 12. 19. 14.  4. 22.  5.
 17.  2.  3. 10. nan]
```

```
# Remove deceased customers
CustomerDemographic_data = CustomerDemographic_data[CustomerDemographic_data['deceased_indicator'] == 'N']

# Drop columns
CustomerDemographic_data.drop(['first_name', 'last_name', 'default', 'job_title', 'deceased_indicator'], axis=1, inplace=True)

# Drop empty row
CustomerDemographic_data.dropna(axis=0, inplace=True)

# Correct the value of same attribute
CustomerDemographic_data['gender'].replace({'M' : 'Male'}, inplace=True)
CustomerDemographic_data['gender'].replace({'F' : 'Female'}, inplace=True)
CustomerDemographic_data['gender'].replace({'Femal' : 'Female'}, inplace=True)
```

```
# Create age column
CustomerDemographic_data['DOB'] = pd.to_datetime(CustomerDemographic_data['DOB'])
CustomerDemographic_data['age'] = (dt.datetime.now() - CustomerDemographic_data['DOB']) / np.timedelta64(1, 'Y')

# Check the age range (oldest and youngest customers)
print(CustomerDemographic_data['age'].min(), CustomerDemographic_data['age'].max())
```

```
21.544196756681192 179.76510478354004
<ipython-input-571-69d4fd58b041>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
  CustomerDemographic_data.drop(['first_name', 'last_name', 'default', 'job_title', 'deceased_indicator'], axis=1, inplace=True)
<ipython-input-571-69d4fd58b041>:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
  CustomerDemographic_data.dropna(axis=0, inplace=True)
<ipython-input-571-69d4fd58b041>:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
  CustomerDemographic_data['gender'].replace({'M' : 'Male'}, inplace=True)
<ipython-input-571-69d4fd58b041>:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
  CustomerDemographic_data['gender'].replace({'F' : 'Female'}, inplace=True)
<ipython-input-571-69d4fd58b041>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
  CustomerDemographic_data['gender'].replace({'Femal' : 'Female'}, inplace=True)
<ipython-input-571-69d4fd58b041>:18: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
  CustomerDemographic_data['DOB'] = pd.to_datetime(CustomerDemographic_data['DOB'])
<ipython-input-571-69d4fd58b041>:19: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
  CustomerDemographic_data['age'] = (dt.datetime.now() - CustomerDemographic_data['DOB']) / np.timedelta64(1, 'Y')
```

```
CustomerDemographic_data['gender'].value_counts()
```

```
Female    1689
Male      1565
U            1
Name: gender, dtype: int64
```

```
CustomerDemographic_data = CustomerDemographic_data[CustomerDemographic_data['age'] <= 100]
```

```
CustomerDemographic_data = CustomerDemographic_data[CustomerDemographic_data['age'] <= 100]
```

```
CustomerDemographic_data['DOB'] = pd.to_datetime(CustomerDemographic_data['DOB'])
CustomerDemographic_data['age'] = (dt.datetime.now() - CustomerDemographic_data['DOB']) / np.timedelta64(1, 'Y')

# Check the age range (oldest and youngest customers)
print(CustomerDemographic_data['age'].min(), CustomerDemographic_data['age'].max())
```

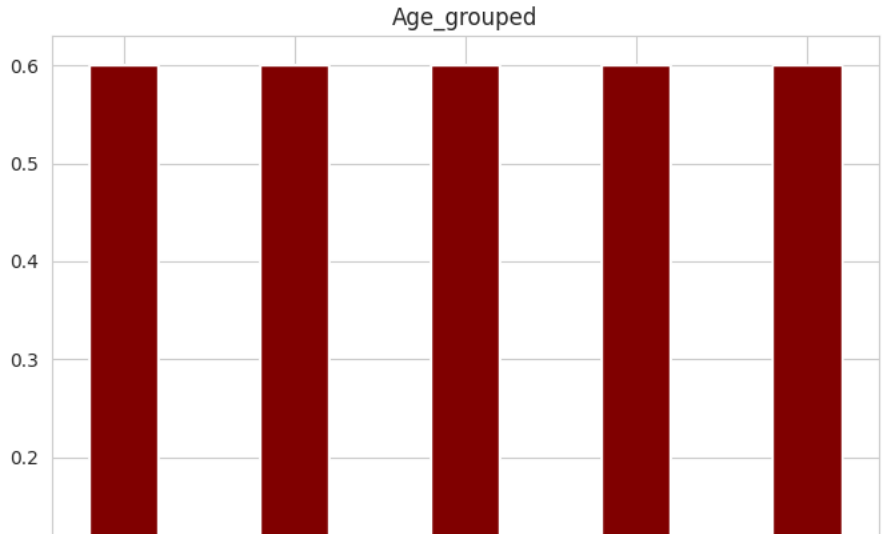
```
21.544196758558368 91.92757218721194
```

```
# Create age group column
ag = pd.Series(['20-34','35-49', '50-64', '65-79', '80-94'], dtype='category')
CustomerDemographic_data['age_group'] = ag

CustomerDemographic_data.loc[CustomerDemographic_data['age']<=34, 'age_group'] = ag[0]
CustomerDemographic_data.loc[(CustomerDemographic_data['age']>34) & (CustomerDemographic_data['age']<=49), 'age_group'] = ag[1]
CustomerDemographic_data.loc[(CustomerDemographic_data['age']>49) & (CustomerDemographic_data['age']<=64), 'age_group'] = ag[2]
CustomerDemographic_data.loc[(CustomerDemographic_data['age']>64) & (CustomerDemographic_data['age']<=79), 'age_group'] = ag[3]
CustomerDemographic_data.loc[CustomerDemographic_data['age']>79, 'age_group'] = ag[4]

print(CustomerDemographic_data.info())
CustomerDemographic_data.head()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3254 entries, 0 to 3998
Data columns (total 10 columns):
 #   Column                               Non-Null Count  Dtype
---  ------                               --------------  -----
 0   customer_id                          3254 non-null   int64
 1   gender                               3254 non-null   object
 2   past_3_years_bike_related_purchases  3254 non-null   int64
 3   DOB                                  3254 non-null   datetime64[ns]
 4   job_industry_category                3254 non-null   object
 5   wealth_segment                       3254 non-null   object
 6   owns_car                             3254 non-null   object
 7   tenure                               3254 non-null   float64
 8   age                                  3254 non-null   float64
 9   age_group                            3254 non-null   category
dtypes: category(1), datetime64[ns](1), float64(2), int64(2), object(4)
memory usage: 257.6+ KB
None
```

	customer_id	gender	past_3_years_bike_related_purchases	DOB	job_industry_category	wealth_segment	owns_car	tenure	age	a
0	1	Female	93	1953-10-12	Health	Mass Customer	Yes	11.0	69.955868	
1	2	Male	81	1980-12-16	Financial Services	Mass Customer	Yes	16.0	42.776666	
2	3	Male	61	1954-01-20	Property	Mass Customer	Yes	15.0	69.682078	
3	4	Male	33	1961-10-03	IT	Mass Customer	No	7.0	61.980345	
5	6	Male	35	1966-09-16	Retail	High Net Worth	Yes	13.0	57.027472	

```
plt.bar(CustomerDemographic_data['age_group'], color ='maroon', height = 0.6,
        width = 0.4)

plt.title("Age_grouped")
plt.show()
```




```
Transactions_data['profit'] = Transactions_data['list_price']-Transactions_data['standard_cost']


# Create recency column
Transactions_data['recency'] = (Transactions_data['transaction_date'].max()-Transactions_data['transaction_date'])/np.tim

print(Transactions_data.info())
Transactions_data.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20000 entries, 0 to 19999
Data columns (total 15 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   transaction_id           20000 non-null  int64
 1   product_id               20000 non-null  int64
 2   customer_id              20000 non-null  int64
 3   transaction_date         20000 non-null  datetime64[ns]
 4   online_order             19640 non-null  float64
 5   order_status             20000 non-null  object
 6   brand                    19803 non-null  object
 7   product_line             19803 non-null  object
 8   product_class            19803 non-null  object
 9   product_size             19803 non-null  object
 10  list_price               20000 non-null  float64
 11  standard_cost            19803 non-null  float64
 12  product_first_sold_date  19803 non-null  datetime64[ns]
 13  profit                   19803 non-null  float64
 14  recency                  20000 non-null  float64
dtypes: datetime64[ns](2), float64(5), int64(3), object(5)
memory usage: 2.3+ MB
None
```

	transaction_id	product_id	customer_id	transaction_date	online_order	order_status	brand	product_line	product_class	product_
0	1	2	2950	2017-02-25	0.0	Approved	Solex	Standard	medium	me
1	2	3	3120	2017-05-21	1.0	Approved	Trek Bicycles	Standard	medium	
2	3	37	402	2017-10-16	0.0	Approved	OHM Cycles	Standard	low	me
3	4	88	3135	2017-08-31	0.0	Approved	Norco Bicycles	Standard	medium	me
4	5	78	787	2017-10-01	1.0	Approved	Giant Bicycles	Standard	medium	

```
# Create RFM score dataframe from Transactions dataframe
rfm_df = Transactions_data.groupby('customer_id').aggregate({'recency':'min', 'customer_id':'count', 'profit':'sum'})
rfm_df.rename(columns={'customer_id':'frequency'}, inplace=True)

rfm_df.head()
```

customer_id	recency	frequency	profit
1	7.0	11	3018.09
2	128.0	3	2226.26
3	102.0	8	3362.81
4	195.0	2	220.57

```
rfm_df['r'] = pd.qcut(rfm_df['recency'], q=4, labels=[4, 3, 2, 1])
rfm_df['f'] = pd.qcut(rfm_df['frequency'], q=4, labels=[1, 2, 3, 4])
rfm_df['m'] = pd.qcut(rfm_df['profit'], q=4, labels=[1, 2, 3, 4])

rfm_df.head()
```

customer_id	recency	frequency	profit	r	f	m
1	7.0	11	3018.09	4	4	3
2	128.0	3	2226.26	1	1	2
3	102.0	8	3362.81	1	4	3
4	195.0	2	220.57	1	1	1
5	16.0	6	2394.94	4	2	2

```
# Create weighted RFM Score column
rfm_df['rfm_score'] = 100*rfm_df['r'].astype(int) + 10*rfm_df['f'].astype(int) + rfm_df['m'].astype(int)

# Create customer profile column
# Customer profile rankings start at Bronze, Silver, Gold and Platinum
rfm_df['customer_profile'] = pd.qcut(rfm_df['rfm_score'], q=4, labels=['Bronze', 'Silver', 'Gold', 'Platinum'])

print(rfm_df.info())
rfm_df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3494 entries, 1 to 5034
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   recency           3494 non-null   float64
 1   frequency         3494 non-null   int64
 2   profit            3494 non-null   float64
 3   r                 3494 non-null   category
 4   f                 3494 non-null   category
 5   m                 3494 non-null   category
 6   rfm_score         3494 non-null   int64
 7   customer_profile  3494 non-null   category
dtypes: category(4), float64(2), int64(2)
memory usage: 150.9 KB
None
```

customer_id	recency	frequency	profit	r	f	m	rfm_score	customer_profile
1	7.0	11	3018.09	4	4	3	443	Platinum
2	128.0	3	2226.26	1	1	2	112	Bronze
3	102.0	8	3362.81	1	4	3	143	Bronze
4	195.0	2	220.57	1	1	1	111	Bronze
5	16.0	6	2394.94	4	2	2	422	Platinum

```
# Correct the value of same attribute
CustomerAddress_data['state'].replace({'New South Wales' : 'NSW'}, inplace=True)
CustomerAddress_data['state'].replace({'Victoria':'VIC' }, inplace=True)
CustomerAddress_data['state'].replace({'Queensland':'QLD'}, inplace=True)


# Check the entries for each column
for col in CustomerAddress_data.columns:
```

```
print('{} : {}'.format(col, CustomerAddress_data[col].unique()))
```

```
 4178 2112 2033 4401 3186 4017 2315 2285 2219 4509 2759 2747 2227 2025
 3191 3025 2263 2154 2119 3016 4113 2032 4352 3020 2116 3057 2099 3749
 2148 3145 2021 2333 2783 2280 4120 3638 2074 2880 2430 4560 2088 2220
 3031 2250 2261 3196 4680 3143 2063 3021 2138 4811 2085 3084 3170 2066
 4078 2222 3152 2159 4655 4220 4012 2015 2776 3011 2761 2502 2110 2566
 2506 2508 2036 2018 3666 4514 2525 4152 2200 3023 3500 2283 2102 2040
 2304 2340 3580 3355 2287 2324 2323 2320 2346 2031 2064 2176 3195 2010
 2768 2752 3802 3071 4735 3081 3205 2165 3125 4214 2030 2767 2798 2117
 4551 3064 3810 4151 2830 3030 4812 2141 3029 2173 3199 4506 2035 3340
 3182 2450 2216 3796 3197 2232 2365 4701 4210 2223 4122 2515 3437 3338
 2203 3356 3223 3032 3105 3101 2763 2089 2126 2147 2251 3127 3087 3131
 3134 2193 4055 2530 4217 4020 2179 3130 4570 3215 3165 3936 2265 3034
 2062 2162 3033 2144 3805 2642 3228 3444 2076 4124 4305 2579 2835 3337
 2536 2770 2646 4558 2211 2570 4552 3082 2360 4223 4370 3150 2549 3040
 4209 4173 2068 3806 2259 2260 4074 4215 3008 2748 2567 3015 4221 3121
 4870 2118 2050 2120 3812 3111 3012 2197 3162 2037 3141 3178 3564 3550
 3775 3004 3185 2447 3910 3621 4702 2133 3075 3911 2487 4123 2151 3156
 3126 2027 3124 2564 2049 3690 3282 2576 3108 4165 2218 2190 2337 2307
 2121 2716 3168 3076 2478 2560 2087 4505 2630 2225 3013 3136 2111 2106
 3930 2291 2305 2300 4179 2539 3207 4340 3137 3068 4580 2041 4504 2306
 4075 3975 4114 4341 4022 4820 3028 4035 2326 3429 4218 2380 3171 2565
 3850 3807 4800 2577 2199 2134 4053 2800 2048 2194 2557 2548 2262 2463
 2481 2072 3155 3095 4207 4110 2017 2769 4562 2571 3480 3551 2753 3093
 4615 3174 2192 2290 3073 4810 4121 2229 4118 2594 3750 4019 2540 2293
 2143 3630 2164 4869 2137 2354 4814 2282 2292 2171 2077 4873 4006 2754
 2152 2067 2122 3976 2000 3175 2156 2228 3187 2680 2527 2163 2336 4415
 3139 3147 3623 4356 2034 2196 2177 2234 2537 4720 3765 2486 4511 2795
 3188 3730 3060 4131 2518 3158 2477 2195 2575 3400 3037 4507 2113 2443
 3177 2641 4212 3241 3428 4227 4034 2558 2460 3173 2289 4500 4825 4161
 3757 2871 2146 4380 3024 3934 3109 2136 2484 3225 4068 2214 3809 2454
 4556 4573 2131 3184 2619 3138 3441 2101 4510 2647 2731 3284 2573 2206
 4815 2526 4502 2710 2090 3038 3142 3442 4064 2529 3039 2061 3644 2104
 2044 3043 2873 2541 2321 2079 3072 2007 2620 2231 2580 2020 2852 3677
 4818 4519 4806 4610 4512 4306 4304 4301 4205 4051 4130 4115 4164 4128
 3940 3939 3840 3804 2582 3747 3620 3151 3380 3280 3224 3222 3226 3190
 3169 3163 3161 3791 4119 3140 3818 3106 3099 3754 3756 3061 3048 3036
 3107 3022 3049 2870 2762 2794 2773 2640 2439 2528 2440 2325 2161 2848
 4878 2775 2224 3219 2441 4877 3264 2681 3018 2043 2107 4567 4011 2221
 2028 3941 3146 3585 4105 2094 2284 3808 3149 3114 3352 3129 4154 2114
 4721 2011 4157 4799 2095 3103 4224 4344 2009 3144 3181 2665 2474 3166
 2820 4059 4160 3189 4030 2042 4032 4103 2868 2358 2449 2299 4014 3056
 2281 4007 2318 3821 4109 4216 2470 2563 2264 2445 4060 3214 4077 2327
 4104 3172 2505 2115 2469 2705 3088 3966 3912 2446 2810 4061 3083 3915
 4153 2267 3458 4421 3505 3006 4037 3918 2335 3079 2400 3148 2092 2714
 3555 2008 3631 4228 4101 2022 4568 3115 4343 3116 2208 4125 4070 3194
 2533 2758 4159 4879 2500 2298 2178 3240 2671 4455 4650 3160 3250 3342
 3523 2350 2785 3995 3351 3000 2462 3041 3260 2534 4565 3618 2780 2167
 3279 4106 4700 2799 4564 4860 3922 2256 4031 2779 4660 4073 3067 2843
 3027 3066 2572 2128 4054 4069 2343 3377 3065 2295 4270 2316 3179 3933
 4883 2081 2471 3053 4285 2546 4275 3132 3042 3085 2071 2821 4390 3793
 3128 2869 3616 4750 3059 3153 2130 3213 3862 3610 4102 2060 3786 2019
 2191 2713 3094 3122 2850 4272 3051 3634 3803 4515 2700 3123 2294 3978
 3305 2198 2778 4730 4155 4000 4225 3556 4575 4715 2429 4555 2132 2045
 2024 4076 3321 3193 2390 2806 2257 4172 3078 3104 3860 2658 3052 4163
 2590 2550 2877 4311 3089]
state : ['NSW' 'QLD' 'VIC']
country : ['Australia']
property_valuation : [10  9  4 12  8  6  7  3  5 11  1  2]
```

```
CustomerDemographic_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3254 entries, 0 to 3998
Data columns (total 10 columns):
 #   Column                               Non-Null Count  Dtype         
---  ------                               --------------  -----         
 0   customer_id                          3254 non-null   int64         
 1   gender                               3254 non-null   object        
 2   past_3_years_bike_related_purchases  3254 non-null   int64         
 3   DOB                                  3254 non-null   datetime64[ns]
 4   job_industry_category                3254 non-null   object        
 5   wealth_segment                       3254 non-null   object        
 6   owns_car                             3254 non-null   object        
 7   tenure                               3254 non-null   float64       
 8   age                                  3254 non-null   float64       
 9   age_group                            3254 non-null   category      
dtypes: category(1), datetime64[ns](1), float64(2), int64(2), object(4)
memory usage: 257.6+ KB
```

```
CustomerAddress_data.columns
```

```
Index(['customer_id', 'address', 'postcode', 'state', 'country',
       'property_valuation'],
      dtype='object')
```

```
# Check duplications
print('Duplicated rows:')
CustomerAddress_data[CustomerAddress_data.duplicated()]
```

Duplicated rows:

customer_id	address	postcode	state	country	property_valuation

```
# Join Customer Demographic, Customer Address and RFM score dataframes on customer id
dataset = rfm_df.merge(CustomerAddress_data.merge(CustomerDemographic_data, how='inner', on='customer_id'), how='inner', on='customer_id')

print(dataset.info())
dataset.head()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2851 entries, 0 to 2850
Data columns (total 23 columns):
 #   Column                               Non-Null Count  Dtype         
---  ------                               --------------  -----         
 0   customer_id                          2851 non-null   int64         
 1   recency                              2851 non-null   float64       
 2   frequency                            2851 non-null   int64         
 3   profit                               2851 non-null   float64       
 4   r                                    2851 non-null   category      
 5   f                                    2851 non-null   category      
 6   m                                    2851 non-null   category      
 7   rfm_score                            2851 non-null   int64         
 8   customer_profile                     2851 non-null   category      
 9   address                              2851 non-null   object        
 10  postcode                             2851 non-null   int64         
 11  state                                2851 non-null   object        
 12  country                              2851 non-null   object        
 13  property_valuation                   2851 non-null   int64         
 14  gender                               2851 non-null   object        
 15  past_3_years_bike_related_purchases  2851 non-null   int64         
 16  DOB                                  2851 non-null   datetime64[ns]
 17  job_industry_category                2851 non-null   object        
 18  wealth_segment                       2851 non-null   object        
 19  owns_car                             2851 non-null   object        
 20  tenure                               2851 non-null   float64       
 21  age                                  2851 non-null   float64       
 22  age_group                            2851 non-null   category      
dtypes: category(5), datetime64[ns](1), float64(4), int64(6), object(7)
memory usage: 438.1+ KB
None
```

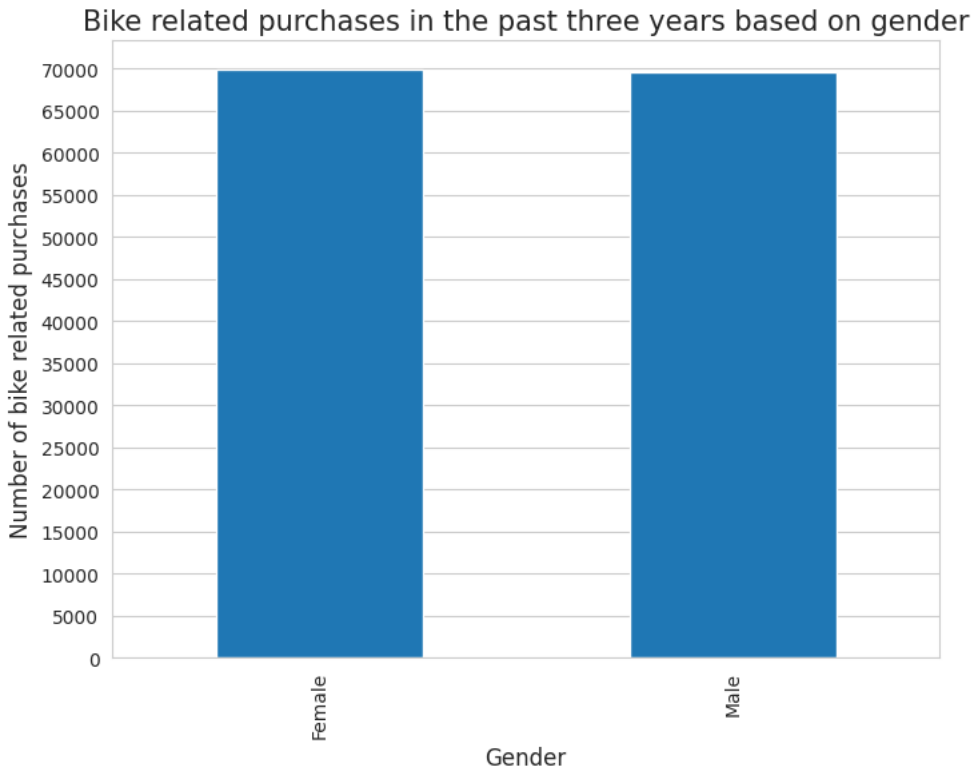
	customer_id	recency	frequency	profit	r	f	m	rfm_score	customer_profile	address	...	property_valuation	gender	past_3_year
0	1	7.0	11	3018.09	4	4	3	443	Platinum	060 Morning Avenue	...	10	Female	
1	2	128.0	3	2226.26	1	1	2	112	Bronze	6 Meadow Vale Court	...	10	Male	
2	4	195.0	2	220.57	1	1	1	111	Bronze	0 Holy Cross Court	...	9	Male	
3	6	64.0	5	3946.55	2	2	3	223	Silver	9 Oakridge Court	...	9	Male	
4	7	253.0	3	220.11	1	1	1	111	Bronze	4 Delaware Trail	...	9	Female	

5 rows × 23 columns

Data Insights

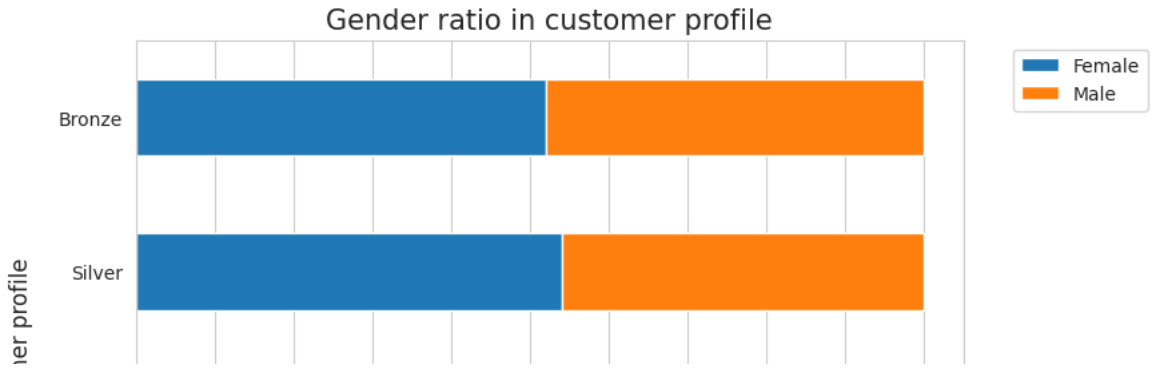
```
df = pd.read_csv('dataset.csv')

# Set grid for all figures
sns.set_style('whitegrid')

plt.rcParams['figure.figsize'] = (8,6)


df1 = df.groupby('gender')['past_3_years_bike_related_purchases'].sum()
df1.plot(kind='bar')
plt.grid(axis='x')
plt.title('Bike related purchases in the past three years based on gender', fontsize=15)
plt.ylabel('Number of bike related purchases', fontsize=12)
plt.yticks(np.arange(0, 75000, 5000))
plt.xlabel('Gender', fontsize=12)
```

```
Text(0.5, 0, 'Gender')
```




```
df2 = df.groupby(['customer_profile', 'gender']).size().unstack()
sort_cp = ['Platinum', 'Gold', 'Silver', 'Bronze']
df2 = df2.loc[sort_cp]

df2.apply(lambda x : x/x.sum(), axis=1).plot(kind='barh', stacked=True)
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(axis='y')
plt.title('Gender ratio in customer profile', fontsize=15)
plt.ylabel('Customer profile', fontsize=12)
plt.xlabel('Gender ratio', fontsize=12)
plt.xticks(np.arange(0, 1+0.1, 0.1))
```

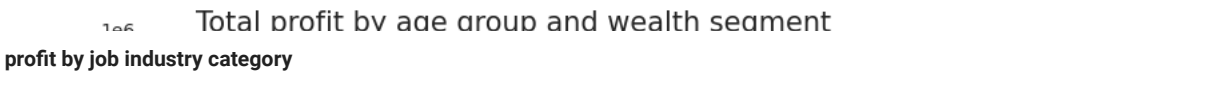
```
([<matplotlib.axis.XTick at 0x7b66b4d254b0>,
  <matplotlib.axis.XTick at 0x7b66b4d26620>,
  <matplotlib.axis.XTick at 0x7b66b4d25ae0>,
  <matplotlib.axis.XTick at 0x7b66b4c61ae0>,
  <matplotlib.axis.XTick at 0x7b66b4c62590>,
  <matplotlib.axis.XTick at 0x7b66b4c627d0>,
  <matplotlib.axis.XTick at 0x7b66b4c63280>,
  <matplotlib.axis.XTick at 0x7b66b4c63d30>,
  <matplotlib.axis.XTick at 0x7b66b4cf0820>,
  <matplotlib.axis.XTick at 0x7b66b4c62e30>,
  <matplotlib.axis.XTick at 0x7b66b4cf11b0>],
 [Text(0.0, 0, '0.0'),
  Text(0.1, 0, '0.1'),
  Text(0.2, 0, '0.2'),
  Text(0.30000000000000004, 0, '0.3'),
  Text(0.4, 0, '0.4'),
  Text(0.5, 0, '0.5'),
  Text(0.6000000000000001, 0, '0.6'),
  Text(0.7000000000000001, 0, '0.7'),
  Text(0.8, 0, '0.8'),
  Text(0.9, 0, '0.9'),
  Text(1.0, 0, '1.0')])
```



Total profit by age group and wealth segment

```
df3 = df.groupby(['age_group', 'wealth_segment'])['profit'].sum().unstack().fillna(0)

df3.plot(kind='bar')
plt.grid(axis = 'x')
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.title('Total profit by age group and wealth segment', fontsize=15)
plt.ylabel('Total profit', fontsize=12)
plt.xlabel('Age group', fontsize=12)
```

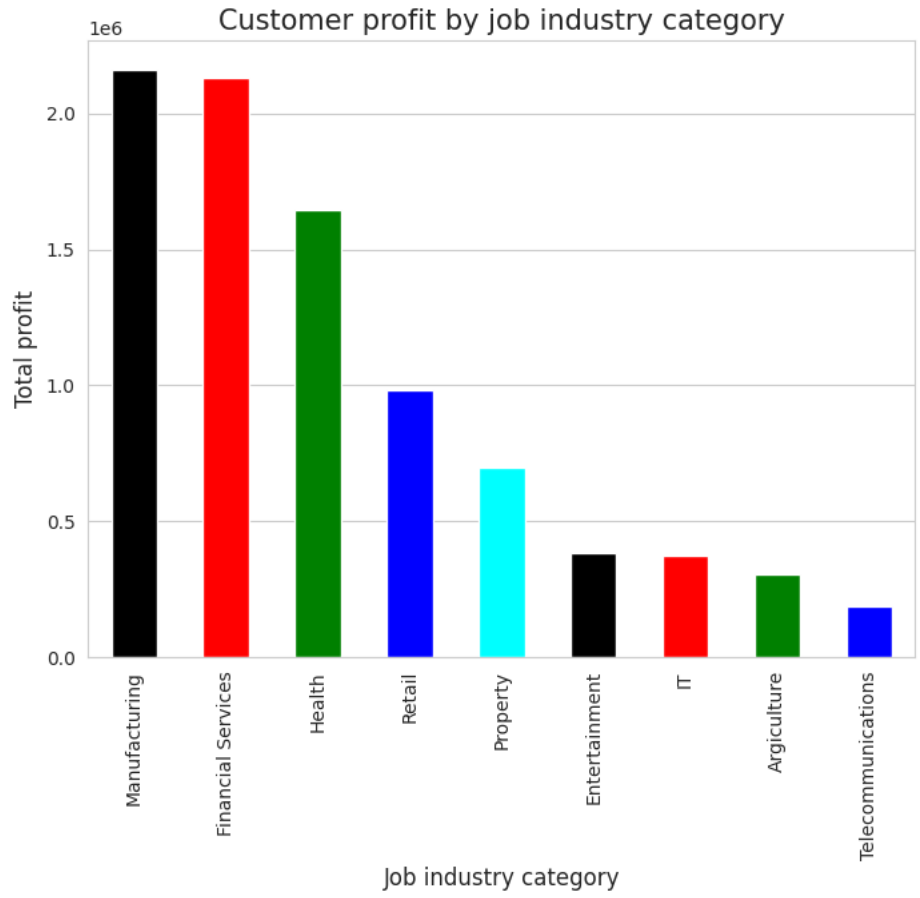
```
Text(0.5, 0, 'Age group')
```



Total profit by job industry category

```
df4 = df.groupby('job_industry_category')['profit'].sum()
df4 = df4.sort_values(ascending=False)

df4.plot(kind='bar', color=['black', 'red', 'green', 'blue', 'cyan'])
plt.grid(axis='x')
plt.title('Customer profit by job industry category', fontsize=15)
plt.ylabel('Total profit', fontsize=12)
plt.xlabel('Job industry category', fontsize=12)
```

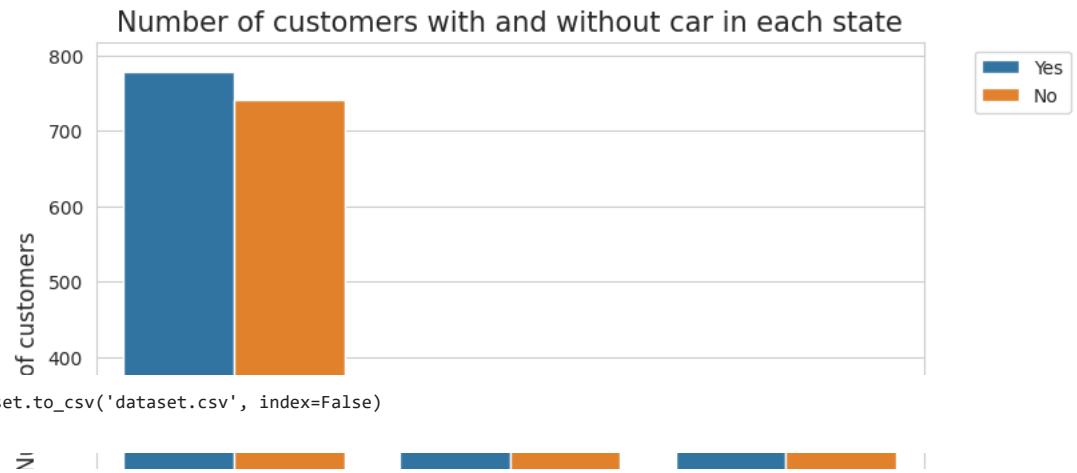
```
Text(0.5, 0, 'Job industry category')
```



Number of customers with and without cars in each state¶

```
sns.countplot(x='state', hue='owns_car', data=df)
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.title('Number of customers with and without car in each state', fontsize=15)
plt.ylabel('Number of customers', fontsize=12)
plt.xlabel('State', fontsize=12)
```

```
Text(0.5, 0, 'State')
```




```
dataset.to_csv('dataset.csv', index=False)
```

```
NewCustomerList_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 18 columns):
 #   Column                               Non-Null Count  Dtype
---  ------                               --------------  -----
 0   first_name                           1000 non-null   object
 1   last_name                            971 non-null    object
 2   gender                               1000 non-null   object
 3   past_3_years_bike_related_purchases  1000 non-null   int64
 4   DOB                                  983 non-null    datetime64[ns]
 5   job_title                            894 non-null    object
 6   job_industry_category                835 non-null    object
 7   wealth_segment                       1000 non-null   object
 8   deceased_indicator                   1000 non-null   object
 9   owns_car                             1000 non-null   object
 10  tenure                               1000 non-null   int64
 11  address                              1000 non-null   object
 12  postcode                             1000 non-null   int64
 13  state                                1000 non-null   object
 14  country                              1000 non-null   object
 15  property_valuation                   1000 non-null   int64
 16  Rank                                 1000 non-null   int64
 17  Value                                1000 non-null   float64
dtypes: datetime64[ns](1), float64(1), int64(5), object(11)
memory usage: 140.8+ KB
```

```
CustomerDemographic_data['job_industry_category'].value_counts()
```

```
Manufacturing         796
Financial Services    767
Health                595
Retail                357
Property              267
IT                    151
Entertainment         136
Argiculture           113
Telecommunications     72
Name: job_industry_category, dtype: int64
```

```
CustomerDemographic_data['wealth_segment'].value_counts()
```

```
Mass Customer        1635
High Net Worth        826
Affluent Customer     793
Name: wealth_segment, dtype: int64
```

```
CustomerDemographic_data.head(5)
```

	customer_id	gender	past_3_years_bike_related_purchases	DOB	job_industry_category	wealth_segment	owns_car	tenure	age	a
0	1	Female	93	1953-10-12	Health	Mass Customer	Yes	11.0	69.955868	
1	2	Male	81	1980-12-16	Financial Services	Mass Customer	Yes	16.0	42.776666	
2	3	Male	61	1954-01-20	Property	Mass Customer	Yes	15.0	69.682078	

```
CustomerDemographic_data['owns_car'].value_counts()
```

```
Yes    1657
No     1597
Name: owns_car, dtype: int64
```

```
CustomerDemographic_data['tenure'].value_counts()
```

```
7.0     196
5.0     188
11.0    184
16.0    184
10.0    181
18.0    176
12.0    173
8.0     173
14.0    164
9.0     160
13.0    158
6.0     157
4.0     156
17.0    151
15.0    142
19.0    138
1.0     136
3.0     135
2.0     124
20.0     81
22.0     50
21.0     47
Name: tenure, dtype: int64
```

```
#customers data
```

```
CustomerAddress_data.head(5)
```

	customer_id	address	postcode	state	country	property_valuation
0	1	060 Morning Avenue	2016	NSW	Australia	10
1	2	6 Meadow Vale Court	2153	NSW	Australia	10
2	4	0 Holy Cross Court	4211	QLD	Australia	9
3	5	17979 Del Mar Point	2448	NSW	Australia	4
4	6	9 Oakridge Court	3216	VIC	Australia	9

```
CustomerAddress_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3999 entries, 0 to 3998
Data columns (total 6 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   customer_id         3999 non-null   int64 
 1   address             3999 non-null   object
 2   postcode            3999 non-null   int64 
 3   state               3999 non-null   object
 4   country             3999 non-null   object
 5   property_valuation  3999 non-null   int64 
dtypes: int64(3), object(3)
memory usage: 187.6+ KB
```

```
CustomerAddress_data.isnull().sum()
```

```
customer_id           0
address               0
postcode              0
state                 0
country               0
property_valuation    0
dtype: int64
```

```
CustomerAddress_data.duplicated().sum()
```

```
0
```

```
CustomerAddress_data.nunique()
```

```
customer_id           3999
address               3996
postcode               873
state                    3
country                  1
property_valuation      12
dtype: int64
```

****END TASK ****