Task-1

Accesing the Data

In [1]:
```
#KPMG provides us multiple Dataset, So loading them first
import pandas as pd
import numpy as np
data = pd.ExcelFile("C:/Users/91913/Downloads/KPMG_VI_New_raw_data_update_final (2).xlsx
```

Reading each file

In [2]:
```
Transactions=pd.read_excel(data,'Transactions')
CustomerDemographic = pd.read_excel(data, 'CustomerDemographic')
CustomerAddress = pd.read_excel(data, 'CustomerAddress')
```

Deleting the names and setting row 1 as the columns

In [3]:
```
Transactions.head()
```

Out[3]:

	Note: The data and information in this document is reflective of a hypothetical situation and client. This document is to be used for KPMG Virtual Internship purposes only.	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unna
0	transaction_id	product_id	customer_id	transaction_date	online_order	order_status	brand	product_line	produ
1	1	2	2950	2017-02-25 00:00:00	False	Approved	Solex	Standard	r
2	2	3	3120	2017-05-21 00:00:00	True	Approved	Trek Bicycles	Standard	r
3	3	37	402	2017-10-16 00:00:00	False	Approved	OHM Cycles	Standard	
4	4	88	3135	2017-08-31 00:00:00	False	Approved	Norco Bicycles	Standard	r

In [4]:
```
Transactions.columns=Transactions.iloc[0]
```

In [5]:
```
Transactions.drop(0,axis=0,inplace=True)
```

First look

In [6]:
```
Transactions
```

Out[6]:

	transaction_id	product_id	customer_id	transaction_date	online_order	order_status	brand	product_line
1	1	2	2950	2017-02-25 00:00:00	False	Approved	Solex	Standard
2	2	3	3120	2017-05-21 00:00:00	True	Approved	Trek Bicycles	Standard
3	3	37	402	2017-10-16 00:00:00	False	Approved	OHM Cycles	Standard
4	4	88	3135	2017-08-31 00:00:00	False	Approved	Norco Bicycles	Standard
5	5	78	787	2017-10-01 00:00:00	True	Approved	Giant Bicycles	Standard
...	...	...	...	...	...	...	...	...
19996	19996	51	1018	2017-06-24 00:00:00	True	Approved	OHM Cycles	Standard
19997	19997	41	127	2017-11-09 00:00:00	True	Approved	Solex	Road
19998	19998	87	2284	2017-04-14 00:00:00	True	Approved	OHM Cycles	Standard
19999	19999	6	2764	2017-07-03 00:00:00	False	Approved	OHM Cycles	Standard
20000	20000	11	1144	2017-09-22 00:00:00	True	Approved	Trek Bicycles	Standard

20000 rows × 13 columns

Looking at the dimensionality

In [7]:
```
Transactions.shape
```

Out[7]:
```
(20000, 13)
```

Looking at the info

In [8]:
```
Transactions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20000 entries, 1 to 20000
Data columns (total 13 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   transaction_id           20000 non-null  object
 1   product_id               20000 non-null  object
 2   customer_id              20000 non-null  object
 3   transaction_date         20000 non-null  object
```

```
 4   online_order             19640 non-null  object
 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  object
 11  standard_cost            19803 non-null  object
 12  product_first_sold_date  19803 non-null  object
dtypes: object(13)
memory usage: 2.0+ MB
```

By looking at the info, the dtype of many columns are not correct.We will change them further.

Looking at the missing values

In [9]:
```python
Transactions.isna().sum()
```

Out[9]:
```
0
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
```

In [10]:
```python
Transactions["online_order"].fillna("na",inplace=True)
Transactions["brand"].fillna("not known",inplace=True)
Transactions["product_line"].fillna("not known",inplace=True)
Transactions["product_class"].fillna("not known",inplace=True)
Transactions["product_size"].fillna("not known",inplace=True)
Transactions["standard_cost"].fillna(Transactions["standard_cost"].mean(),inplace=True)
Transactions["product_first_sold_date"].fillna("0",inplace=True)
```

In [11]:
```python
Transactions.isnull().sum()
```

Out[11]:
```
0
transaction_id             0
product_id                 0
customer_id                0
transaction_date           0
online_order               0
order_status               0
brand                      0
product_line               0
product_class              0
product_size               0
list_price                 0
standard_cost              0
product_first_sold_date    0
dtype: int64
```

In [12]:
```python
Transactions.duplicated().sum()
```

Out[12]:
```
0
```

There are no duplicate values, all the data is unique.

Recasting the Data type

In [13]:
```
Transactions=Transactions.astype({"transaction_id":"int64",
                                    "product_id":"int64",
                                    "customer_id":"int64",
                                    "list_price":"int64",
                                    "standard_cost":"int64",









                                      })
```

In [14]:
```
Transactions['transaction_date'] = pd.to_datetime(Transactions['transaction_date'])
```

In [15]:
```
Transactions
```

Out[15]:

	transaction_id	product_id	customer_id	transaction_date	online_order	order_status	brand	product_line
1	1	2	2950	2017-02-25	False	Approved	Solex	Standard
2	2	3	3120	2017-05-21	True	Approved	Trek Bicycles	Standard
3	3	37	402	2017-10-16	False	Approved	OHM Cycles	Standard
4	4	88	3135	2017-08-31	False	Approved	Norco Bicycles	Standard
5	5	78	787	2017-10-01	True	Approved	Giant Bicycles	Standard
...	...	...	...	...	...	...	...	...
19996	19996	51	1018	2017-06-24	True	Approved	OHM Cycles	Standard
19997	19997	41	127	2017-11-09	True	Approved	Solex	Road
19998	19998	87	2284	2017-04-14	True	Approved	OHM Cycles	Standard
19999	19999	6	2764	2017-07-03	False	Approved	OHM Cycles	Standard
20000	20000	11	1144	2017-09-22	True	Approved	Trek Bicycles	Standard

20000 rows × 13 columns

In [16]:
```
Transactions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20000 entries, 1 to 20000
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             20000 non-null  object
 5   order_status             20000 non-null  object
 6   brand                    20000 non-null  object
 7   product_line             20000 non-null  object
 8   product_class            20000 non-null  object
 9   product_size             20000 non-null  object
 10  list_price               20000 non-null  int64
 11  standard_cost            20000 non-null  int64
 12  product_first_sold_date  20000 non-null  object
dtypes: datetime64[ns](1), int64(5), object(7)
memory usage: 2.0+ MB
```

looking at the unique values in the Data

In [17]:
```
Transactions.nunique()
```

Out[17]:
```
0
transaction_id             20000
product_id                   101
customer_id                 3494
transaction_date             364
online_order                   3
order_status                   2
brand                          7
product_line                   5
product_class                  4
product_size                   4
list_price                   274
standard_cost                 97
product_first_sold_date      101
dtype: int64
```

Exploring the columns

In [18]:
```
Transactions.columns
```

Out[18]:
```
Index(['transaction_id', 'product_id', 'customer_id', 'transaction_date',
       'online_order', 'order_status', 'brand', 'product_line',
       'product_class', 'product_size', 'list_price', 'standard_cost',
       'product_first_sold_date'],
      dtype='object', name=0)
```

In [19]:
```
Transactions['brand'].value_counts()
```

Out[19]:
```
Solex             4253
Giant Bicycles    3312
WeareA2B          3295
OHM Cycles        3043
Trek Bicycles     2990
Norco Bicycles    2910
not known          197
Name: brand, dtype: int64
```

In [20]:
```
Transactions['order_status'].value_counts()
```

Out[20]:
```
Approved     19821
Cancelled      179
Name: order_status, dtype: int64
```

In [21]:
```
Transactions['product_line'].value_counts()
```

Out[21]:
```
Standard     14176
Road          3970
Touring       1234
Mountain       423
not known      197
Name: product_line, dtype: int64
```

In [22]:
```
Transactions['product_class'].value_counts()
```

Out[22]:
```
medium       13826
high          3013
low           2964
not known      197
Name: product_class, dtype: int64
```

In [23]:
```
Transactions['online_order'].value_counts()
```

Out[23]:
```
True     9829
False    9811
na        360
Name: online_order, dtype: int64
```

In [24]:
```
Transactions['product_first_sold_date'].value_counts()
```

Out[24]:
```
33879    234
41064    229
37823    227
39880    222
38216    220
        ...
41848    169
42404    168
41922    166
37659    163
34586    162
Name: product_first_sold_date, Length: 101, dtype: int64
```

In [ ]:

Adding profit and profit% columns

In [25]:
```
Transactions["profit"]=Transactions["list_price"]-Transactions["standard_cost"]
Transactions["profit_percentage"]=(Transactions["list_price"]-Transactions["standard_cos
```

Exploring the Data in Customer Demographic Data set

In [26]:
```
cd=pd.read_excel(data,"CustomerDemographic")
```

In [27]:
```
cd
```

Out[27]:

Note: The data and information in this document is reflective of	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6

a hypothetical situation and client. This document is to be used for KPMG Virtual Internship purposes only.							
0	customer_id	first_name	last_name	gender	past_3_years_bike_related_purchases	DOB	job_title
1	1	Laraine	Medendorp	F	93	1953-10-12 00:00:00	Executive Secretary
2	2	Eli	Bockman	Male	81	1980-12-16 00:00:00	Administrative Officer
3	3	Arlin	Dearle	Male	61	1954-01-20 00:00:00	Recruiting Manager
4	4	Talbot	NaN	Male	33	1961-10-03 00:00:00	NaN
...	...	...	...	...	...	...	...
3996	3996	Rosalia	Halgarth	Female	8	1975-08-09 00:00:00	VP Product Management
3997	3997	Blanch	Nisuis	Female	87	2001-07-13 00:00:00	Statistician I
3998	3998	Sarene	Woolley	U	60	NaN	Assistant Manager
3999	3999	Patrizius	NaN	Male	11	1973-10-24 00:00:00	NaN
4000	4000	Kippy	Oldland	Male	76	1991-11-05 00:00:00	Software Engineer IV

4001 rows × 13 columns

In [28]:
```
cd.columns=cd.iloc[0]
```

In [29]:
```
cd=cd.drop(0,axis=0)
```

In [30]:
```
cd.tail()
```

Out[30]:

	customer_id	first_name	last_name	gender	past_3_years_bike_related_purchases	DOB	job_title	job_
3996	3996	Rosalia	Halgarth	Female	8	1975-08-09 00:00:00	VP Product Management	

3997	3997	Blanch	Nisuis	Female	87	2001-07-13 00:00:00	Statistician II
3998	3998	Sarene	Woolley	U	60	NaN	Assistant Manager
3999	3999	Patrizius	NaN	Male	11	1973-10-24 00:00:00	NaN
4000	4000	Kippy	Oldland	Male	76	1991-11-05 00:00:00	Software Engineer IV

In [31]:
```
cd.shape
```
Out[31]:
```
(4000, 13)
```

In [32]:
```
cd.duplicated().sum()
```
Out[32]:
```
0
```

No duplicate value presents

In [33]:
```
cd.describe().T
```
Out[33]:

0	count	unique	top	freq
customer_id	4000	4000	1	1
first_name	4000	3139	Max	5
last_name	3875	3725	Pristnor	3
gender	4000	6	Female	2037
past_3_years_bike_related_purchases	4000	100	16	56
DOB	3913	3448	1978-01-30 00:00:00	7
job_title	3494	195	Business Systems Development Analyst	45
job_industry_category	3344	9	Manufacturing	799
wealth_segment	4000	3	Mass Customer	2000
deceased_indicator	4000	2	N	3998
default	3698	90	100	113
owns_car	4000	2	Yes	2024
tenure	3913	22	7	235

In [34]:
```
cd.info()
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 1 to 4000
Data columns (total 13 columns):
 #   Column                               Non-Null Count  Dtype
---  ------                               --------------  -----
 0   customer_id                          4000 non-null   object
 1   first_name                           4000 non-null   object
```

```
 2   last_name                          3875 non-null   object
 3   gender                             4000 non-null   object
 4   past_3_years_bike_related_purchases  4000 non-null   object
 5   DOB                                3913 non-null   object
 6   job_title                          3494 non-null   object
 7   job_industry_category              3344 non-null   object
 8   wealth_segment                     4000 non-null   object
 9   deceased_indicator                 4000 non-null   object
 10  default                            3698 non-null   object
 11  owns_car                           4000 non-null   object
 12  tenure                             3913 non-null   object
dtypes: object(13)
memory usage: 406.4+ KB
```

unique values

In [35]:
```
cd.nunique()
```

Out[35]:
```
0
customer_id                            4000
first_name                             3139
last_name                              3725
gender                                    6
past_3_years_bike_related_purchases     100
DOB                                    3448
job_title                               195
job_industry_category                     9
wealth_segment                            3
deceased_indicator                        2
default                                  90
owns_car                                  2
tenure                                   22
dtype: int64
```

missing values

In [36]:
```
cd.isnull().sum()
```

Out[36]:
```
0
customer_id                              0
first_name                               0
last_name                              125
gender                                   0
past_3_years_bike_related_purchases      0
DOB                                     87
job_title                              506
job_industry_category                  656
wealth_segment                           0
deceased_indicator                       0
default                                302
owns_car                                 0
tenure                                  87
dtype: int64
```

In [37]:
```
cd["last_name"].fillna("not_known",inplace=True)
cd["job_title"].fillna("not_known",inplace=True)
cd["job_industry_category"].fillna("not_known",inplace=True)
cd["tenure"].fillna(int(0),inplace=True)
```

In [38]:
```
cd.isnull().sum()
```

Out[38]:
```
0
customer_id                            0
first_name                             0
last_name                              0
gender                                 0
past_3_years_bike_related_purchases    0
DOB                                   87
job_title                              0
job_industry_category                  0
wealth_segment                         0
deceased_indicator                     0
default                              302
owns_car                               0
tenure                                 0
dtype: int64
```

We will delete the Column "default" because of having non-readable values.

In [39]:
```
cd.drop("default",axis=1,inplace=True)
```

In [40]:
```
cd
```

Out[40]:

	customer_id	first_name	last_name	gender	past_3_years_bike_related_purchases	DOB	job_title	j
1	1	Laraine	Medendorp	F	93	1953-10-12 00:00:00	Executive Secretary	
2	2	Eli	Bockman	Male	81	1980-12-16 00:00:00	Administrative Officer	
3	3	Arlin	Dearle	Male	61	1954-01-20 00:00:00	Recruiting Manager	
4	4	Talbot	not_known	Male	33	1961-10-03 00:00:00	not_known	
5	5	Sheila-kathryn	Calton	Female	56	1977-05-13 00:00:00	Senior Editor	
...	...	...	...	...	...	...	...	
3996	3996	Rosalia	Halgarth	Female	8	1975-08-09 00:00:00	VP Product Management	
3997	3997	Blanch	Nisuis	Female	87	2001-07-13 00:00:00	Statistician II	
3998	3998	Sarene	Woolley	U	60	NaN	Assistant Manager	
3999	3999	Patrizius	not_known	Male	11	1973-10-24 00:00:00	not_known	
4000	4000	Kippy	Oldland	Male	76	1991-11-05 00:00:00	Software Engineer IV	

4000 rows × 12 columns

In [41]:
```
cd.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 1 to 4000
Data columns (total 12 columns):
 #   Column                               Non-Null Count  Dtype 
---  ------                               --------------  ----- 
 0   customer_id                          4000 non-null   object
 1   first_name                           4000 non-null   object
 2   last_name                            4000 non-null   object
 3   gender                               4000 non-null   object
 4   past_3_years_bike_related_purchases  4000 non-null   object
 5   DOB                                  3913 non-null   object
 6   job_title                            4000 non-null   object
 7   job_industry_category                4000 non-null   object
 8   wealth_segment                       4000 non-null   object
 9   deceased_indicator                   4000 non-null   object
 10  owns_car                             4000 non-null   object
 11  tenure                               4000 non-null   int64 
dtypes: int64(1), object(11)
memory usage: 375.1+ KB
```

Recasting the data types

In [42]:
```
cd['DOB']=pd.to_datetime(cd['DOB'])
```

In [43]:
```
cd=cd.astype({'past_3_years_bike_related_purchases':'int64',
              'tenure':'float64','customer_id':"int64"



             })
```

In [44]:
```
cd.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 1 to 4000
Data columns (total 12 columns):
 #   Column                               Non-Null Count  Dtype         
---  ------                               --------------  -----         
 0   customer_id                          4000 non-null   int64         
 1   first_name                           4000 non-null   object        
 2   last_name                            4000 non-null   object        
 3   gender                               4000 non-null   object        
 4   past_3_years_bike_related_purchases  4000 non-null   int64         
 5   DOB                                  3913 non-null   datetime64[ns]
 6   job_title                            4000 non-null   object        
 7   job_industry_category                4000 non-null   object        
 8   wealth_segment                       4000 non-null   object        
 9   deceased_indicator                   4000 non-null   object        
 10  owns_car                             4000 non-null   object        
 11  tenure                               4000 non-null   float64       
dtypes: datetime64[ns](1), float64(1), int64(2), object(8)
memory usage: 375.1+ KB
```

In [45]:
```
cd.isnull().sum()
```

Out[45]:
```
0
customer_id                          0
```

```
first_name                              0
last_name                               0
gender                                  0
past_3_years_bike_related_purchases     0
DOB                                    87
job_title                               0
job_industry_category                   0
wealth_segment                          0
deceased_indicator                      0
owns_car                                0
tenure                                  0
dtype: int64
```

In [46]:
```
cd.dropna(inplace=True)
```

Exploring the columns

In [47]:
```
cd.columns
```

Out[47]:
```
Index(['customer_id', 'first_name', 'last_name', 'gender',
       'past_3_years_bike_related_purchases', 'DOB', 'job_title',
       'job_industry_category', 'wealth_segment', 'deceased_indicator',
       'owns_car', 'tenure'],
      dtype='object', name=0)
```

In [48]:
```
cd ['gender'].value_counts()
```

Out[48]:
```
Female    2037
Male      1872
F            1
U            1
Femal        1
M            1
Name: gender, dtype: int64
```

In [49]:
```
cd["gender"]=cd["gender"].str.replace("Femaleemale","Female")
```

In [50]:
```
cd ['gender'].value_counts()
```

Out[50]:
```
Female    2037
Male      1872
F            1
U            1
Femal        1
M            1
Name: gender, dtype: int64
```

In [51]:
```
cd.drop(cd.index[(cd["gender"] == "F")],axis=0,inplace=True)
```

In [52]:
```
cd.drop(cd.index[(cd["gender"] == "Femal")],axis=0,inplace=True)
```

In [53]:
```
cd.drop(cd.index[(cd["gender"] == "M")],axis=0,inplace=True)
```

In [54]:
```
cd ['gender'].value_counts()
```

Out[54]:
```
Female    2037
Male      1872
U            1
Name: gender, dtype: int64
```

In [55]:
```
cd["gender"]=cd["gender"].str.replace("U","others")
```

In [56]:
```
cd ['gender'].value_counts()
```
Out[56]:
```
Female    2037
Male      1872
others       1
Name: gender, dtype: int64
```
In [57]:
```
cd ['job_title'].value_counts()
```
Out[57]:
```
not_known                               497
Tax Accountant                           43
Business Systems Development Analyst     43
Social Worker                            42
Recruiting Manager                       41
                                       ...
Database Administrator I                  4
Health Coach I                            3
Health Coach III                          3
Research Assistant III                    3
Developer I                               1
Name: job_title, Length: 196, dtype: int64
```
In [58]:
```
cd ['job_industry_category'].value_counts()
```
Out[58]:
```
Manufacturing         796
Financial Services    767
not_known             655
Health                595
Retail                358
Property              266
IT                    152
Entertainment         136
Argiculture           113
Telecommunications     72
Name: job_industry_category, dtype: int64
```
In [59]:
```
cd ['wealth_segment'].value_counts()
```
Out[59]:
```
Mass Customer        1951
High Net Worth        996
Affluent Customer     963
Name: wealth_segment, dtype: int64
```
In [60]:
```
cd ['deceased_indicator'].value_counts()
```
Out[60]:
```
N    3908
Y       2
Name: deceased_indicator, dtype: int64
```
In [61]:
```
cd ['owns_car'].value_counts()
```
Out[61]:
```
Yes    1971
No     1939
Name: owns_car, dtype: int64
```
In [62]:
```
cd ['tenure'].value_counts()
```
Out[62]:
```
7.0     235
5.0     228
11.0    220
10.0    218
16.0    215
8.0     211
18.0    207
12.0    202
9.0     200
14.0    200
```

```
6.0      192
4.0      191
13.0     190
17.0     182
15.0     179
1.0      166
3.0      160
19.0     159
2.0      150
20.0      96
22.0      55
21.0      54
Name: tenure, dtype: int64
```

Adding the Age Column

In [63]:

```
from datetime import datetime,date as dt

cd["year"]=cd["DOB"].dt.year
today=dt.today()
cd.astype({"year":"int64"})
cd["age"]=today.year-cd["year"]
cd.drop("year",axis=1,inplace=True)
```

In [64]:

```
cd
```

Out[64]:

	customer_id	first_name	last_name	gender	past_3_years_bike_related_purchases	DOB	job_title	job_
2	2	Eli	Bockman	Male	81	1980-12-16	Administrative Officer	
3	3	Arlin	Dearle	Male	61	1954-01-20	Recruiting Manager	
4	4	Talbot	not_known	Male	33	1961-10-03	not_known	
5	5	Sheila-kathryn	Calton	Female	56	1977-05-13	Senior Editor	
6	6	Curr	Duckhouse	Male	35	1966-09-16	not_known	
...	...	...	...	...	...	...	...	
3995	3995	Rusty	Iapico	Male	93	1975-12-12	Staff Scientist	
3996	3996	Rosalia	Halgarth	Female	8	1975-08-09	VP Product Management	
3997	3997	Blanch	Nisuis	Female	87	2001-07-13	Statistician II	
3999	3999	Patrizius	not_known	Male	11	1973-10-24	not_known	
4000	4000	Kippy	Oldland	Male	76	1991-11-05	Software Engineer IV	

3910 rows × 13 columns

Customer address data

In [65]:
```
ca=pd.read_excel(data,'CustomerAddress')
```

In [66]:
```
ca
```

Out[66]:

	Note: The data and information in this document is reflective of a hypothetical situation and client. This document is to be used for KPMG Virtual Internship purposes only.	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5
0	customer_id	address	postcode	state	country	property_valuation
1	1	060 Morning Avenue	2016	New South Wales	Australia	10
2	2	6 Meadow Vale Court	2153	New South Wales	Australia	10
3	4	0 Holy Cross Court	4211	QLD	Australia	9
4	5	17979 Del Mar Point	2448	New South Wales	Australia	4
...	...	...	...	...	...	...
3995	3999	1482 Hauk Trail	3064	VIC	Australia	3
3996	4000	57042 Village Green Point	4511	QLD	Australia	6
3997	4001	87 Crescent Oaks Alley	2756	NSW	Australia	10
3998	4002	8194 Lien Street	4032	QLD	Australia	7
3999	4003	320 Acker Drive	2251	NSW	Australia	7

4000 rows × 6 columns

In [67]:
```
ca.columns=ca.iloc[0]
```

In [68]:
```
ca.drop(0,axis=0,inplace=True)
```

In [69]:
```
ca.head()
```

Out[69]:

	customer_id	address	postcode	state	country	property_valuation
1	1	060 Morning Avenue	2016	New South Wales	Australia	10
2	2	6 Meadow Vale Court	2153	New South Wales	Australia	10
3	4	0 Holy Cross Court	4211	QLD	Australia	9

4	5	17979 Del Mar Point	2448	New South Wales	Australia	4
5	6	9 Oakridge Court	3216	VIC	Australia	9

In [70]:
```
ca.nunique()
```

Out[70]:
```
0
customer_id           3999
address               3996
postcode               873
state                    5
country                  1
property_valuation      12
dtype: int64
```

In [71]:
```
ca.duplicated().sum()
```

Out[71]:
```
0
```

No duplicates present

In [72]:
```
ca.shape
```

Out[72]:
```
(3999, 6)
```

In [73]:
```
ca.describe()
```

Out[73]:

	customer_id	address	postcode	state	country	property_valuation
count	3999	3999	3999	3999	3999	3999
unique	3999	3996	873	5	1	12
top	1	3 Mariners Cove Terrace	2170	NSW	Australia	9
freq	1	2	31	2054	3999	647

In [74]:
```
ca.info()
```

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

Missing values

In [75]:
```
ca.isnull().sum()
```

Out[75]:
```
0
customer_id           0
```

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

In [76]:
```
ca.isna().sum()
```

Out[76]:
```
0
customer_id          0
address              0
postcode             0
state                0
country              0
property_valuation   0
dtype: int64
```

No missing or na values

Recasting the data types

In [77]:
```
ca=ca.astype({"customer_id":"int64","postcode":"int64","property_valuation":"int64"})
```

In [78]:
```
ca.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3999 entries, 1 to 3999
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
```

Exploring the columns

In [79]:
```
ca.columns
```

Out[79]:
```
Index(['customer_id', 'address', 'postcode', 'state', 'country',
       'property_valuation'],
      dtype='object', name=0)
```

In [80]:
```
ca['address'].value_counts()
```

Out[80]:
```
3 Mariners Cove Terrace      2
3 Talisman Place             2
64 Macpherson Junction       2
359 Briar Crest Road         1
4543 Service Terrace         1
                            ..
5063 Shopko Pass             1
09 Hagan Pass                1
87897 Lighthouse Bay Pass    1
```

```
294 Lawn Junction            1
320 Acker Drive              1
Name: address, Length: 3996, dtype: int64
```

In [81]:
```
ca['state'].value_counts()
```

Out[81]:
```
NSW                2054
VIC                 939
QLD                 838
New South Wales      86
Victoria             82
Name: state, dtype: int64
```

Changing the names,Victoria to VIC and New South Wales to NSW

In [82]:
```
ca['state'].replace('New South Wales', 'NSW', inplace=True)
ca['state'].replace('Victoria', 'VIC', inplace=True)
ca.dropna(inplace=True)
ca
```

Out[82]:

	customer_id	address	postcode	state	country	property_valuation
1	1	060 Morning Avenue	2016	NSW	Australia	10
2	2	6 Meadow Vale Court	2153	NSW	Australia	10
3	4	0 Holy Cross Court	4211	QLD	Australia	9
4	5	17979 Del Mar Point	2448	NSW	Australia	4
5	6	9 Oakridge Court	3216	VIC	Australia	9
...	...	...	...	...	...	...
3995	3999	1482 Hauk Trail	3064	VIC	Australia	3
3996	4000	57042 Village Green Point	4511	QLD	Australia	6
3997	4001	87 Crescent Oaks Alley	2756	NSW	Australia	10
3998	4002	8194 Lien Street	4032	QLD	Australia	7
3999	4003	320 Acker Drive	2251	NSW	Australia	7

3999 rows × 6 columns

In [83]:
```
ca['state'].value_counts()
```

Out[83]:
```
NSW    2140
VIC    1021
QLD     838
Name: state, dtype: int64
```

All the columns have correct information.

TASK -2

Sprocket Central Pty Ltd has given us a new list of 1000 potential customers with their demographics and attributes.

In [84]:
```
Ncl=pd.read_excel(data,"NewCustomerList")
```

In [85]:
```
Ncl
```

Out[85]:

	Note: The data and information in this document is reflective of a hypothetical situation and client. This document is to be used for KPMG Virtual Internship purposes only.	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	
0	first_name	last_name	gender	past_3_years_bike_related_purchases	DOB	job_title	job_in
1	Chickie	Brister	Male	86	1957-07-12	General Manager	
2	Morly	Genery	Male	69	1970-03-22	Structural Engineer	
3	Ardelis	Forrester	Female	10	1974-08-28 00:00:00	Senior Cost Accountant	Fi
4	Lucine	Stutt	Female	64	1979-01-28	Account Representative III	
...	...	...	...	...	...	...	
996	Ferdinand	Romanetti	Male	60	1959-10-07	Paralegal	Fi
997	Burk	Wortley	Male	22	2001-10-17	Senior Sales Associate	
998	Melloney	Temby	Female	17	1954-10-05	Budget/Accounting Analyst IV	Fi
999	Dickie	Cubbini	Male	30	1952-12-17	Financial Advisor	Fi
1000	Sylas	Duffill	Male	56	1955-10-02	Staff Accountant IV	

1001 rows × 23 columns

In [86]:
```
Ncl.columns=Ncl.iloc[0]
```

In [87]:
```
Ncl.drop(0,axis=0,inplace=True)
```

In [88]:
```
Ncl.head()
```

Out[88]:

	first_name	last_name	gender	past_3_years_bike_related_purchases	DOB	job_title	job_industry_catego
1	Chickie	Brister	Male	86	1957-07-12	General Manager	Manufacturir
2	Morly	Genery	Male	69	1970-03-22	Structural Engineer	Proper
3	Ardelis	Forrester	Female	10	1974-08-28 00:00:00	Senior Cost Accountant	Financial Servic
4	Lucine	Stutt	Female	64	1979-01-28	Account Representative III	Manufacturir
5	Melinda	Hadlee	Female	34	1965-09-21	Financial Analyst	Financial Servic

5 rows × 23 columns

Dropping unknown columns

In [89]:

```
Ncl.drop(Ncl.columns[[16,17,18,19,20]], axis=1, inplace=True)
```

In [90]:

```
Ncl.shape
```

Out[90]:

```
(1000, 18)
```

In [91]:

```
Ncl.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 1 to 1000
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   object
 4   DOB                                  983 non-null    object
 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   object
 11  address                              1000 non-null   object
 12  postcode                             1000 non-null   object
 13  state                                1000 non-null   object
 14  country                              1000 non-null   object
 15  property_valuation                   1000 non-null   object
 16  Rank                                 1000 non-null   object
 17  Value                                1000 non-null   object
dtypes: object(18)
memory usage: 140.8+ KB
```

In [92]:

```
Ncl.describe().T
```

Out[92]:

	count	unique	top	freq
0				

first_name	1000	940	Rozamond	3
last_name	971	961	Sissel	2
gender	1000	3	Female	513
past_3_years_bike_related_purchases	1000	100	60	20
DOB	983	961	1965-07-03	2
job_title	894	184	Associate Professor	15
job_industry_category	835	9	Financial Services	203
wealth_segment	1000	3	Mass Customer	508
deceased_indicator	1000	1	N	1000
owns_car	1000	2	No	507
tenure	1000	23	9	79
address	1000	1000	45 Shopko Center	1
postcode	1000	522	2145	9
state	1000	3	NSW	506
country	1000	1	Australia	1000
property_valuation	1000	16	9	173
Rank	1000	324	760	13
Value	1000.0	324.0	0.6375	13.0

In [93]:
```
Ncl.nunique()
```

Out[93]:
```
0
first_name                            940
last_name                             961
gender                                  3
past_3_years_bike_related_purchases   100
DOB                                   961
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                     16
Rank                                  324
Value                                 324
dtype: int64
```

In [94]:
```
Ncl.isnull().sum()
```

Out[94]:
```
0
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
```

In [95]:
```
Ncl["last_name"].fillna("Not known",inplace=True)
Ncl["job_title"].fillna("Not known",inplace=True)
Ncl["job_industry_category"].fillna("Not known",inplace=True)
```

In [96]:
```
Ncl.isnull().sum()
```

Out[96]:
```
0
first_name                            0
last_name                             0
gender                                0
past_3_years_bike_related_purchases   0
DOB                                  17
job_title                             0
job_industry_category                 0
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
```

In [97]:
```
Ncl['DOB']= pd.to_datetime(Ncl["DOB"])
Ncl.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 1 to 1000
Data columns (total 18 columns):
 #   Column                               Non-Null Count  Dtype
---  ------                               --------------  -----
 0   first_name                           1000 non-null   object
 1   last_name                            1000 non-null   object
 2   gender                               1000 non-null   object
 3   past_3_years_bike_related_purchases  1000 non-null   object
 4   DOB                                  983 non-null    datetime64[ns]
 5   job_title                            1000 non-null   object
 6   job_industry_category                1000 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   object
 11  address                              1000 non-null   object
 12  postcode                             1000 non-null   object
 13  state                                1000 non-null   object
 14  country                              1000 non-null   object
 15  property_valuation                   1000 non-null   object
 16  Rank                                 1000 non-null   object
 17  Value                                1000 non-null   object
```

```
dtypes: datetime64[ns](1), object(17)
memory usage: 140.8+ KB
```

In [98]:
```
Ncl=Ncl.astype({"past_3_years_bike_related_purchases":"int64",
                                    "tenure":"int64",
                                    "postcode":"int64",
                                    "property_valuation":"int64",
                                    "Value":"float64" ,
                                    "Rank":"int64",


                                      })
```

In [99]:
```
Ncl.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 1 to 1000
Data columns (total 18 columns):
 #   Column                               Non-Null Count  Dtype
---  ------                               --------------  -----
 0   first_name                           1000 non-null   object
 1   last_name                            1000 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                            1000 non-null   object
 6   job_industry_category                1000 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
```

In [100…
```
Ncl.duplicated().sum()
```

Out[100]:
```
0
```

No duplicated values found.

Exploring the columns

In [101…
```
Ncl.columns
```

Out[101]:
```
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', name=0)
```

In [102…
```
Ncl['gender'].value_counts()
```

Out[102]:
```
Female    513
Male      470
```

```
U          17
Name: gender, dtype: int64
```

In [103…

```
Ncl["gender"]=Ncl["gender"].str.replace("U","others")
```

In [104…

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

Out[104]:

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

In [105…

```
Ncl.dropna(inplace=True)
```

In [106…

```
 from datetime import datetime,date as dt
Ncl['Year'] = Ncl['DOB'].dt.year
today=dt.today()
today.year
```

Out[106]:

```
2023
```

In [107…

```
Ncl.astype({"Year":"int64"})
Ncl["age"]=today.year-Ncl["Year"]
```

In [108…

```
Ncl
```

Out[108]:

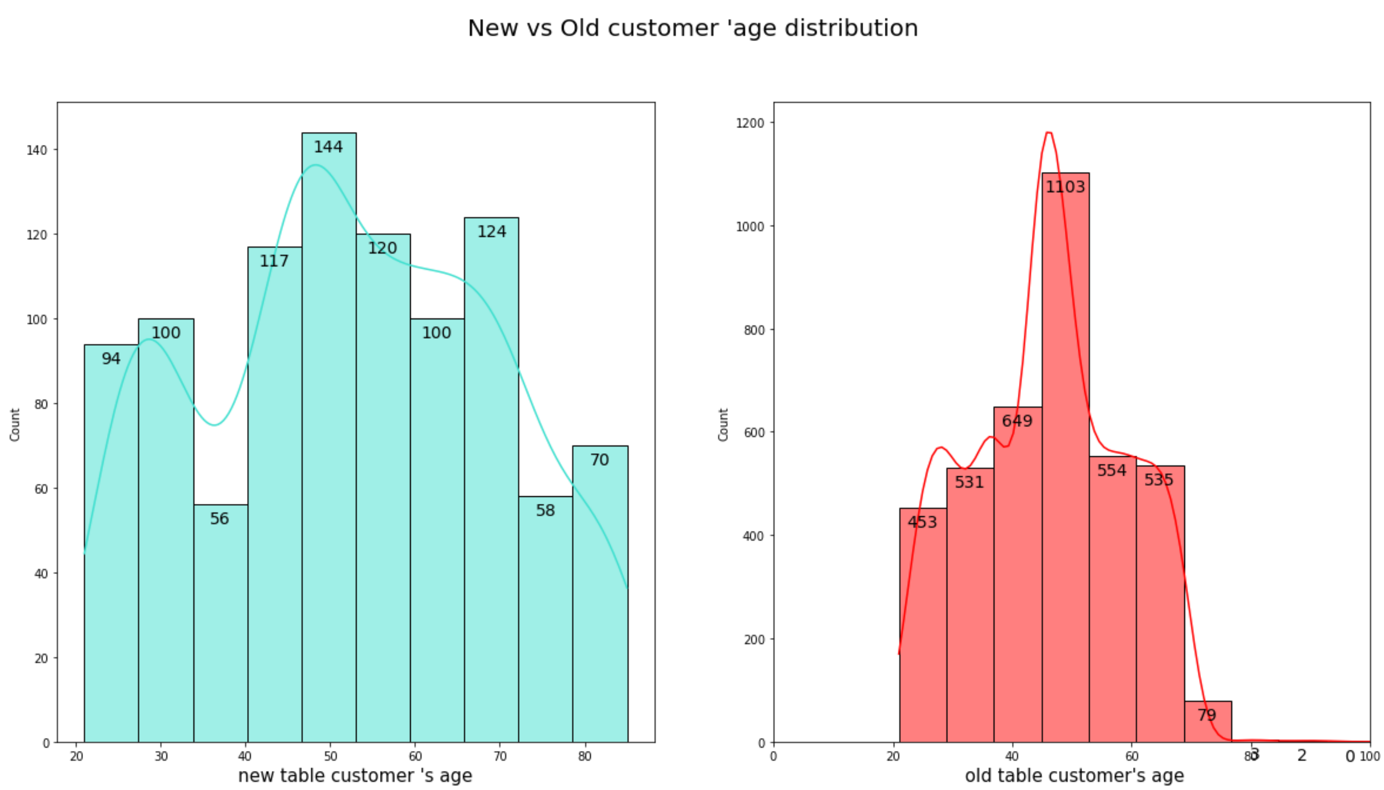
	first_name	last_name	gender	past_3_years_bike_related_purchases	DOB	job_title	job_industry_c
1	Chickie	Brister	Male	86	1957-07-12	General Manager	Manuf
2	Morly	Genery	Male	69	1970-03-22	Structural Engineer	
3	Ardelis	Forrester	Female	10	1974-08-28	Senior Cost Accountant	Financial
4	Lucine	Stutt	Female	64	1979-01-28	Account Representative III	Manuf
5	Melinda	Hadlee	Female	34	1965-09-21	Financial Analyst	Financial
...	...	...	...	...	...	...	
996	Ferdinand	Romanetti	Male	60	1959-10-07	Paralegal	Financial
997	Burk	Wortley	Male	22	2001-10-17	Senior Sales Associate	
998	Melloney	Temby	Female	17	1954-10-05	Budget/Accounting Analyst IV	Financial
999	Dickie	Cubbini	Male	30	1952-12-17	Financial Advisor	Financial
1000	Sylas	Duffill	Male	56	1955-10-02	Staff Accountant IV	

983 rows × 20 columns

Comparing the new customer and old customer table

In [109…

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(20,10))
plt.subplot(121)

c=sns.histplot(data=Ncl,bins=10,x="age",color="turquoise",kde=True)
for i in c.patches:
  c.annotate(format(round(i.get_height()), '.0f'),
                   (i.get_x() + i.get_width() / 2., i.get_height()),
                   ha='center', va='center',color='black',
                   size=14,
                   xytext=(0, -12),
                   textcoords='offset points')

plt.xlabel("new table customer 's age",fontsize=15)

plt.subplot(122)
d=sns.histplot(data=cd,bins=20,x="age",color="red",kde=True)
for i in d.patches:
  d.annotate(format(round(i.get_height()), '.0f'),
                   (i.get_x() + i.get_width() / 2., i.get_height()),
                   ha='center', va='center',color='black',
                   size=14,
                   xytext=(0, -12),
                   textcoords='offset points')

plt.xlim(0, 100)
plt.xlabel(" old table customer's age",fontsize=15)
plt. suptitle("New vs Old customer 'age distribution",fontsize=20)
plt.show()
```



1)Most customer's age is in between 40-49 in new. In old, most of the people 's age is also between 40-49. 2)The lowest group of age in old table is 80-100. 3)The lowest group in "new " table is 35-40. 4)There is a steep drop of customers in new table between 32-39. 5) Age group from 50-60 are considered most populated in both the tables.

Create a pivot table to make further visualisations

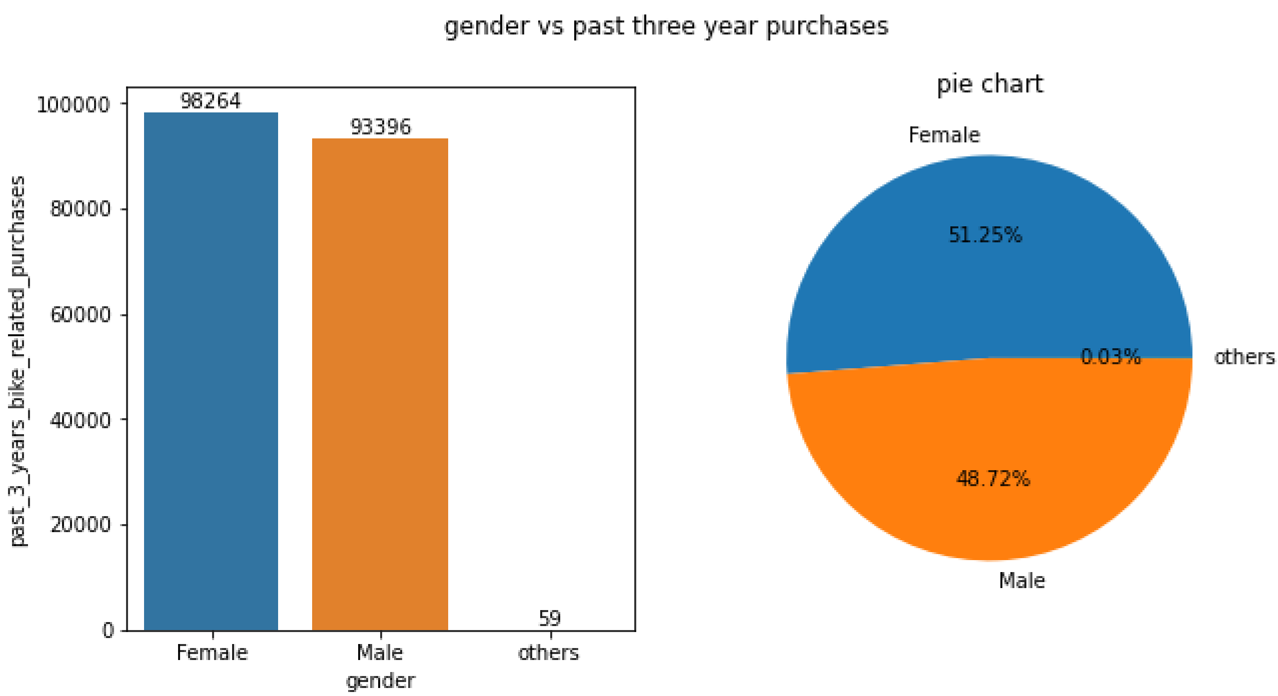
In [110…
```
table=pd.pivot_table(data=cd,index="gender",aggfunc="sum").reset_index("gender")
table
```

Out[110]:

	gender	age	customer_id	past_3_years_bike_related_purchases	tenure
0	Female	93693	4130807	98264	21708.0
1	Male	85843	3692409	93396	19931.0
2	others	180	34	59	20.0

In [111…
```
plt.figure(figsize=(10,5))
plt.subplot(121)
d=sns.barplot(data=table,x="gender",y="past_3_years_bike_related_purchases")
for i in d.patches:
  d.annotate(format(round(i.get_height()), '.0f'),
                   (i.get_x() + i.get_width() / 2., i.get_height()),
                   ha='center', va='center',color="black",
                   size=10,
                   xytext=(0, 5),
                   textcoords='offset points')

plt.subplot(122)

plt.pie(data=table, x="past_3_years_bike_related_purchases",autopct='%.2f%%',labels=["Fe
plt.suptitle("gender vs past three year purchases")
plt.title("pie chart")
```

Out[111]:
```
Text(0.5, 1.0, 'pie chart')
```



Job distribution in old vs New table

In [112…
```
data=Ncl["job_industry_category"].value_counts()
```

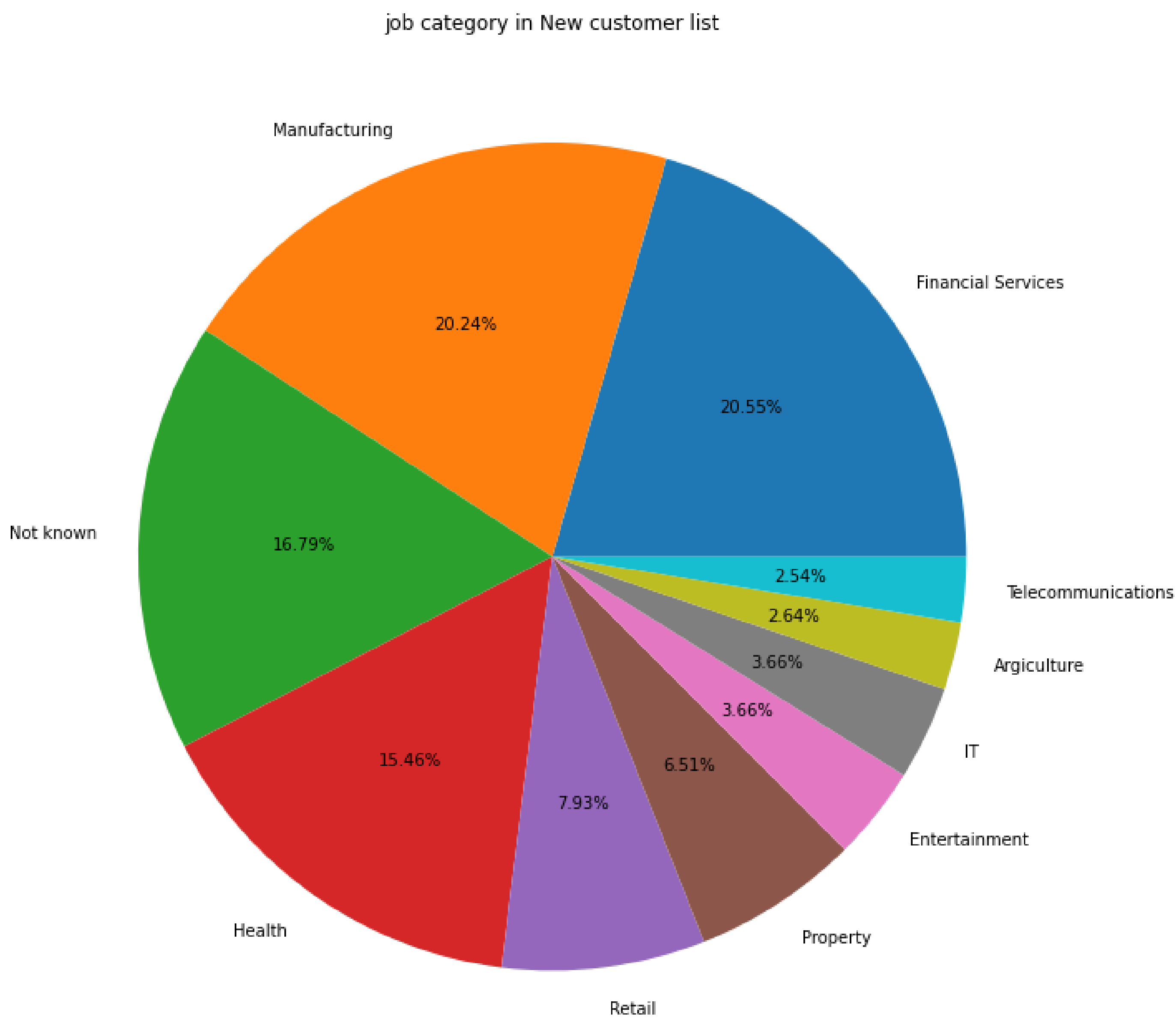
```
data
```

Out[112]:

```
Financial Services    202
Manufacturing         199
Not known             165
Health                152
Retail                 78
Property               64
Entertainment          36
IT                     36
Argiculture            26
Telecommunications     25
Name: job_industry_category, dtype: int64
```

In [113…

```
keys=[202,199,165,152,78,64,36,36,26,25]
plt.figure(figsize=(10,10))
plt.pie(keys,labels=["Financial Services","Manufacturing", "Not known" , "Health " , "Re

plt.title("job category in New customer list")
plt.tight_layout()
```



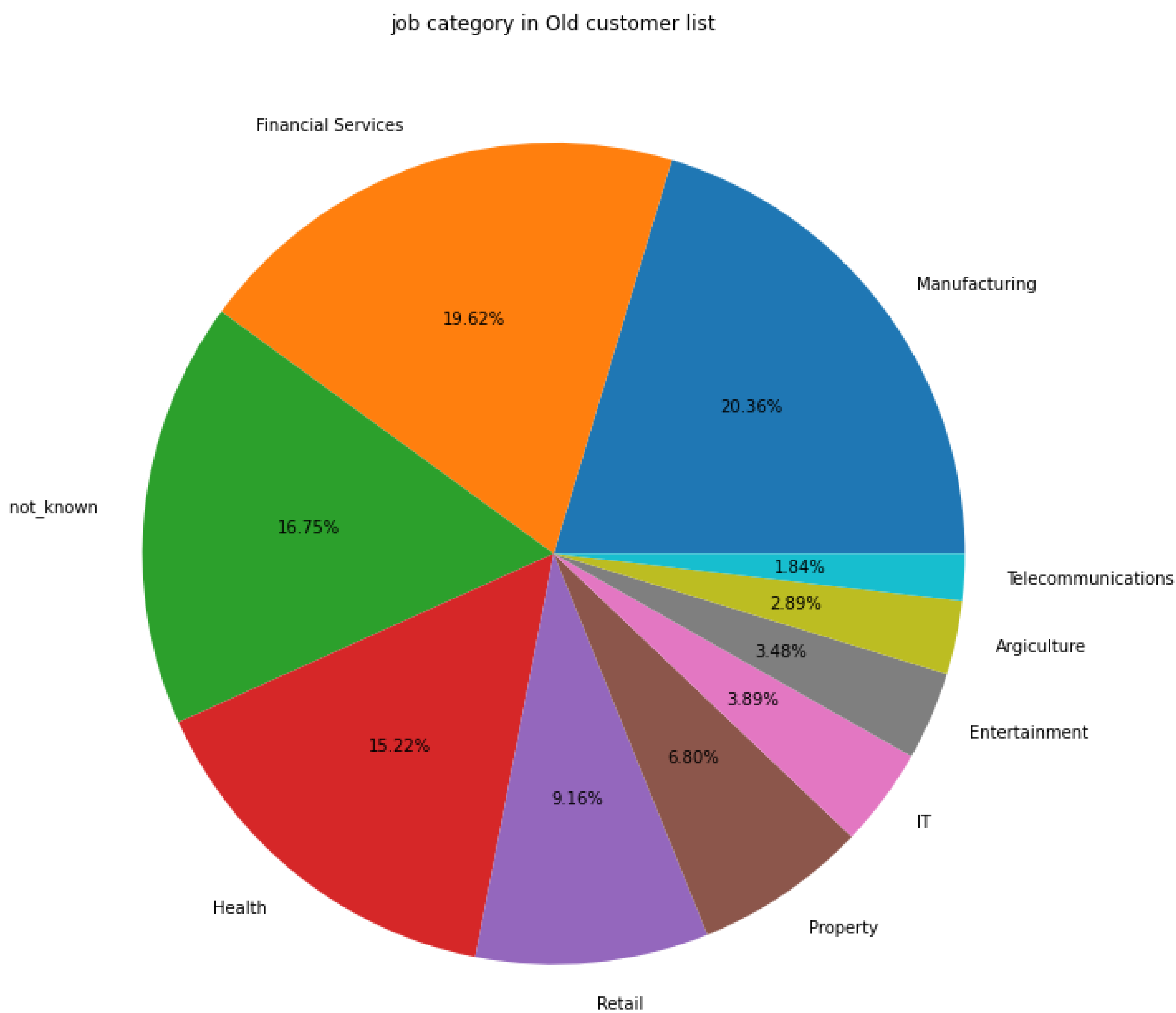
1)20.55% of new customers are involved in Financial services which is the highest in new table. 2)Agriculture and Telecommunications are lowest in new customer table. 17% of jobs are still unidentified.

In [114…

```
data=cd["job_industry_category"].value_counts()
data
```

```
Out[114]: Manufacturing         796
          Financial Services    767
          not_known             655
          Health                595
          Retail                358
          Property              266
          IT                    152
          Entertainment         136
          Argiculture           113
          Telecommunications     72
          Name: job_industry_category, dtype: int64
```

```
In [115…: keys=[796,767,655,595,358,266,152,136,113,72]
          labels=['Manufacturing ' ,'Financial Services ','not_known '
                 ,'Health ','Retail','Property','IT','Entertainment' ,'Argiculture',"Telecommunica
          plt.figure(figsize=(10,10))
          plt.pie(keys,labels=labels ,autopct="%.2f%%")

          plt.title("job category in Old customer list")
          plt.tight_layout()
```



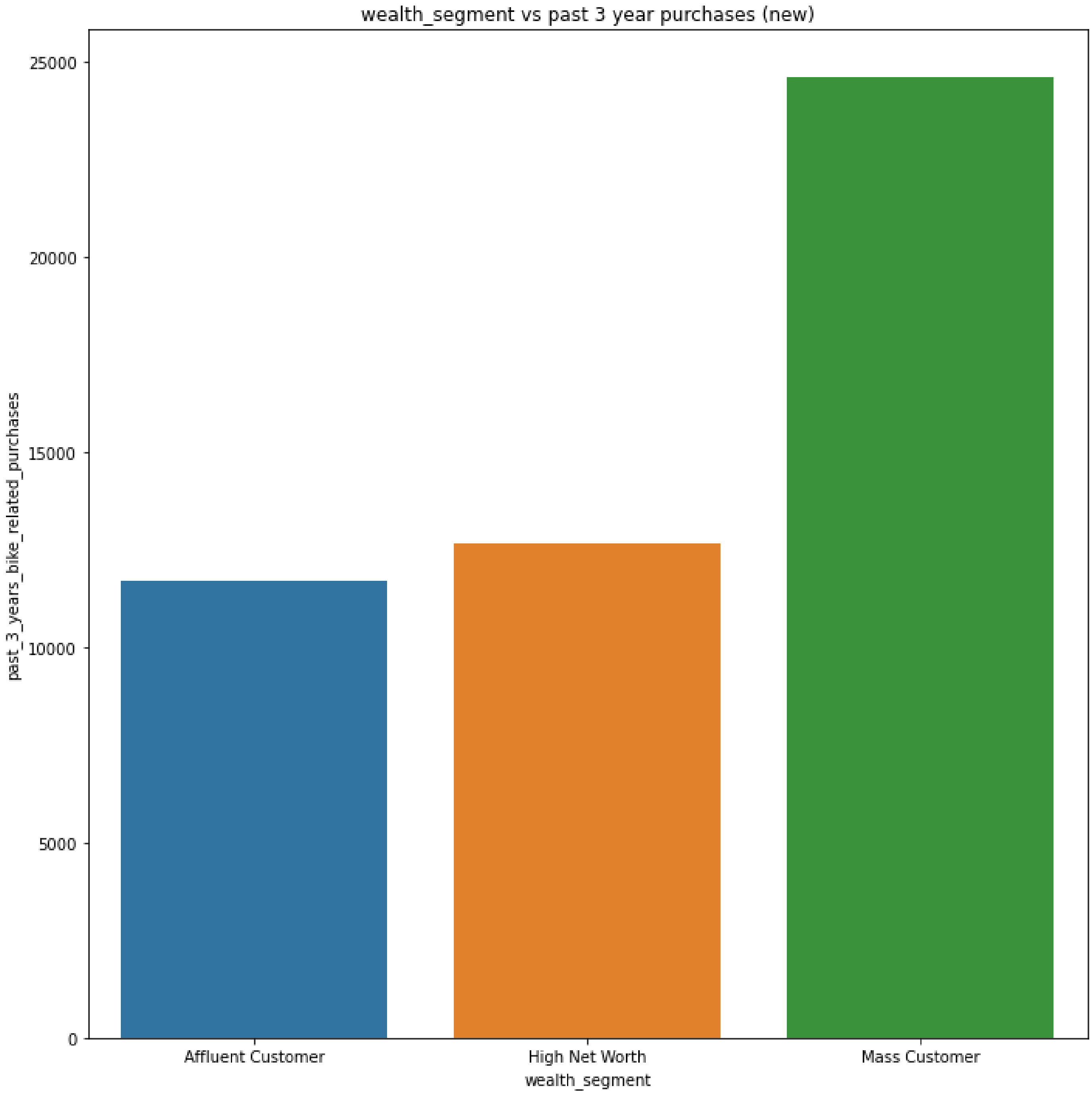
Similar pattern obtained as of new list, here, the manufacturing consist of 20.36% and Financial services consist of 19.62%.

In [116…

```
data=Ncl.groupby("wealth_segment").sum().reset_index()
```

In [117…

```
plt.figure(figsize=(10,10))
sns.barplot(data=data,x='wealth_segment',y='past_3_years_bike_related_purchases')
plt.title('wealth_segment vs past 3 year purchases (new)')
plt.tight_layout()
```



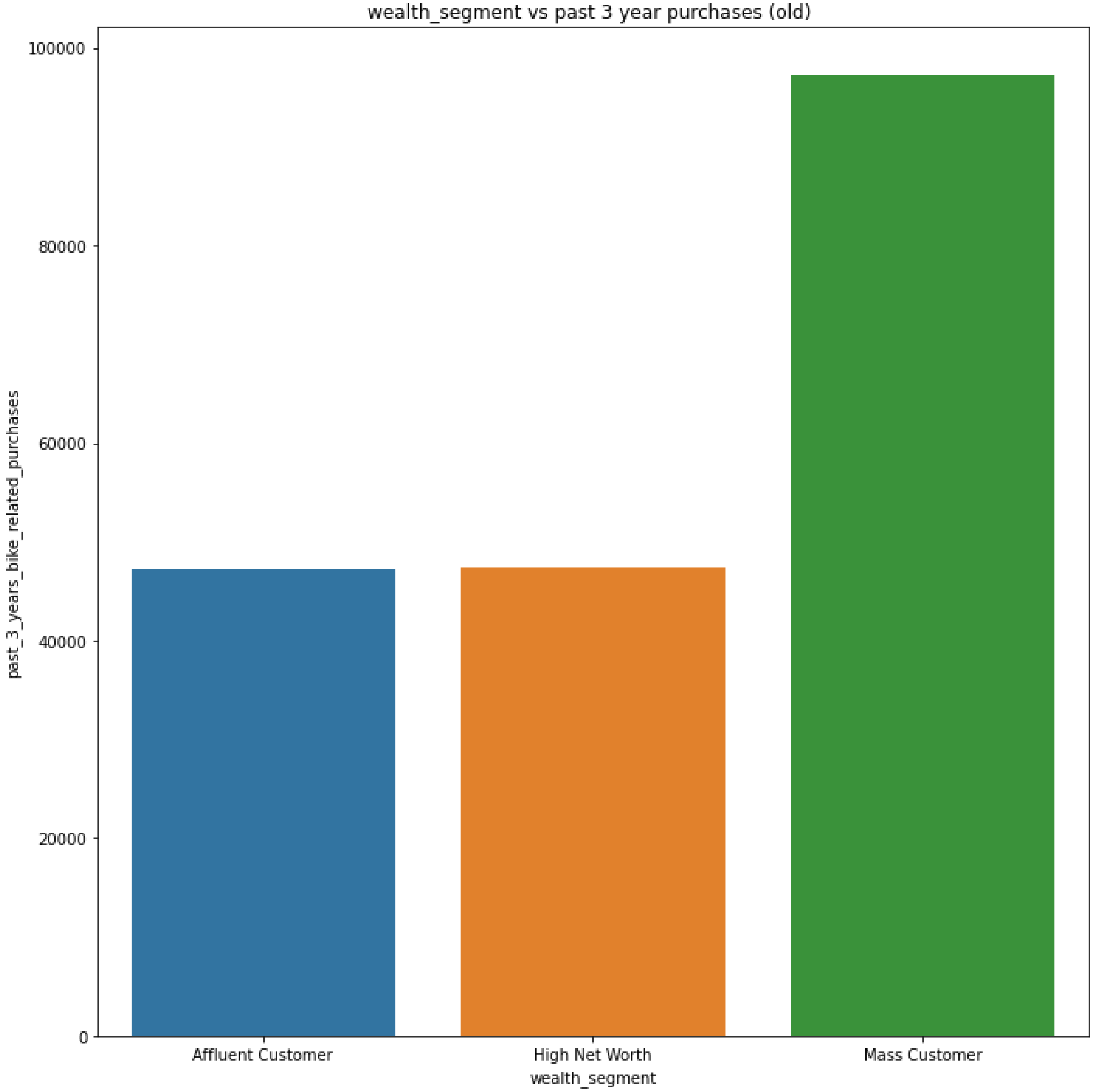
Mass customer has the highest purchases in last three years.Affluent and high net worth customers some what share same records.

In [118…

```
data=cd.groupby("wealth_segment").sum().reset_index()
```

In [119…

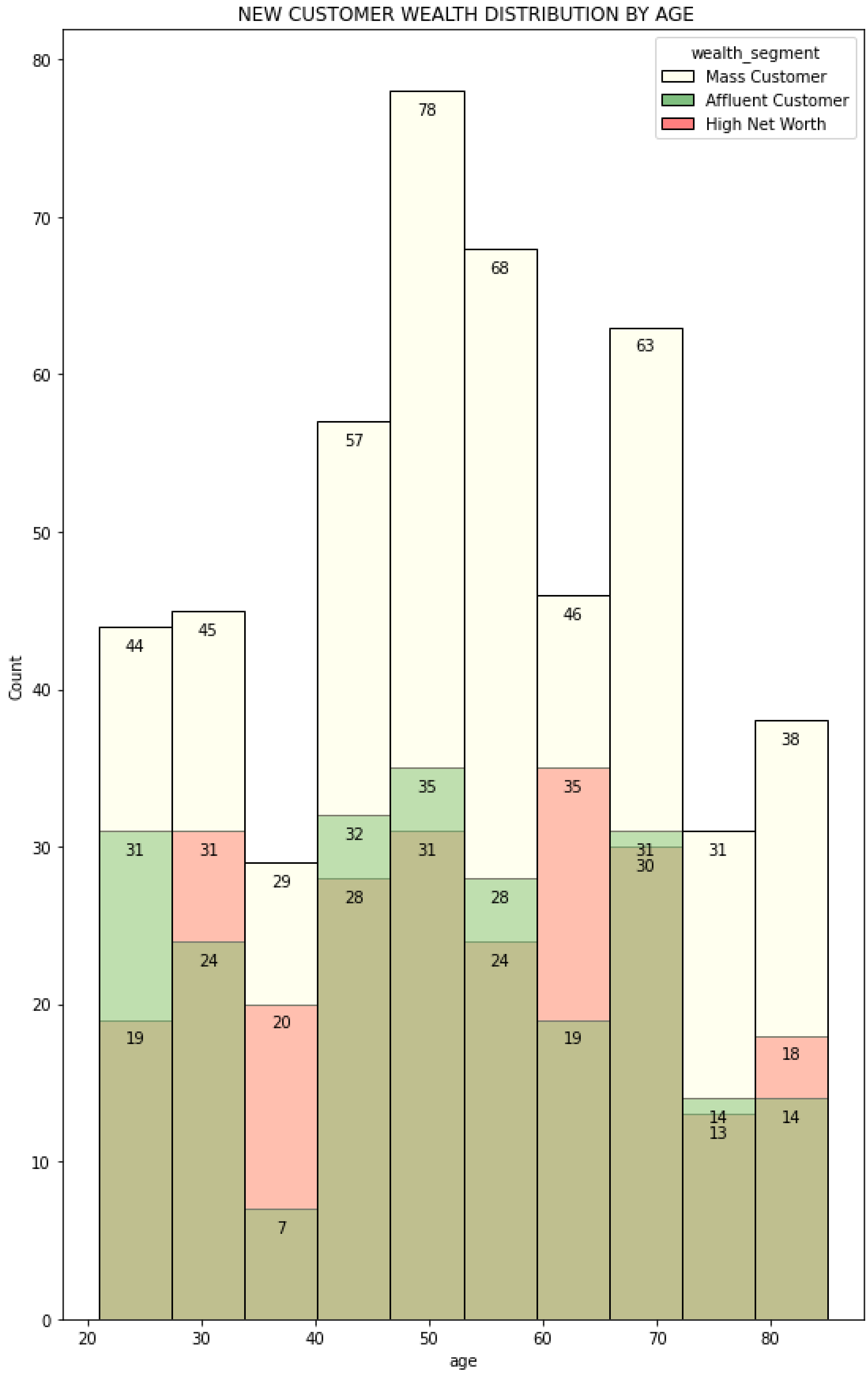
```
plt.figure(figsize=(10,10))
sns.barplot(data=data,x='wealth_segment',y='past_3_years_bike_related_purchases')
plt.title('wealth_segment vs past 3 year purchases (old) ')
plt.tight_layout()
```



Affluent customers and High net worht customers have similar purchases in last three years in the old data.

In [120…

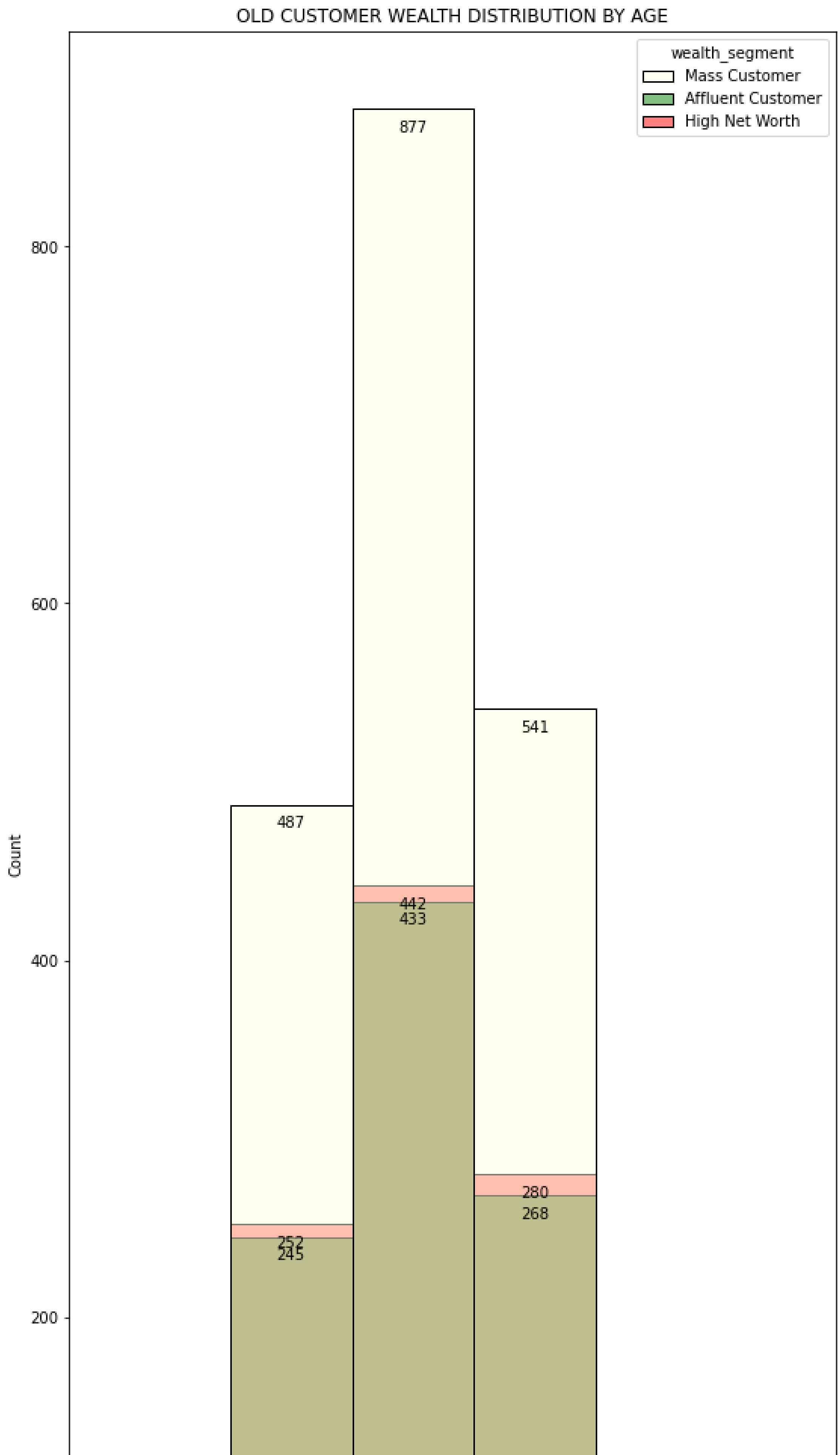
```
plt.figure(figsize=(20,15))
plt.subplot(121)

c=sns.histplot(data=Ncl,bins=10,x="age",hue="wealth_segment",palette=['lightyellow','gre
for i in c.patches:
  c.annotate(format(round(i.get_height()), '.0f'),
                   (i.get_x() + i.get_width() / 2., i.get_height()),
                   ha='center', va='center',color='black',
                   size=10,
                   xytext=(0, -12),
                   textcoords='offset points')
plt.title(' NEW CUSTOMER WEALTH DISTRIBUTION BY AGE')
plt.show()
```

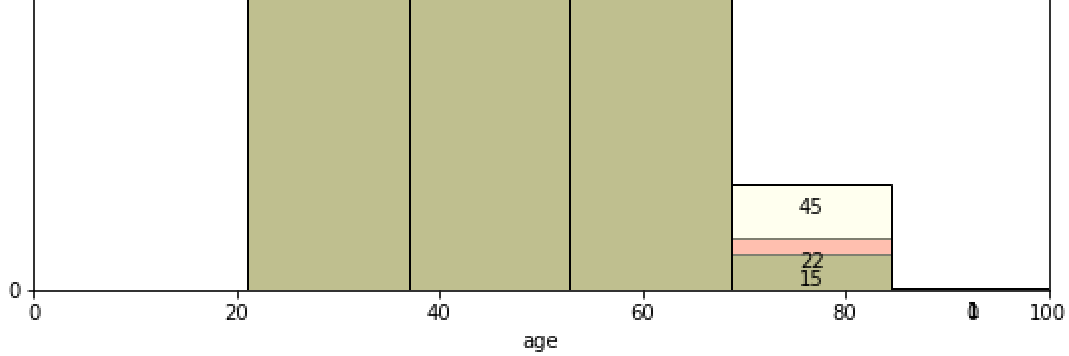


In [121…

```
plt.figure(figsize=(20,20))
plt.subplot(121)
np.sqrt(cd["age"])
c=sns.histplot(data=cd,bins=10,x="age",hue="wealth_segment",palette=['lightyellow','gree
for i in c.patches:
  c.annotate(format(round(i.get_height()), '.0f'),
                   (i.get_x() + i.get_width() / 2., i.get_height()),
                   ha='center', va='center',color='black',
                   size=10,
```

```
                      xytext=(0, -12),textcoords='offset points'
                      )
plt.xlim(0,100)
plt.title("OLD CUSTOMER WEALTH DISTRIBUTION BY AGE")
plt.show()
```





In both the table, mass customers have high purchases and there age is ranging at 45-55.

In [122…

```
data=pd.DataFrame(Ncl.groupby('state')["owns_car"].value_counts())
data.rename(columns={"owns_car":"count"},inplace=True)
data.reset_index("state",inplace=True)

data.reset_index("owns_car",inplace=True)
data
```

Out[122]:

	owns_car	state	count
0	No	NSW	267
1	Yes	NSW	232
2	Yes	QLD	125
3	No	QLD	101
4	No	VIC	129
5	Yes	VIC	129

In [123…

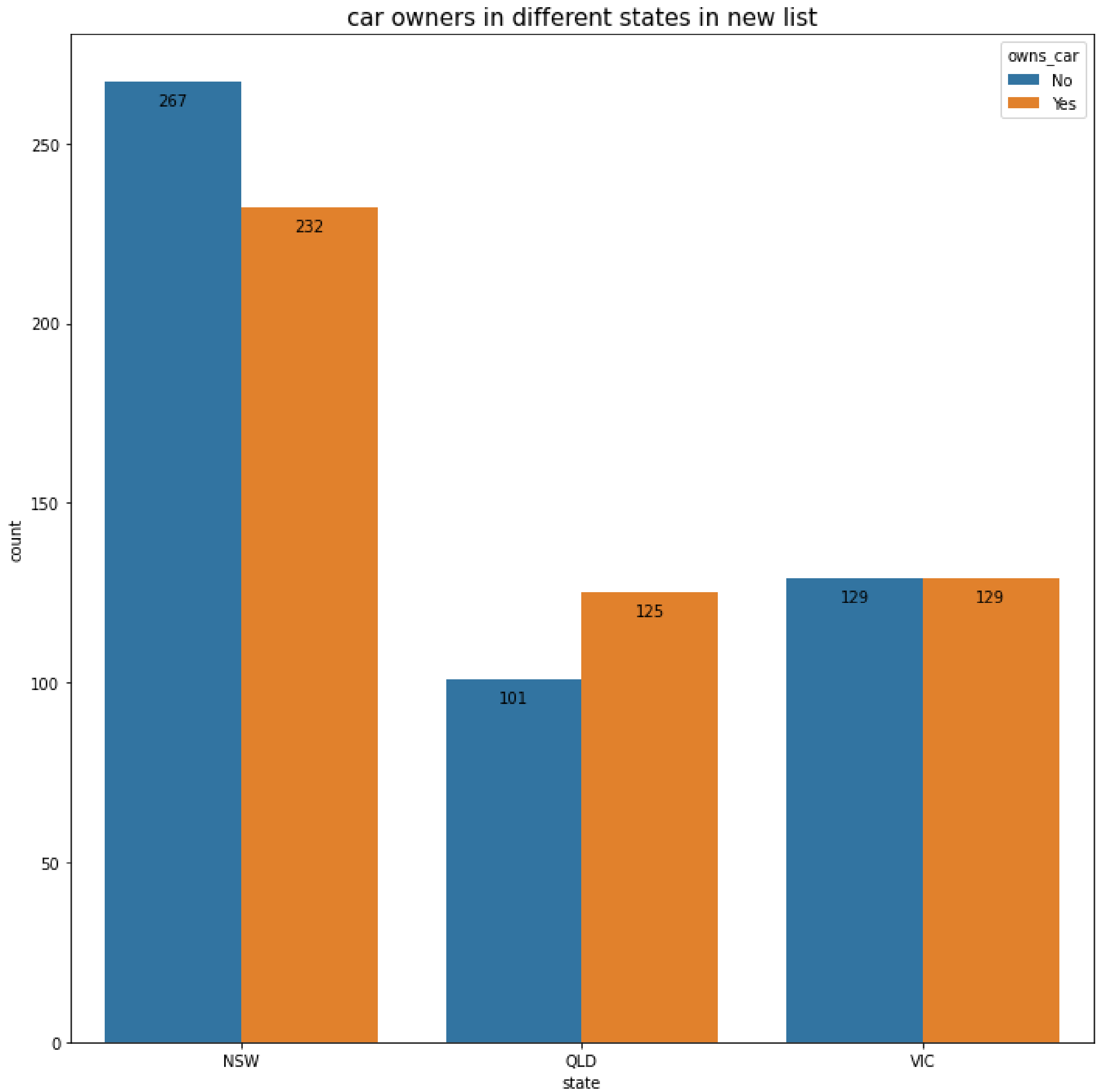
```
range(6)
```

Out[123]:

```
range(0, 6)
```

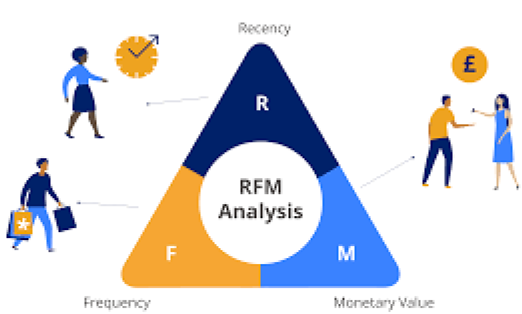
In [124…

```
plt.figure(figsize=(10,10))
c=sns.barplot(data=data,x='state',y='count',hue="owns_car")
plt.title("car owners in different states in new list",fontsize=15)
for i in c.patches:
  c.annotate(format(round(i.get_height()), '.0f'),
                   (i.get_x() + i.get_width() / 2., i.get_height()),
                   ha='center', va='center',color='black',
                   size=10,
                   xytext=(0, -12),textcoords='offset points'
                   )

plt.tight_layout()
```



NSW has largest amount o people that do not owns a car. Victoria has these numbers spread evenly. QLD has relatively a high number of customers that owns a car.

RFM Analysis



Recency- How recent was the customer's last purchase? Customers who recently made a purchase will still have the product on their mind and are more likely to purchase or use the product again. Businesses often measure recency in days. But, depending on the product, they may measure it in years, weeks or even hours.Frequency- How often did this customer make a purchase in a given period? Customers who purchased once are often are more likely to purchase again. Additionally, first time customers may be good targets for follow-up advertising to convert them into more frequent customers.Monetary- How much money did the customer spend in a given period? Customers who spend a lot of money are more likely to spend money in the future and have a high value to a business.

Creating a new column to know last purchase

In [125…

```
most_recent_purchase =Transactions['transaction_date'].max()
Transactions['last_purchase_days_ago'] = most_recent_purchase -Transactions['transaction
Transactions['last_purchase_days_ago'] /= np.timedelta64(1, 'D')

Transactions.head(25)
```

Out[125]:

	transaction_id	product_id	customer_id	transaction_date	online_order	order_status	brand	product_line
1	1	2	2950	2017-02-25	False	Approved	Solex	Standard
2	2	3	3120	2017-05-21	True	Approved	Trek Bicycles	Standard
3	3	37	402	2017-10-16	False	Approved	OHM Cycles	Standard
4	4	88	3135	2017-08-31	False	Approved	Norco Bicycles	Standard
5	5	78	787	2017-10-01	True	Approved	Giant Bicycles	Standard
6	6	25	2339	2017-03-08	True	Approved	Giant Bicycles	Road
7	7	22	1542	2017-04-21	True	Approved	WeareA2B	Standard
8	8	15	2459	2017-07-15	False	Approved	WeareA2B	Standard
9	9	67	1305	2017-08-10	False	Approved	Solex	Standard
10	10	12	3262	2017-08-30	True	Approved	WeareA2B	Standard
11	11	5	1986	2017-01-17	False	Approved	Trek Bicycles	Mountain
12	12	61	2783	2017-01-05	True	Approved	OHM Cycles	Standard
13	13	35	1243	2017-02-26	True	Approved	Trek Bicycles	Standard
14	14	16	2717	2017-09-10	False	Approved	Norco Bicycles	Standard
15	15	12	247	2017-06-11	False	Approved	Giant Bicycles	Standard
16	16	3	2961	2017-10-10	False	Approved	Trek Bicycles	Standard
17	17	79	2426	2017-04-03	False	Approved	Norco Bicycles	Standard
18	18	33	1842	2017-06-02	False	Approved	Giant Bicycles	Standard
19	19	54	2268	2017-04-06	True	Approved	WeareA2B	Standard
20	20	25	3002	2017-01-28	True	Approved	Giant	Road

							Bicycles	
21	21	27	1582	2017-10-09	False	Approved	Trek Bicycles	Standard
22	22	37	595	2017-06-29	True	Approved	OHM Cycles	Standard
23	23	37	2001	2017-04-08	True	Approved	OHM Cycles	Standard
24	24	82	515	2017-10-18	False	Approved	Giant Bicycles	Road
25	25	89	2822	2017-06-11	False	Approved	WeareA2B	Touring

In [126…

```
rfmTable =Transactions.groupby('customer_id').agg({
    'last_purchase_days_ago': lambda x: x.min(),
    'customer_id': lambda x: len(x),
    'profit': lambda x: x.sum()
})

rfmTable.rename(columns={
    'last_purchase_days_ago': 'recency',
    'customer_id': 'frequency',
    'profit': 'monetary_value'
    }, inplace=True
)
```

In [127…

```
rfmTable.shape
```

Out[127]:

```
(3494, 3)
```

In [128…

```
rfmTable.head()
```

Out[128]:

	recency	frequency	monetary_value
customer_id			
1	7.0	11	3016
2	128.0	3	2226
3	102.0	8	3363
4	195.0	2	221
5	16.0	6	2394

In [129…

```
quartiles = rfmTable.quantile(q=[0.25,0.50,0.75])
quartiles
```

Out[129]:

	recency	frequency	monetary_value
0.25	17.0	4.0	1874.00
0.50	44.0	6.0	2891.50
0.75	85.0	7.0	4240.75

Giving the rfm score

In [130…
```
rfmTable['R_rank'] = rfmTable['recency'].rank(ascending=False)
rfmTable['F_rank'] = rfmTable['frequency'].rank(ascending=True)
rfmTable['M_rank'] = rfmTable['monetary_value'].rank(ascending=True)

rfmTable["R_rank_norm"]=(rfmTable['R_rank']/(rfmTable['R_rank'].max()))*100
rfmTable["F_rank_norm"]=(rfmTable['F_rank']/(rfmTable['F_rank'].max()))*100
rfmTable["M_rank_norm"]=(rfmTable['M_rank']/(rfmTable['M_rank'].max()))*100
rfmTable.drop(["R_rank","F_rank","M_rank"],axis=1,inplace=True)
rfmTable
```

Out[130]:

customer_id	recency	frequency	monetary_value	R_rank_norm	F_rank_norm	M_rank_norm
1	7.0	11	3016	89.988476	97.838534	52.575844
2	128.0	3	2226	12.921348	12.367592	33.714940
3	102.0	8	3363	19.014693	83.395362	59.645106
4	195.0	2	221	3.817344	4.308617	1.516886
5	16.0	6	2394	77.297609	57.171486	37.893532
...	...	...	...	...	...	...
3497	52.0	3	1649	44.281187	12.367592	20.105896
3498	127.0	6	3147	13.065399	57.171486	55.051517
3499	51.0	7	4957	44.929415	72.129974	84.559244
3500	144.0	6	1787	9.478536	57.171486	22.953635
5034	84.0	3	269	25.669836	12.367592	1.888952

3494 rows × 6 columns

RFM score is calculated based upon recency, frequency, monetary value normalize ranks. Based upon this score we divide our customers. Here we rate them on a scale of 5. Formula used for calculating rfm score is : 0.15*Recency score + 0.28*Frequency score + 0.57 *Monetary score

In [131…
```
rfmTable['RFM_Score'] = 0.15*rfmTable['R_rank_norm']+0.28 * \
    rfmTable['F_rank_norm']+0.57*rfmTable['M_rank_norm']
rfmTable['RFM_Score'] *= 0.05
rfmTable= rfmTable.round(2)
rfmTable.shape
rfmTable.reset_index("customer_id")
rfmTable.head(5)
```

Out[131]:

customer_id	recency	frequency	monetary_value	R_rank_norm	F_rank_norm	M_rank_norm	RFM_Score
1	7.0	11	3016	89.99	97.84	52.58	3.54
2	128.0	3	2226	12.92	12.37	33.71	1.23
3	102.0	8	3363	19.01	83.40	59.65	3.01
4	195.0	2	221	3.82	4.31	1.52	0.13
5	16.0	6	2394	77.30	57.17	37.89	2.46

Dividing the customers into categories :- Top Customer High value customer Medium Value Customer Low value cutomer Lost customer

In [132…
```
rfmTable["Customer_segment"] = np.where(rfmTable['RFM_Score'] >
                                        4.5, "Top Customers",
```

```
                                        (np.where(
                                          rfmTable['RFM_Score'] > 4,
                                          "High value Customer",
                                          (np.where(
    rfmTable['RFM_Score'] > 3,
                             "Medium Value Customer",
                             np.where(rfmTable['RFM_Score'] > 1.6,
                            'Low Value Customers', 'Lost Customers'))))))
rfmTable[['RFM_Score', 'Customer_segment']].head(20)
```

Out[132]:

customer_id	RFM_Score	Customer_segment
1	3.54	Medium Value Customer
2	1.23	Lost Customers
3	3.01	Medium Value Customer
4	0.13	Lost Customers
5	2.46	Low Value Customers
6	2.86	Low Value Customers
7	0.23	Lost Customers
8	4.63	Top Customers
9	2.07	Low Value Customers
10	3.62	Medium Value Customer
11	3.03	Medium Value Customer
12	3.07	Medium Value Customer
13	3.68	Medium Value Customer
14	1.67	Low Value Customers
15	1.85	Low Value Customers
16	2.95	Low Value Customers
17	2.11	Low Value Customers
18	2.90	Low Value Customers
19	1.78	Low Value Customers
20	2.65	Low Value Customers

In [133…

```
plt.figure(figsize=(10,10))
sns.scatterplot(data=rfmTable,x=rfmTable['recency'],y=rfmTable['monetary_value'],hue="Cu
plt.title('Recency v/s Avg Profit')
plt.xlabel("Recency")
plt.ylabel('monetary value')
plt.show()
```

It shows that less recency customers have'nt generated much of the monetary value. Lost customers haven't generated that much profits. Top customers have some High monetary value that even touches 12000 value but low recency. Medium value customers Coagulated at 3000-4000.

In [134…

```
data=Transactions.groupby("product_class").sum().reset_index("product_class")


data['profit_percentage']=(100*data['profit_percentage'])/data['profit_percentage'].sum(
data
```

Out[134]:

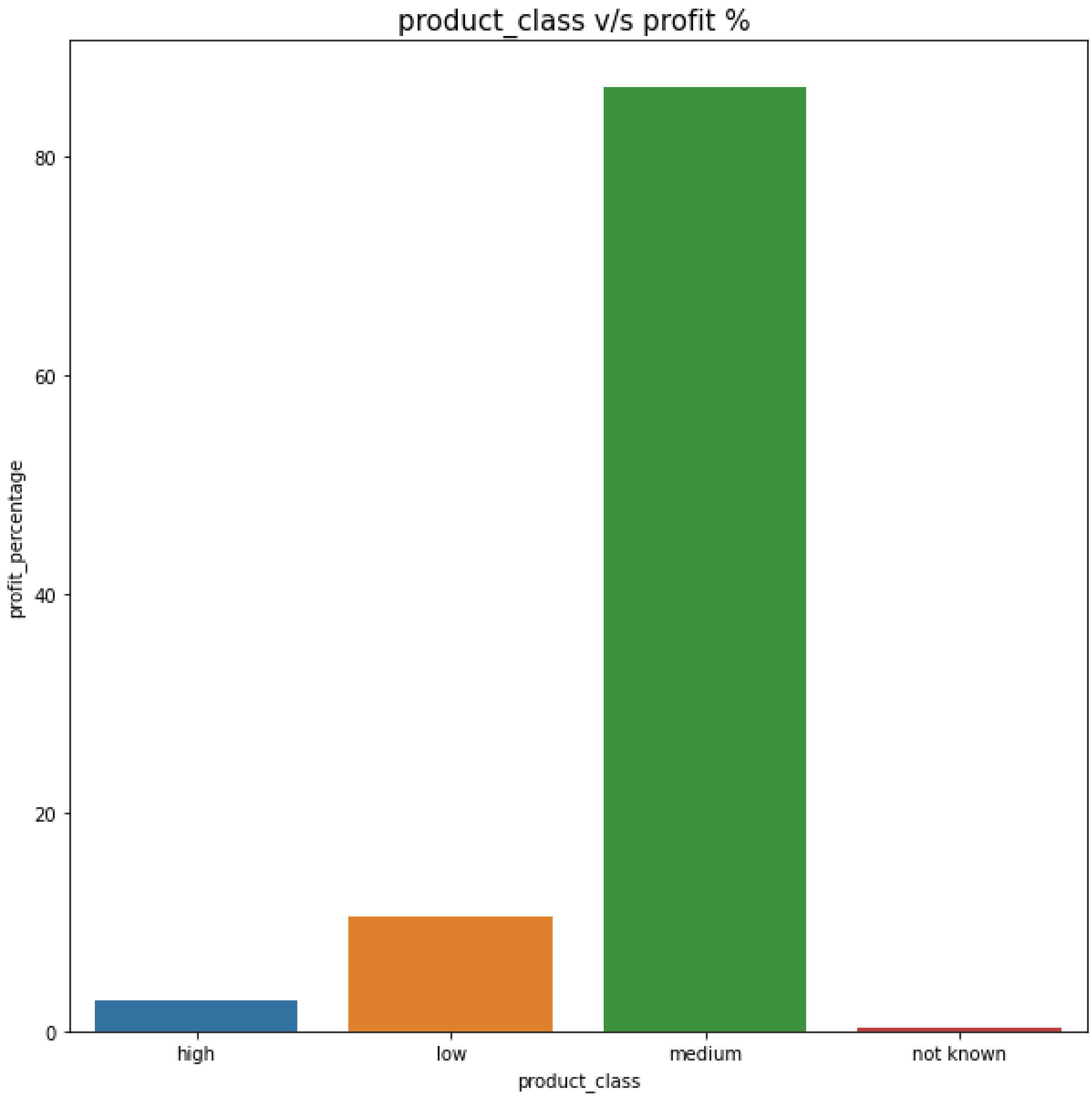
	product_class	transaction_id	product_id	customer_id	list_price	standard_cost	profit	profit_percentage	la
0	high	30530037	120232	5236135	3287856	2171944	1115912	2.848345	
1	low	29355828	140918	5189023	2748093	1542882	1205211	10.551437	
2	medium	138024914	646143	24018100	15895665	7285718	8609947	86.303028	
3	not known	2099221	0	321663	214809	109532	105277	0.297190	

In [135…

```
plt.figure(figsize=(10,10))
c=sns.barplot(data=data,x='product_class',y="profit_percentage")
plt.title("product_class v/s profit % ",fontsize=15)
```

Out[135]:

```
Text(0.5, 1.0, 'product_class v/s profit % ')
```



We can see that about 85 % of profit is been generated by the medium product_class. Somewhat 0.30% data is unknown. 10% profit is also obtained from the low product_class. Our target audience is basically who purchases the medium class product.

Merging the rfm Table and Old Customers Table

In [136…
```
data=pd.merge(cd,rfmTable,on="customer_id",how="inner")
```

In [137…
```
(data['Customer_segment']=="Top Customers").value_counts()
```

Out[137]:
```
False    3273
True      141
Name: Customer_segment, dtype: int64
```

In [138…
```
 df=data.groupby("Customer_segment")["job_industry_category"].value_counts()
df1=pd.DataFrame(df)
df2=df1.iloc[40:]
df2.rename(columns={'job_industry_category':"count"},inplace=True)

df2.reset_index("job_industry_category",inplace=True)
df2
```

```
C:\Users\91913\AppData\Local\Temp\ipykernel_16880\1344750093.py:4: SettingWithCopyWarnin
g:
A value is trying to be set on a copy of a slice from a DataFrame
```

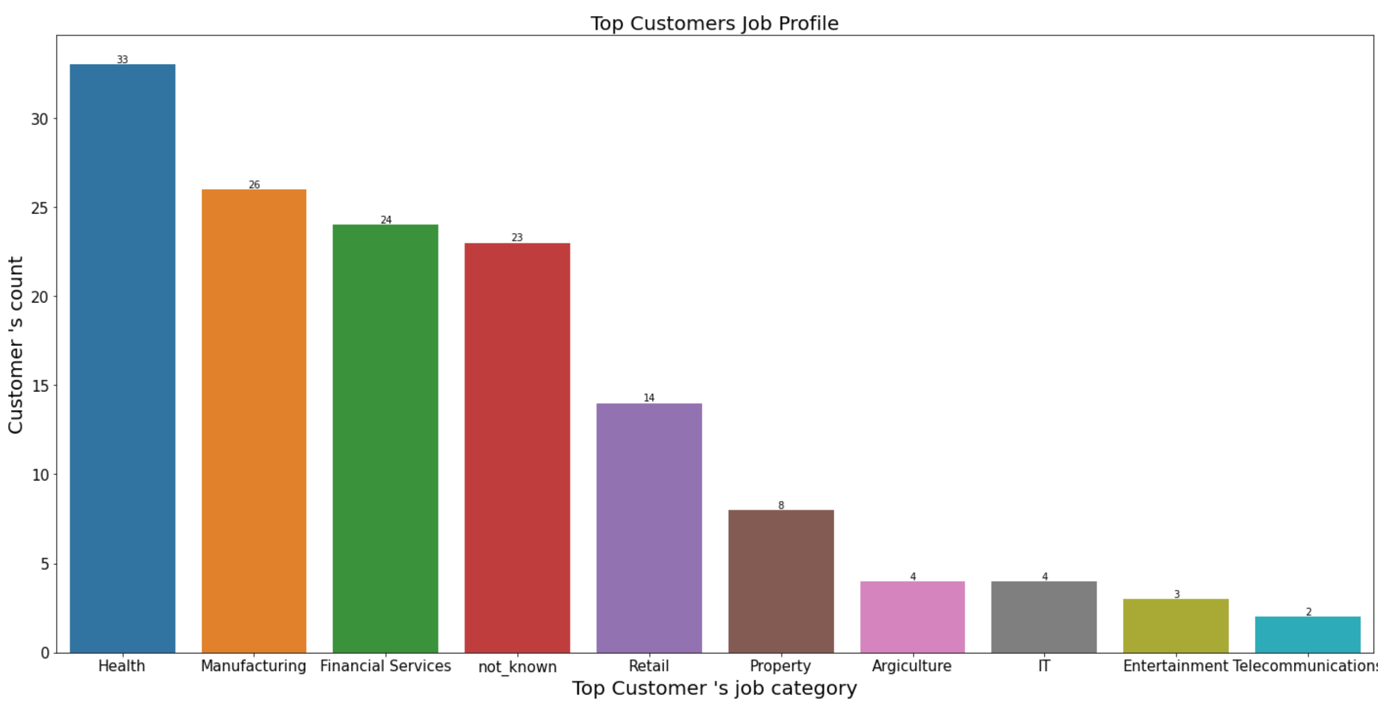
```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df2.rename(columns={'job_industry_category':"count"},inplace=True)
```

Out[138]:

Customer_segment	job_industry_category	count
Top Customers	Health	33
Top Customers	Manufacturing	26
Top Customers	Financial Services	24
Top Customers	not_known	23
Top Customers	Retail	14
Top Customers	Property	8
Top Customers	Argiculture	4
Top Customers	IT	4
Top Customers	Entertainment	3
Top Customers	Telecommunications	2

In [139…

```
plt.figure(figsize=(20,10))
d=sns.barplot(data=df2,x="job_industry_category",y="count")
plt.xlabel("Top Customer 's job category",fontsize=20)

for i in d.containers:
    d.bar_label(i,)
plt.ylabel("Customer 's count",fontsize=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.title("Top Customers Job Profile",fontsize=20)
plt.tight_layout()
```



Looking from the bar graph we get to know that the most of the top customer belong from the Health and Manufacturing sector. 23 customers also contribute but they are from unknown profiles.

In [140…

```
df3=df1.iloc[10:20]
df3.rename(columns={'job_industry_category':"count"},inplace=True)
```

```
df3.reset_index("job_industry_category",inplace=True)
df3
```

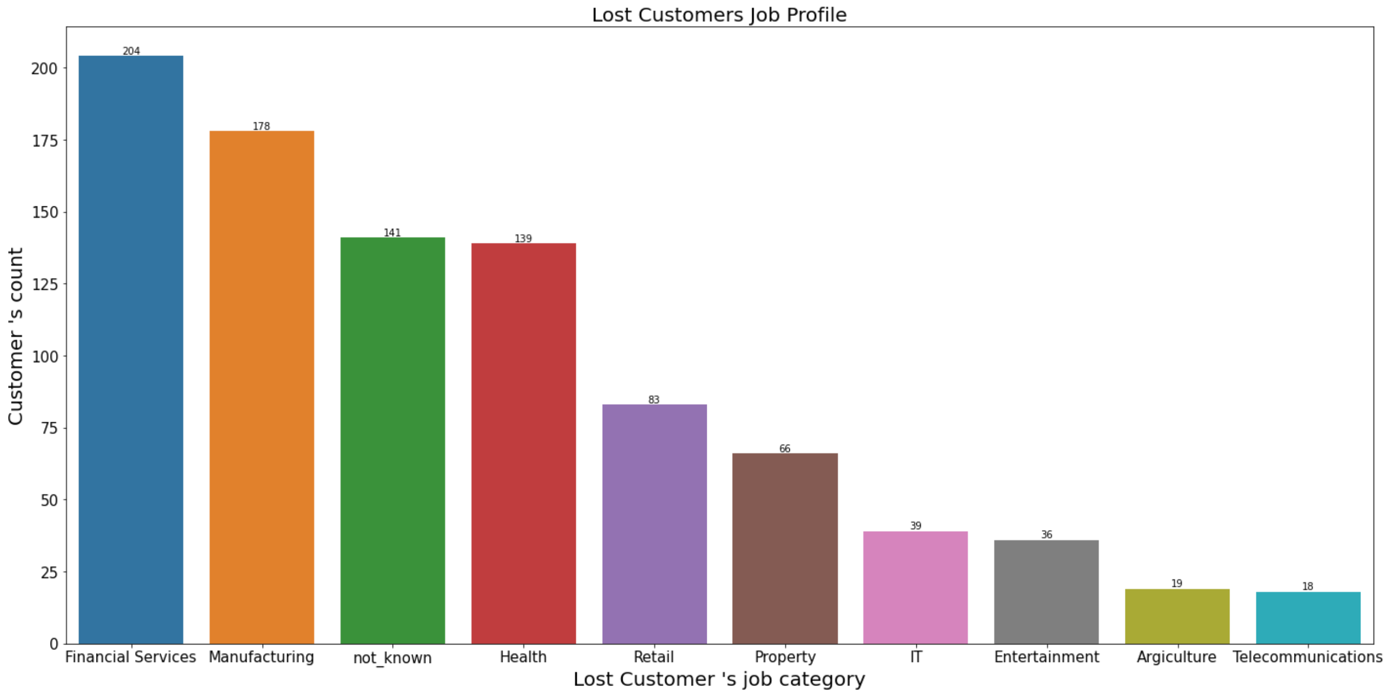
```
C:\Users\91913\AppData\Local\Temp\ipykernel_16880\1467183473.py:2: SettingWithCopyWarnin
g:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  df3.rename(columns={'job_industry_category':"count"},inplace=True)
```

Out[140]:

	job_industry_category	count
Customer_segment		
Lost Customers	Financial Services	204
Lost Customers	Manufacturing	178
Lost Customers	not_known	141
Lost Customers	Health	139
Lost Customers	Retail	83
Lost Customers	Property	66
Lost Customers	IT	39
Lost Customers	Entertainment	36
Lost Customers	Argiculture	19
Lost Customers	Telecommunications	18

In [141…

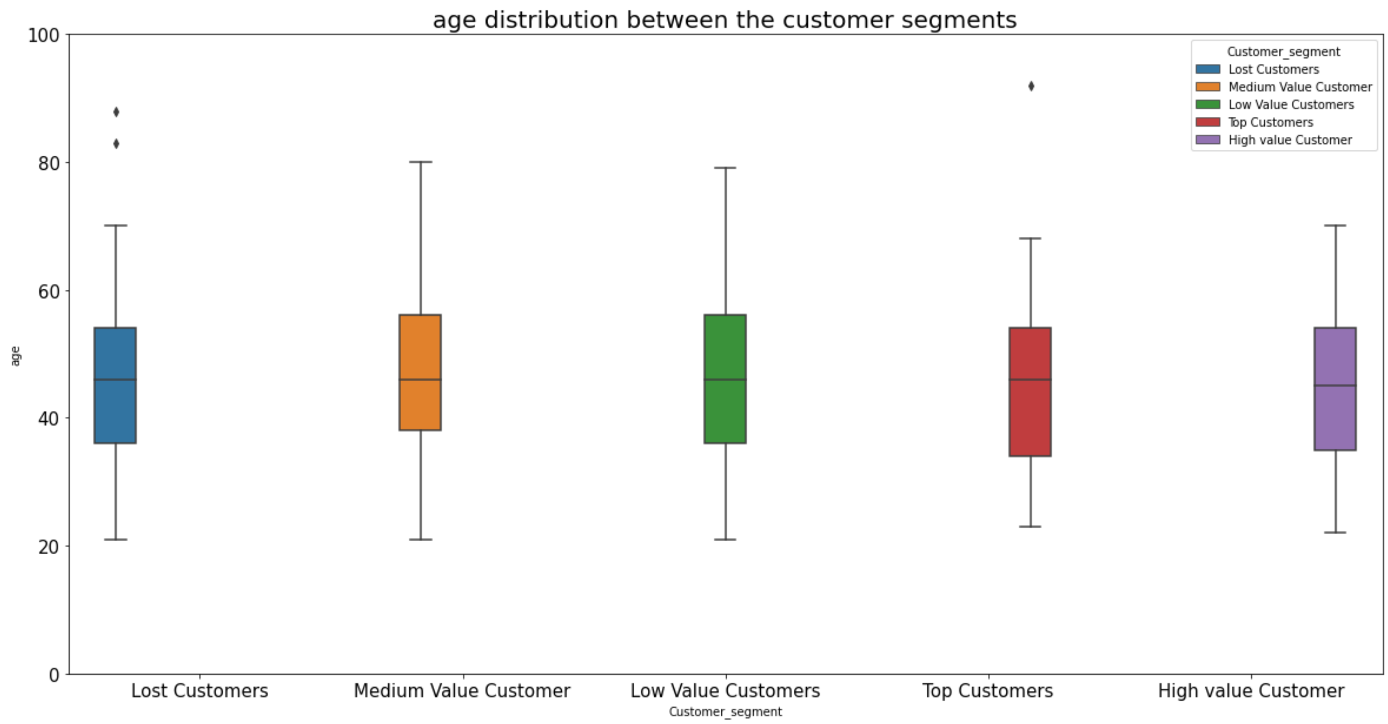
```
plt.figure(figsize=(20,10))
d=sns.barplot(data=df3,x="job_industry_category",y="count")
plt.xlabel("Lost Customer 's job category",fontsize=20)

for i in d.containers:
    d.bar_label(i,)
plt.ylabel("Customer 's count",fontsize=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.title("Lost Customers Job Profile",fontsize=20)
plt.tight_layout()
```



The customers that we lost mostly belong from the Financial services background and manufacturing.

In [142…

```
plt.figure(figsize=(20,10))
sns.boxplot(data=data,x="Customer_segment",y="age",hue="Customer_segment")
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.ylim(0,100)
plt.title("age distribution between the customer segments",fontsize=20)
plt.show()
```



Most of the age of the customers lie between 40-50. Middle value customers have a high upper limit of approx 80 years.

Task -3

Tableau interactive Dashboard

1)Saving all the updated and corrected files to use as tableau data.

In [143...

```
pip install openpyxl
```

```
Requirement already satisfied: openpyxl in c:\users\91913\anaconda3\lib\site-packages
(3.0.9)
Requirement already satisfied: et-xmlfile in c:\users\91913\anaconda3\lib\site-packages
(from openpyxl) (1.1.0)
Note: you may need to restart the kernel to use updated packages.
```

In [144...

```
Transactions.to_excel("Transactions.xlsx")
Ncl.to_excel("Ncl.xlsx")
```

In [145...

```
cd.to_excel("cd.xlsx")
```

In [146...

```
ca.to_excel("ca.xlsx")
```

In [147...

```
rfmTable.to_excel('rfm.xlsx')
%%html
<div class='tableauPlaceholder' id='viz1684846726063' style='position: relative'><noscri
```