# KPMG VIRTUAL INTERNSHIP PROJECT

## TASK: 1 - Data Quality Assessment

Assessment of data quality and completeness in preparation for analysis.

The client provided KPMG with 3 datasets:

1.Customer Demographic

2.Customer Addresses

3.Transactions data in the past 3 months

In [1]:

```
# Importing the required libraries
import pandas as pd
import numpy as np
```

# Reading files

In [9]:

```
data = pd.ExcelFile("KPMG.xlsx")
```

# Reading each file seperately

In [12]:

```
Transactions = pd.read_excel(data, 'Transactions', header=1)
NewCustomerList = pd.read_excel(data, 'NewCustomerList', header=1)
CustomerDemographic = pd.read_excel(data, 'CustomerDemographic', header=1)
CustomerAddress = pd.read_excel(data, 'CustomerAddress', header=1)
```

```
C:\Users\Jay\AppData\Local\Temp/ipykernel_4212/2496028931.py:2: FutureWarning: Infer
ring datetime64[ns] from data containing strings is deprecated and will be removed i
n a future version. To retain the old behavior explicitly pass Series(data, dtype={v
alue.dtype})
  NewCustomerList = pd.read_excel(data, 'NewCustomerList', header=1)
C:\Users\Jay\AppData\Local\Temp/ipykernel_4212/2496028931.py:3: FutureWarning: Infer
ring datetime64[ns] from data containing strings is deprecated and will be removed i
n a future version. To retain the old behavior explicitly pass Series(data, dtype={v
alue.dtype})
  CustomerDemographic = pd.read_excel(data, 'CustomerDemographic', header=1)
```

# Exploring Transactions Data Set

In [16]:

```
Transactions.head(5)
```

Out[16]:

| | transaction_id | product_id | customer_id | transaction_date | online_order | order_status | brand | pro |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 2 | 2950 | 2017-02-25 | 0.0 | Approved | Solex | |

| | transaction_id | product_id | customer_id | transaction_date | online_order | order_status | brand | pro |
|---|---|---|---|---|---|---|---|---|
| **1** | 2 | 3 | 3120 | 2017-05-21 | 1.0 | Approved | Trek Bicycles | |
| **2** | 3 | 37 | 402 | 2017-10-16 | 0.0 | Approved | OHM Cycles | |
| **3** | 4 | 88 | 3135 | 2017-08-31 | 0.0 | Approved | Norco Bicycles | |
| **4** | 5 | 78 | 787 | 2017-10-01 | 1.0 | Approved | Giant Bicycles | |

In [17]:

```
Transactions.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
```

In [18]:

```
#Using only the required columns
Transactions = Transactions.iloc[:, 0:13]
Transactions.head()
```

Out[18]:

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

In [19]:

```
Transactions.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
```

In [20]:

```
#Checking the shape of the data
Transactions.shape
```

Out[20]:

```
(20000, 13)
```

In [21]:

```
#Checking for null values
Transactions.isnull().sum()
```

Out[21]:

```
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
```

# There are missing values in 7 columns. They can be dropped or treated according to the nature of analysis

In [22]:

```
#Checking for duplicate values
Transactions.duplicated().sum()
```

Out[22]:

```
0
```

# There are no duplicate values, so the data is unique.

In [23]:

```
#check for uniqueness of each column
Transactions.nunique()
```

Out[23]:

```
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
```

# Exploring the columns

In [27]:
```
Transactions.columns
```

Out[27]:
```
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')
```

In [28]:
```
Transactions['order_status'].value_counts()
```

Out[28]:
```
Approved     19821
Cancelled      179
Name: order_status, dtype: int64
```

In [29]:
```
Transactions['brand'].value_counts()
```

Out[29]:
```
Solex             4253
Giant Bicycles    3312
WeareA2B          3295
OHM Cycles        3043
Trek Bicycles     2990
Norco Bicycles    2910
Name: brand, dtype: int64
```

In [30]:
```
Transactions['product_line'].value_counts()
```

Out[30]:
```
Standard    14176
Road         3970
Touring      1234
Mountain      423
Name: product_line, dtype: int64
```

In [31]:
```
Transactions['product_class'].value_counts()
```

Out[31]:
```
medium    13826
high       3013
low        2964
Name: product_class, dtype: int64
```

In [32]:
```
Transactions['product_size'].value_counts()
```

Out[32]:
```
medium    12990
large      3976
```

```
small      2837
Name: product_size, dtype: int64
```

In [33]:

```
Transactions['product_first_sold_date']
```

Out[33]:

```
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
```

In [34]:

```
#convert date column from integer to datetime
Transactions['product_first_sold_date'] = pd.to_datetime(Transactions['product_first
Transactions['product_first_sold_date'].head()
```

Out[34]:

```
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]
```

In [35]:

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

Out[35]:

```
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]
```

## The values in the product_first_sold_date columns are not correct as it shows everything happening the same day at different times.

# Exploring New Customer List Data Set

In [36]:
```
NewCustomerList.head(5)
```

Out[36]:

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

5 rows × 23 columns

In [37]:
```
NewCustomerList.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
```

In [38]:
```
#Dropping the unnamed columns
NewCustomerList.drop(['Unnamed: 16', 'Unnamed: 17', 'Unnamed: 18',
       'Unnamed: 19', 'Unnamed: 20'], axis=1, inplace=True)
```

In [39]:
```
#Checking the shape of the dataset
NewCustomerList.shape
```
Out[39]:
```
(1000, 18)
```
In [40]:
```
#Checking for null values
NewCustomerList.isnull().sum()
```
Out[40]:
```
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
```
# There are missing values in 4 columns. They can be dropped or treated according to the nature of analysis

In [41]:
```
#Checking for duplicate values
NewCustomerList.duplicated().sum()
```
Out[41]:
```
0
```
# There are no duplicate values.

In [43]:
```
#Checking for uniquess of each column
NewCustomerList.nunique()
```
Out[43]:
```
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
```

# Exploring the columns

In [44]:

```
NewCustomerList.columns
```

Out[44]:

```
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')
```

In [45]:

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

Out[45]:

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

# There are 17 columns with unknown/unspecified gender.

In [47]:

```
NewCustomerList['DOB'].value_counts()
```

Out[47]:

```
1998-02-05    2
1978-01-15    2
1977-11-08    2
1951-11-28    2
1979-07-28    2
             ..
1945-08-08    1
1943-08-27    1
1999-10-24    1
1976-01-24    1
1955-10-02    1
Name: DOB, Length: 958, dtype: int64
```

In [48]:

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

Out[48]:

```
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
```

In [49]:

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

Out[49]:

```
Mass Customer        508
High Net Worth       251
Affluent Customer    241
Name: wealth_segment, dtype: int64
```

In [50]:
```
NewCustomerList['state'].value_counts()
```
Out[50]:
```
NSW    506
VIC    266
QLD    228
Name: state, dtype: int64
```

In [51]:
```
NewCustomerList['owns_car'].value_counts()
```
Out[51]:
```
No     507
Yes    493
Name: owns_car, dtype: int64
```

In [52]:
```
NewCustomerList['deceased_indicator'].value_counts()
```
Out[52]:
```
N    1000
Name: deceased_indicator, dtype: int64
```

# Exploring Customer Demographic Data Set

In [53]:
```
CustomerDemographic.head()
```
Out[53]:

| | customer_id | first_name | last_name | gender | past_3_years_bike_related_purchases | DOB | job |
|---|---|---|---|---|---|---|---|
| **0** | 1 | Laraine | Medendorp | F | 93 | 1953-10-12 | Exe Sec |
| **1** | 2 | Eli | Bockman | Male | 81 | 1980-12-16 | Administ C |
| **2** | 3 | Arlin | Dearle | Male | 61 | 1954-01-20 | Recr Ma |
| **3** | 4 | Talbot | NaN | Male | 33 | 1961-10-03 | |
| **4** | 5 | Sheila-kathryn | Calton | Female | 56 | 1977-05-13 | Senior |

In [54]:
```
CustomerDemographic.info()
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 0 to 3999
Data columns (total 13 columns):
 #   Column                               Non-Null Count  Dtype
---  ------                               --------------  -----
 0   customer_id                          4000 non-null   int64
 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   int64
 5   DOB                                  3913 non-null   datetime64[ns]
 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   float64
dtypes: datetime64[ns](1), float64(1), int64(2), object(9)
memory usage: 406.4+ KB
```

In [55]:

```
#Checking for null values
CustomerDemographic.isnull().sum()
```

Out[55]:

```
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
```

## There are missing values in 5 columns. They can be dropped or treated according to the nature of analysis

In [56]:

```
#Checking for duplicate data
CustomerDemographic.duplicated().sum()
```

Out[56]:

```
0
```

## There are no duplicate values.

In [57]:

```
#Checking for uniqueness of each column
CustomerDemographic.nunique()
```

Out[57]:

```
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
```

## Exploring the columns

In [70]:

```
CustomerDemographic.columns
```

Out[70]:

```
Index(['customer_id', 'first_name', 'last_name', 'gender',
       'past_3_years_bike_related_purchases', 'DOB', 'job_title',
       'job_industry_category', 'wealth_segment', 'deceased_indicator',
```

```
       'default', 'owns_car', 'tenure'],
      dtype='object')
```

In [71]:

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

Out[71]:

```
Female        2039
Male          1873
Unspecific      88
Name: gender, dtype: int64
```

# Certain categories are not correctly titled. The names in these categories are re-named.

In [72]:

```
#Re-naming the categories
CustomerDemographic['gender'] = CustomerDemographic['gender'].replace('F','Female').
```

In [73]:

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

Out[73]:

```
Female        2039
Male          1873
Unspecific      88
Name: gender, dtype: int64
```

In [74]:

```
CustomerDemographic['past_3_years_bike_related_purchases'].value_counts()
```

Out[74]:

```
16    56
19    56
67    54
20    54
2     50
      ..
8     28
95    27
85    27
86    27
92    24
Name: past_3_years_bike_related_purchases, Length: 100, dtype: int64
```

In [75]:

```
CustomerDemographic['DOB'].value_counts()
```

Out[75]:

```
1978-01-30    7
1964-07-08    4
1962-12-17    4
1978-08-19    4
1977-05-13    4
             ..
1989-06-16    1
1998-09-30    1
1985-03-11    1
1989-10-23    1
1991-11-05    1
Name: DOB, Length: 3448, dtype: int64
```

In [76]:

```
CustomerDemographic['job_title'].value_counts()
```

Out[76]:

```
Business Systems Development Analyst    45
Tax Accountant                          44
Social Worker                           44
Internal Auditor                        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: 195, dtype: int64
```

In [77]:
```
CustomerDemographic['job_industry_category'].value_counts()
```

Out[77]:
```
Manufacturing         799
Financial Services    774
Health                602
Retail                358
Property              267
IT                    223
Entertainment         136
Argiculture           113
Telecommunications     72
Name: job_industry_category, dtype: int64
```

In [78]:
```
CustomerDemographic['wealth_segment'].value_counts()
```

Out[78]:
```
Mass Customer        2000
High Net Worth       1021
Affluent Customer     979
Name: wealth_segment, dtype: int64
```

In [79]:
```
CustomerDemographic['deceased_indicator'].value_counts()
```

Out[79]:
```
N    3998
Y       2
Name: deceased_indicator, dtype: int64
```

In [80]:
```
CustomerDemographic['default'].value_counts()
```

Out[80]:
```
100                                         113
1                                           112
-1                                          111
-100                                         99
Ù¡Ù¢Ù£                                       53
                                           ...
testâ testâ«                                 31
/dev/null; touch /tmp/blns.fail ; echo       30
âªâªtestâª                                   29
ì¸ëºí ë¥´                                      27
,ãã»:*:ã»ãâ( â» Ï â» )ãã»:*:ã»ãâ            25
Name: default, Length: 90, dtype: int64
```

In [81]:
```
CustomerDemographic = CustomerDemographic.drop('default', axis=1)
```

## The values are inconsistent, hence dropping the column.

In [82]:
```
CustomerDemographic.head(5)
```

Out[82]:

| customer_id | first_name | last_name | gender | past_3_years_bike_related_purchases | DOB | job |
|---|---|---|---|---|---|---|

| | customer_id | first_name | last_name | gender | past_3_years_bike_related_purchases | DOB | job |
|---|---|---|---|---|---|---|---|
| **0** | 1 | Laraine | Medendorp | Female | 93 | 1953-10-12 | Exe Sec |
| **1** | 2 | Eli | Bockman | Male | 81 | 1980-12-16 | Administ C |
| **2** | 3 | Arlin | Dearle | Male | 61 | 1954-01-20 | Recr Ma |
| **3** | 4 | Talbot | NaN | Male | 33 | 1961-10-03 | |
| **4** | 5 | Sheila-kathryn | Calton | Female | 56 | 1977-05-13 | Senior |

In [83]:

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

Out[83]:

```
Yes    2024
No     1976
Name: owns_car, dtype: int64
```

In [84]:

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

Out[84]:

```
7.0     235
5.0     228
11.0    221
10.0    218
16.0    215
8.0     211
18.0    208
12.0    202
9.0     200
14.0    200
6.0     192
13.0    191
4.0     191
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
```

## Exploring Customer Address Data Set

In [85]:

```
CustomerAddress.head(5)
```

Out[85]:

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

| | customer_id | address | postcode | state | country | property_valuation |
|---|---|---|---|---|---|---|
| **3** | 5 | 17979 Del Mar Point | 2448 | New South Wales | Australia | 4 |
| **4** | 6 | 9 Oakridge Court | 3216 | VIC | Australia | 9 |

In [86]:

```
CustomerAddress.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
```

In [87]:

```
#Checking for null values.
CustomerAddress.isnull().sum()
```

Out[87]:

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

# There are no null values.

In [88]:

```
#Checking for duplicate values
CustomerAddress.duplicated().sum()
```

Out[88]:

```
0
```

# There are no duplicate values.

In [89]:

```
#Checking for uniqueness of each column
CustomerAddress.nunique()
```

Out[89]:

```
customer_id           3999
address               3996
postcode               873
state                    5
country                  1
property_valuation      12
dtype: int64
```

# Exploring the columns

In [90]:

```
CustomerAddress['postcode'].value_counts()
```

Out[90]:
```
2170    31
2155    30
2145    30
2153    29
3977    26
        ..
3808     1
3114     1
4721     1
4799     1
3089     1
Name: postcode, Length: 873, dtype: int64
```

In [91]:
```
CustomerAddress['state'].value_counts()
```

Out[91]:
```
NSW                2054
VIC                 939
QLD                 838
New South Wales      86
Victoria             82
Name: state, dtype: int64
```

In [92]:
```
CustomerAddress['country'].value_counts()
```

Out[92]:
```
Australia    3999
Name: country, dtype: int64
```

In [93]:
```
CustomerAddress['property_valuation'].value_counts()
```

Out[93]:
```
9     647
8     646
10    577
7     493
11    281
6     238
5     225
4     214
12    195
3     186
1     154
2     143
Name: property_valuation, dtype: int64
```

# All the columns appear to have consistent and correct information.