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
```

Reading the data

In [2]:
```
data = pd.ExcelFile("KPMG_VI_New_raw_data_update_final.xlsx")
```

Reading each file separately

In [3]:
```
Transactions = pd.read_excel(data, 'Transactions')
CustomerDemographic = pd.read_excel(data, 'CustomerDemographic')
CustomerAddress = pd.read_excel(data, 'CustomerAddress')
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_10440\2522582402.py:2: FutureWarning: Inferr
ing datetime64[ns] from data containing strings is deprecated and will be removed in
a future version. To retain the old behavior explicitly pass Series(data, dtype=datet
ime64[ns])
  CustomerDemographic = pd.read_excel(data, 'CustomerDemographic')
```

Exploring Transactions Data Set

In [4]:
```
Transactions.head(5)
```

Out[4]:

	transaction_id	product_id	customer_id	transaction_date	online_order	order_status	brand	proc
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 [5]:
```
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 [6]:
```
#Using only the required columns
Transactions = Transactions.iloc[:, 0:13]
Transactions.head()
```

Out[6]:

	transaction_id	product_id	customer_id	transaction_date	online_order	order_status	brand	proc
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 [7]:
```
#Checking the shape of the data
Transactions.shape
```

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

In [8]:
```
#Checking for null values
Transactions.isnull().sum()
```

Out[8]:
```
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 [9]:
```
#Checking for duplicate values
Transactions.duplicated().sum()
```

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

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

In [10]:
```
#check for uniqueness of each column
Transactions.nunique()
```

Out[10]:
```
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 [11]:
```
Transactions.columns
```

Out[11]:
```
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 [12]:
```
Transactions['order_status'].value_counts()
```

Out[12]:
```
Approved     19821
Cancelled      179
Name: order_status, dtype: int64
```

In [13]:
```
Transactions['brand'].value_counts()
```

Out[13]:
```
Solex              4253
Giant Bicycles     3312
WeareA2B           3295
OHM Cycles         3043
Trek Bicycles      2990
Norco Bicycles     2910
Name: brand, dtype: int64
```

In [14]:
```
Transactions['product_line'].value_counts()
```

Out[14]:
```
Standard    14176
Road         3970
Touring      1234
Mountain      423
Name: product_line, dtype: int64
```

In [15]:
```
Transactions['product_class'].value_counts()
```

Out[15]:
```
medium    13826
high       3013
low        2964
Name: product_class, dtype: int64
```

In [16]:
```
Transactions['product_size'].value_counts()
```

Out[16]:
```
medium    12990
large      3976
small      2837
Name: product_size, dtype: int64
```

In [17]:
```
Transactions['product_first_sold_date']
```

Out[17]:
```
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 [18]:
```
#convert date column from integer to datetime
Transactions['product_first_sold_date'] = pd.to_datetime(Transactions['product_first_s
Transactions['product_first_sold_date'].head(20)
```

Out[18]:
```
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 Customer Demographic Data Set

In [19]:
```
CustomerDemographic.head()
```

Out[19]:

	customer_id	first_name	last_name	gender	past_3_years_bike_related_purchases	DOB	job_
0	1	Laraine	Medendorp	F	93	1953-10-12	Execu Secre
1	2	Eli	Bockman	Male	81	1980-12-16	Administra Of
2	3	Arlin	Dearle	Male	61	1954-01-20	Recru Man
3	4	Talbot	NaN	Male	33	1961-10-03	
4	5	Sheila-kathryn	Calton	Female	56	1977-05-13	Senior Ec

In [20]:
```
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 [21]:
```
#Checking for null values
CustomerDemographic.isnull().sum()
```

Out[21]:
```
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 [22]:
```
#Checking for duplicate data
CustomerDemographic.duplicated().sum()
```

Out[22]:
```
0
```

There are no duplicate values.

In [23]:
```
#Checking for uniqueness of each column
CustomerDemographic.nunique()
```

Out[23]:
```
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 [24]:
```
CustomerDemographic.columns
```

Out[24]:
```
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 [25]:
```
CustomerDemographic['gender'].value_counts()
```

Out[25]:
```
Female    2037
Male      1872
U           88
F            1
Femal        1
M            1
Name: gender, dtype: int64
```

Certain categories are not correctly titled.

In [26]:
```
#Re-naming the categories
CustomerDemographic['gender'] = CustomerDemographic['gender'].replace('F','Female').re
```

In [27]:
```
CustomerDemographic['gender'].value_counts()
```

Out[27]:
```
Female    2039
Male      1873
U           88
Name: gender, dtype: int64
```

In [28]:
```
CustomerDemographic['past_3_years_bike_related_purchases'].value_counts()
```

Out[28]:
```
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 [29]:
```
CustomerDemographic['DOB'].value_counts()
```

Out[29]:
```
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 [30]:
```
CustomerDemographic['job_title'].value_counts()
```

Out[30]:
```
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 [31]:
```
CustomerDemographic['job_industry_category'].value_counts()
```

Out[31]:
```
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 [32]:
```
CustomerDemographic['wealth_segment'].value_counts()
```

Out[32]:
```
Mass Customer        2000
High Net Worth       1021
Affluent Customer     979
Name: wealth_segment, dtype: int64
```

In [33]:
```
CustomerDemographic['deceased_indicator'].value_counts()
```

Out[33]:
```
N    3998
Y       2
Name: deceased_indicator, dtype: int64
```

In [34]:
```
CustomerDemographic['default'].value_counts()
```

Out[34]:
```
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¸ëë°í ë¥´                                   27
,ãã»:*:ã»ãâ( â» Ï â» )ãã»:*:ã»ãâ             25
Name: default, Length: 90, dtype: int64
```

In [35]:
```
CustomerDemographic = CustomerDemographic.drop('default', axis=1)
```

The values are inconsistent, hence dropping the column.

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

Out[36]:

	customer_id	first_name	last_name	gender	past_3_years_bike_related_purchases	DOB	job_
0	1	Laraine	Medendorp	Female	93	1953-10-12	Execu Secre
1	2	Eli	Bockman	Male	81	1980-12-16	Administra Of
2	3	Arlin	Dearle	Male	61	1954-01-20	Recru Man
3	4	Talbot	NaN	Male	33	1961-10-03	
4	5	Sheila-kathryn	Calton	Female	56	1977-05-13	Senior Ec

In [37]:
```
CustomerDemographic['owns_car'].value_counts()
```

Out[37]:
```
Yes    2024
No     1976
Name: owns_car, dtype: int64
```

In [38]:
```
CustomerDemographic['tenure'].value_counts()
```

Out[38]:
```
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 [39]:
```
CustomerAddress.head(5)
```

Out[39]:

	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
3	5	17979 Del Mar Point	2448	New South Wales	Australia	4
4	6	9 Oakridge Court	3216	VIC	Australia	9

In [40]:
```
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 [41]:
```
#Checking for null values.
CustomerAddress.isnull().sum()
```

Out[41]:
```
customer_id           0
address               0
postcode              0
state                 0
country               0
property_valuation    0
dtype: int64
```

There are no null values.

In [42]:
```
#Checking for duplicate values
CustomerAddress.duplicated().sum()
```

Out[42]:
```
0
```

There are no duplicate values.

In [43]:
```
#Checking for uniqueness of each column
CustomerAddress.nunique()
```

Out[43]:
```
customer_id           3999
address               3996
postcode               873
state                    5
country                  1
property_valuation      12
dtype: int64
```

Exploring the columns

In [44]:
```
CustomerAddress['postcode'].value_counts()
```

Out[44]:
```
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 [45]:
```
CustomerAddress['state'].value_counts()
```

Out[45]:
```
NSW                2054
VIC                 939
QLD                 838
New South Wales      86
Victoria             82
Name: state, dtype: int64
```

In [46]:
```
CustomerAddress['country'].value_counts()
```

Out[46]:
```
Australia    3999
Name: country, dtype: int64
```

In [47]:
```
CustomerAddress['property_valuation'].value_counts()
```

Out[47]:
```
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.