Data_Quality_Assessment

March 5, 2021

[194]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
%matplotlib inline
from datetime import date, timedelta
```

0.1 Data Quality Assessment

by Murong (Sophie) Cui

Sprocket Central Pty Ltd , a medium size bikes & cycling accessories organisation, has approached Tony Smith (Partner) in KPMG's Lighthouse & Innovation Team. Sprocket Central Pty Ltd is keen to learn more about KPMG's expertise in its Analytics, Information & Modelling team.

Primarily, Sprocket Central Pty Ltd needs help with its customer and transactions data. The organisation has a large dataset relating to its customers, but their team is unsure how to effectively analyse it to help optimise its marketing strategy.

However, in order to support the analysis, after spoken to the Associate Director for some ideas, she advised that "the importance of optimising the quality of customer datasets cannot be underestimated. The better the quality of the dataset, the better chance you will be able to use it drive company growth."

The client provided KPMG with 3 datasets: - Customer Demographic - Customer Addresses - Transactions data in the past 3 months

Data Wrangling is followed by the guidline below:

[195]:
```
transactions_raw = pd.read_excel('data/KPMG_VI_New_raw_data_update_final.xlsx',␣
 ↪sheet_name = 'Transactions', header = 1)
newCustomer_raw = pd.read_excel('data/KPMG_VI_New_raw_data_update_final.xlsx',␣
 ↪sheet_name = 'NewCustomerList', header = 1)
customerDemographic_raw = pd.read_excel('data/KPMG_VI_New_raw_data_update_final.
 ↪xlsx', sheet_name = 'CustomerDemographic', header = 1)
customerAddress_raw = pd.read_excel('data/KPMG_VI_New_raw_data_update_final.
 ↪xlsx', sheet_name = 'CustomerAddress', header = 1)
```

[196]:
```
transactions = transactions_raw.copy()
newCustomer = newCustomer_raw.copy()
customerDemographic = customerDemographic_raw.copy()
customerAddress = customerAddress_raw.copy()
```

0.1.1 Transaction Data

- transaction_id: trasaction id
- product_id: product id
- customer_id: customer id
- transaction_date: transaction date
- online_order: 1 - ordered online, 0 - ordered onsite
- order_status: approved, canceled
- brand: product of brand
- product_line: Standard, Road, Touring, Mountain
- product_class: low, medium, high
- product_size: small, medium, large
- list_price: list price
- standard_cost: standard cost
- product_first_sold_date: the number of dates when the product first sold since 1899-12-31

0.1.2 Data Wrangling:

1. Uniqueness: Each row represents unique transaction: The dataset has 20000 rows and 13 columns containing 20000 transactions info during 3 months
2. Missing values: There 360 missing values on `online_order`, which indicated no info on whether the order was placed online or not. And there are 197 missing values on `brand`, `product_line`, `product_class`, `product_size`, `standard_cost` and `product_first_sold_date`. Those columns share the same missing pattern and all have product_id of 0.
3. Data Type:
   - convert `product_id`, `transaction_id` and `customer_id` to string
   - convert `product_first_sold_date` to datetime
4. Checking outliers: No outliers
5. Checking Timeline

[197]:
```
print('missing value \n', transactions.isnull().sum())
print('\n shape', transactions.shape)
transactions.head(2)
```

```
missing value
 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

 shape (20000, 13)
```

[197]:

```
   transaction_id  product_id  customer_id transaction_date  online_order  \
0               1           2         2950       2017-02-25           0.0
1               2           3         3120       2017-05-21           1.0

  order_status          brand product_line product_class product_size  \
0     Approved          Solex     Standard        medium       medium
1     Approved  Trek Bicycles     Standard        medium        large

   list_price  standard_cost  product_first_sold_date
0       71.49          53.62                  41245.0
1     2091.47         388.92                  41701.0
```

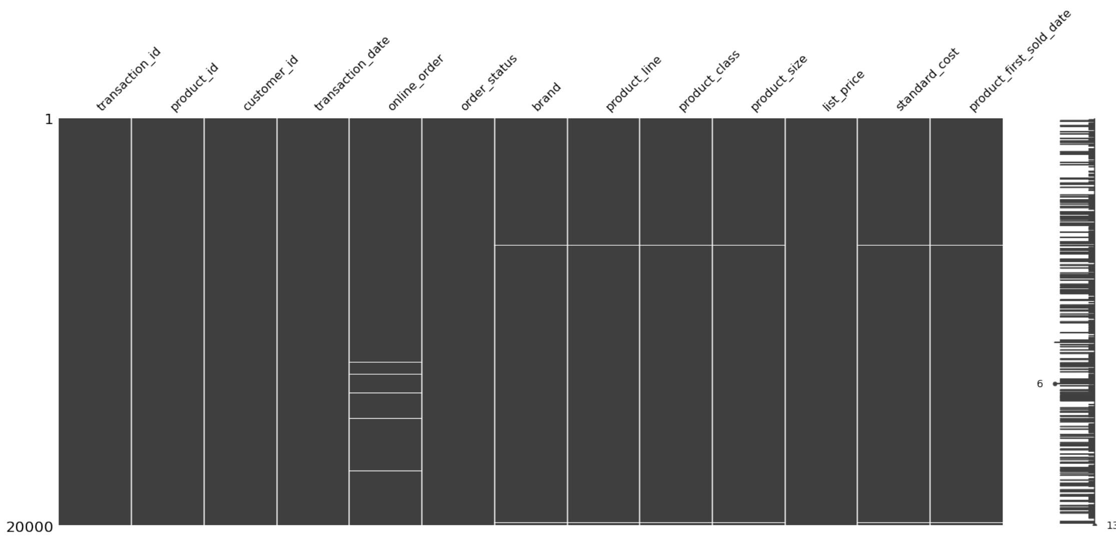
[198]:

```
# 1
assert transactions.shape[0] == transactions.transaction_id.nunique()
```

[199]:

```
# 2
msno.matrix(transactions);
missing = transactions[transactions.brand.isnull()]
print('product_id of missing value', missing.product_id.value_counts().index[0])
```

```
product_id of missing value 0
```



[200]:
```
# 3

transactions['transaction_id'] = transactions['transaction_id'].astype(str)
transactions['product_id'] = transactions['product_id'].astype(str)
transactions['customer_id'] = transactions['customer_id'].astype(str)

transactions['product_first_sold_date'] = pd.to_timedelta(transactions.
↪product_first_sold_date, unit='D') + pd.to_datetime('1899-12-30')
```

[201]:
```
print(transactions.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20000 entries, 0 to 19999
Data columns (total 13 columns):
transaction_id             20000 non-null object
product_id                 20000 non-null object
customer_id                20000 non-null object
transaction_date           20000 non-null datetime64[ns]
online_order               19640 non-null float64
order_status               20000 non-null object
brand                      19803 non-null object
product_line               19803 non-null object
product_class              19803 non-null object
product_size               19803 non-null object
list_price                 20000 non-null float64
standard_cost              19803 non-null float64
product_first_sold_date    19803 non-null datetime64[ns]
dtypes: datetime64[ns](2), float64(3), object(8)
memory usage: 2.0+ MB
```

```
None
```

[202]:
```
# 4
print(transactions[['list_price', 'standard_cost']].describe())
```

```
         list_price  standard_cost
count  20000.000000   19803.000000
mean    1107.829449     556.046951
std      582.825242     405.955660
min       12.010000       7.210000
25%      575.270000     215.140000
50%     1163.890000     507.580000
75%     1635.300000     795.100000
max     2091.470000    1759.850000
```

[203]:
```
# 5
print(transactions['transaction_date'].min(), transactions['transaction_date'].
 ↪max())
print(transactions['transaction_date'].nunique())

pd.date_range(start = '2017-01-01', end = '2017-12-30' ).
 ↪difference(transactions.transaction_date.unique())
```

```
2017-01-01 00:00:00 2017-12-30 00:00:00
364
```

[203]:
```
DatetimeIndex([], dtype='datetime64[ns]', freq=None)
```

[204]:
```
# save dataframe
transactions.to_csv('data/transactions.csv', index = False)
```

0.1.3 Customer Demographic

- customer_id: customer id
- first_name: first anme
- last_name: last name
- gender: gender
- past_3_years_bike_related_purchases: the number of bike-related pourchses customer places in past 3 years
- DOB: Date of Birth
- job_title: job title
- job_industry_category: the industry category working on
- wealth_segment: Mass Customer, High Net Worth, Affluent Customer
- deceased_indicator: the customer is dead or not Y- yes, N- No
- default
- owns_car: owning car or not. Yes, No
- tenure: the length of time an employee has worked for their employer

0.1.4 Data Wrangling

1. Uniqueness: The dataset have 4000 rows and 13 columns, Each row contains one unique customer's demographic info.
2. Missing values: there are 125 missing values on `last_name`, 506 on `job_title`, 656 on `job_indutry_category`, 302 on `defalut` and 87 on `tenure`.
3. Validity:
    - DOB: The smallest DOB is 1843-12-21, it will convert to NaN. (one possible reason is misspelling 1943-12-21)
    - Gender: U - Unknown, converting to NaN, F, Femal - Female, M - Male
4. Accuracy: `default` not being decoded properly
5. Consistensy: check other column, no erroneous values
6. Relevency: `customer_id` convert to str

[205]:
```
print('missing value \n', customerDemographic.isnull().sum())
print('-------')
print('shape', customerDemographic.shape)
print('-------')
print('Data Type', customerDemographic.info())
customerDemographic.head()
```

```
missing value
 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
-------
shape (4000, 13)
-------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 0 to 3999
Data columns (total 13 columns):
customer_id                            4000 non-null int64
first_name                             4000 non-null object
last_name                              3875 non-null object
gender                                 4000 non-null object
past_3_years_bike_related_purchases    4000 non-null int64
DOB                                    3913 non-null datetime64[ns]
```

```
job_title                             3494 non-null object
job_industry_category                 3344 non-null object
wealth_segment                        4000 non-null object
deceased_indicator                    4000 non-null object
default                               3698 non-null object
owns_car                              4000 non-null object
tenure                                3913 non-null float64
dtypes: datetime64[ns](1), float64(1), int64(2), object(9)
memory usage: 406.3+ KB
Data Type None
```

[205]:

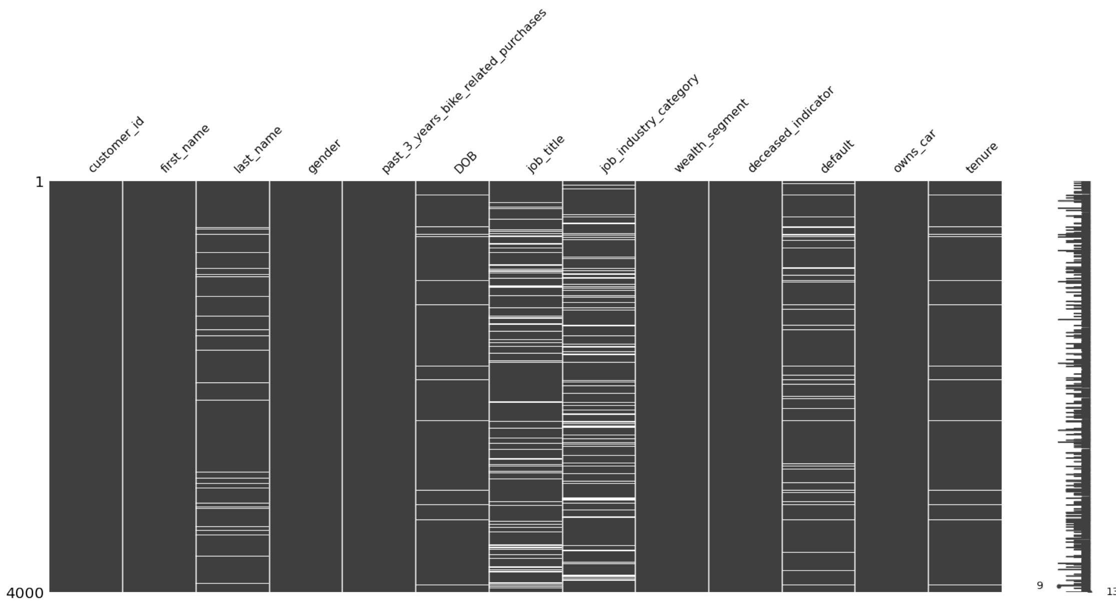
```
   customer_id      first_name  last_name  gender  \
0            1         Laraine  Medendorp       F
1            2             Eli    Bockman    Male
2            3           Arlin     Dearle    Male
3            4          Talbot        NaN    Male
4            5  Sheila-kathryn     Calton  Female

   past_3_years_bike_related_purchases        DOB               job_title  \
0                                   93 1953-10-12     Executive Secretary
1                                   81 1980-12-16  Administrative Officer
2                                   61 1954-01-20      Recruiting Manager
3                                   33 1961-10-03                     NaN
4                                   56 1977-05-13           Senior Editor

  job_industry_category     wealth_segment deceased_indicator  \
0                Health      Mass Customer                  N
1    Financial Services      Mass Customer                  N
2              Property      Mass Customer                  N
3                    IT      Mass Customer                  N
4                   NaN  Affluent Customer                  N

                                              default owns_car  tenure
0                                                  "'      Yes    11.0
1                        <script>alert('hi')</script>      Yes    16.0
2                                 2018-02-01 00:00:00      Yes    15.0
3  () { _; } >_[$($())] { touch /tmp/blns.shellsh…       No     7.0
4                                                 NIL      Yes     8.0
```

[206]:

```python
# 1
assert customerDemographic.shape[0] == customerDemographic.customer_id.nunique()
```

[207]:

```python
# 2
msno.matrix(customerDemographic);
```



[208]:
```python
# 3
customerDemographic['DOB'] = customerDemographic['DOB'].
 ↪replace(customerDemographic.DOB.min(), np.nan)
customerDemographic['gender'] = customerDemographic['gender']\
.replace(['F', 'Femal'], 'Female')\
.replace('M', 'Male')\
.replace('U', np.nan)
```

[209]:
```python
# 4
customerDemographic.default.head()
```

[209]:
```
0                                                 "'
1                           <script>alert('hi')</script>
2                                    2018-02-01 00:00:00
3    () { _; } >_[$($())] { touch /tmp/blns.shellsh…
4                                                    NIL
Name: default, dtype: object
```

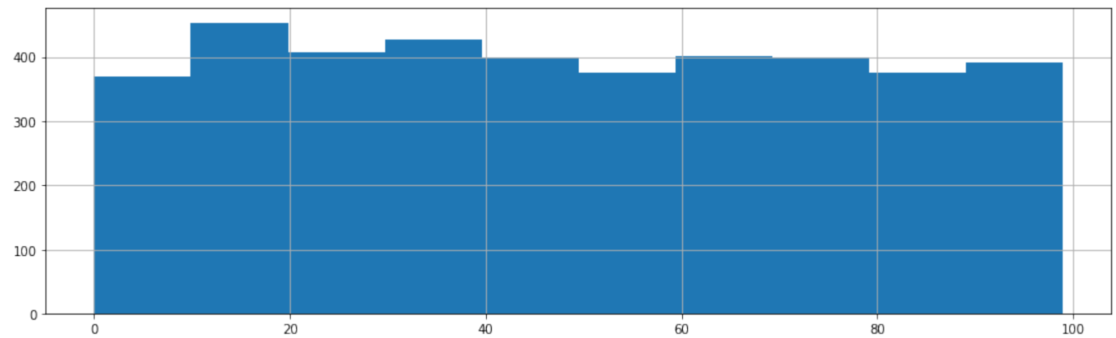
[210]:
```python
# 5
print(customerDemographic.owns_car.value_counts())
print('-----------------')
print(customerDemographic.deceased_indicator.value_counts())
print('-----------------')
print(customerDemographic.wealth_segment.value_counts())
print('-----------------')
print(customerDemographic.job_industry_category.value_counts())
print('-----------------')
```

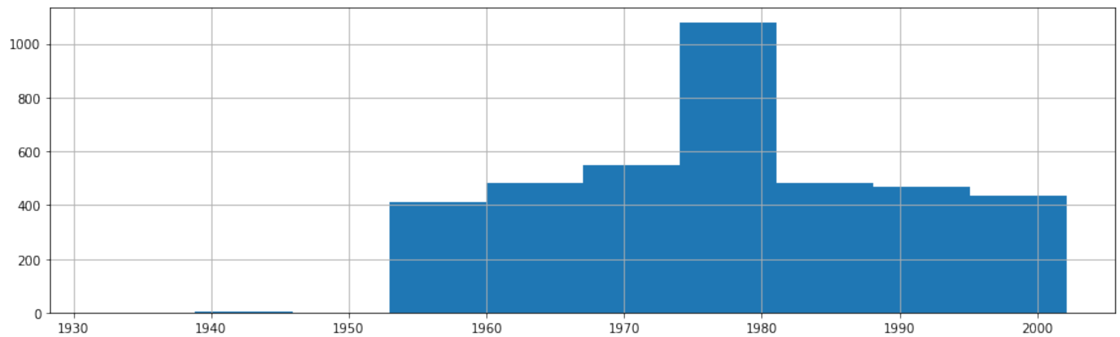
```
print(customerDemographic.gender.value_counts())
```

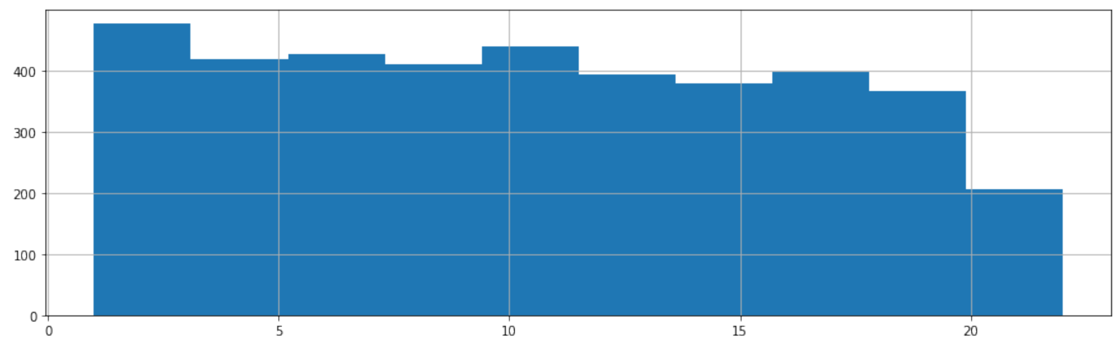
```
Yes    2024
No     1976
Name: owns_car, dtype: int64
----------------
N    3998
Y       2
Name: deceased_indicator, dtype: int64
----------------
Mass Customer        2000
High Net Worth       1021
Affluent Customer     979
Name: wealth_segment, dtype: int64
----------------
Manufacturing         799
Financial Services    774
Health                602
Retail                358
Property              267
IT                    223
Entertainment         136
Argiculture           113
Telecommunications     72
Name: job_industry_category, dtype: int64
----------------
Female    2039
Male      1873
Name: gender, dtype: int64
```

[211]:

```
plt.figure(figsize = (15, 15))
plt.subplot(3,1,1)
customerDemographic.past_3_years_bike_related_purchases.hist();
plt.subplot(3,1,2)
customerDemographic.DOB.hist();
plt.subplot(3,1,3)
customerDemographic.tenure.hist();
```







[212]:
```python
# 6
customerDemographic['customer_id'] = customerDemographic['customer_id'].
↪astype(str)
```

[213]:
```python
# save dataframe
customerDemographic.to_csv('data/customerDemographic.csv', index = False)
```

0.1.5 Customer Address

- customer_id: customer id
- address: address
- postcode: postcode
- state: state
- country: country

- property_valuation: value of the property

0.1.6 Data Wrangling

1. Uniqueness: The dataset have 3999 rows and 6 columns, Each row contains one unique customer's address info.
2. Completeness: No missing value
3. Checking outliers: no outliers
4. Relevency: `customer_id` convert to str

[214]:
```
print('missing value \n', customerAddress.isnull().sum())
print('-------')
print('shape', customerAddress.shape)
print('-------')
print('Data Type', customerAddress.info())
customerAddress.head()
```

```
missing value
 customer_id          0
address             0
postcode            0
state               0
country             0
property_valuation  0
dtype: int64
-------
shape (3999, 6)
-------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3999 entries, 0 to 3998
Data columns (total 6 columns):
customer_id           3999 non-null int64
address               3999 non-null object
postcode              3999 non-null int64
state                 3999 non-null object
country               3999 non-null object
property_valuation    3999 non-null int64
dtypes: int64(3), object(3)
memory usage: 187.5+ KB
Data Type None
```

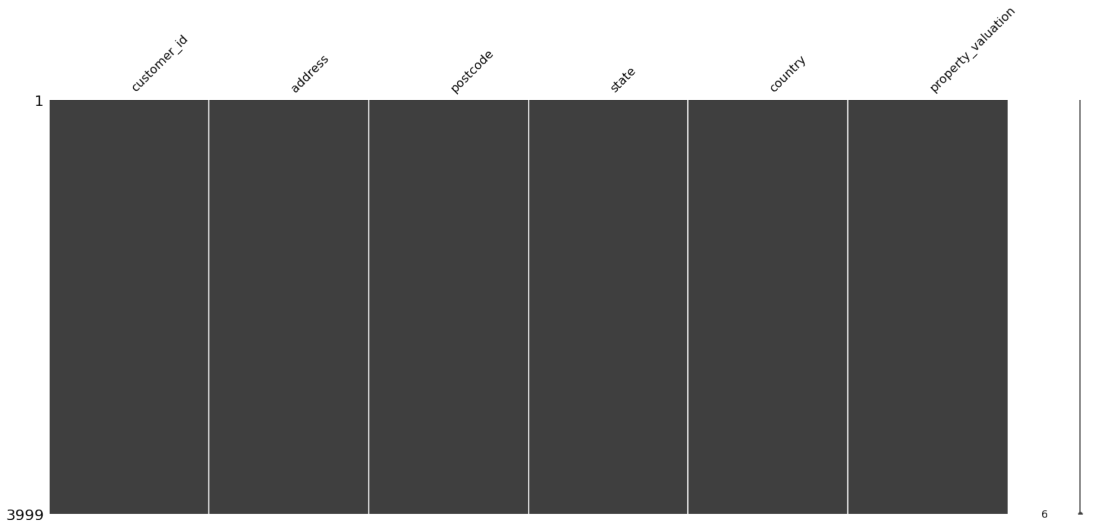
[214]:

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

```
   property_valuation
0                  10
1                  10
2                   9
3                   4
4                   9
```

[215]:
```python
# 1
assert customerAddress.shape[0] == customerAddress.customer_id.nunique()
```

[216]:
```python
# 2
msno.matrix(customerAddress);
```

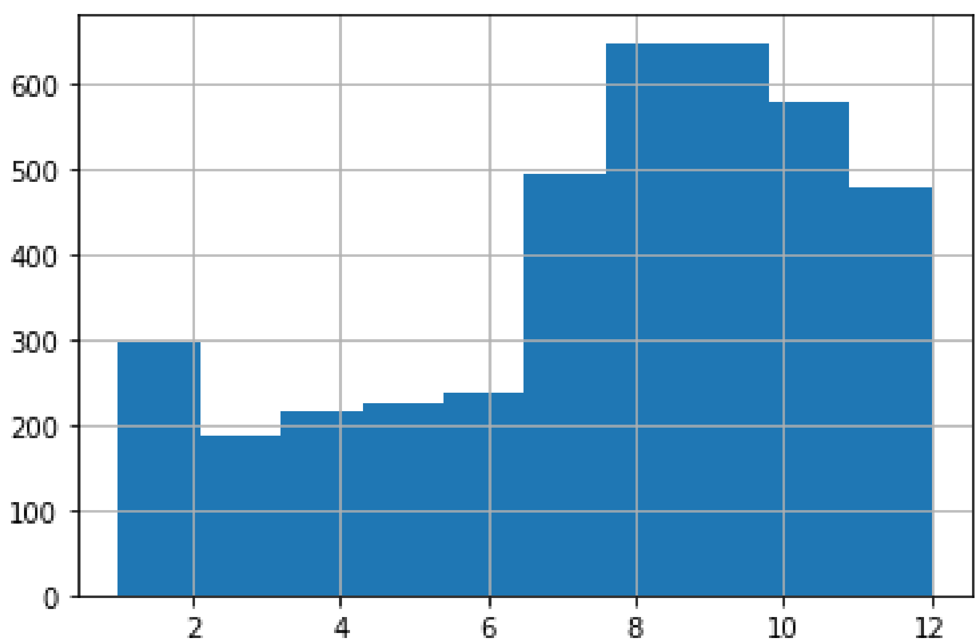


[217]:
```python
# 3
print(customerAddress.state.value_counts())
print('-------------')
print(customerAddress.country.value_counts())
print('-------------')
print(customerAddress.property_valuation.value_counts())
print('-------------')
print(customerAddress.postcode.nunique())
```

```
NSW                2054
VIC                 939
QLD                 838
New South Wales      86
Victoria             82
Name: state, dtype: int64
-------------
Australia    3999
```

```
Name: country, dtype: int64
-------------
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
-------------
873
```

[218]:
```python
customerAddress.property_valuation.hist();
```



[219]:
```python
# 4
customerAddress['customer_id'] = customerAddress['customer_id'].astype(str)
```

[220]:
```python
# save dataframe
customerAddress.to_csv('data/customerAddress.csv', index = False)
```

[ ]:

```

```

[ ]:

```

```

[ ]:

```

```