Business requirements

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.

Task 1

please find the 3 datasets attached from Sprocket Central Pty Ltd:
1) *Customer Demographic*
2) *Customer Addresses*
3) *Transaction data in the past three months*

Can you please review the data quality to ensure that it is ready for our analysis in phase two. Remember to take note of any assumptions or issues we need to go back to the client on. As well as recommendations going forward to mitigate current data quality concerns.

“Hi there – Welcome again to the team! The client has asked our team to assess the quality of their data; as well as make recommendations on ways to clean the underlying data and mitigate these issues. Can you please take a look at the datasets we’ve received and draft an email to them identifying the data quality issues and how this may impact our analysis going forward?

I will send through an example of a typical data quality framework that can be used as a guide. Remember to consider the join keys between the tables too. Thanks again for your help.”

In [47]:

```python
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import matplotlib
```

In [3]:

```python
excelFile = pd.ExcelFile("kpmg.xlsx")       # pip install openpyxl
```

In [4]:

```python
Transactions = pd.read_excel(excelFile, 'Transactions', skiprows=[0])
CustomerDemographic = pd.read_excel(excelFile,'CustomerDemographic', skiprows=[0])
CustomerAddress = pd.read_excel(excelFile, 'CustomerAddress', skiprows=[0])
pd.set_option("display.max_columns",100)
pd.set_option("display.max_rows",None)
```

In [5]:

```
Transactions.columns
```

Out[5]:

```
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 [6]:

```
Transactions = Transactions.iloc[:,0:13]
CustomerDemographic.columns
```

Out[6]:

```
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 [7]:

```
CustomerDemographic = CustomerDemographic.iloc[:,0:13]
CustomerAddress.columns
```

Out[7]:

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

In [8]:

```
CustomerAddress = CustomerAddress.iloc[:,0:6]
CustomerAddress.head(0)
```

Out[8]:

customer_id	address	postcode	state	country	property_valuation

In [9]:

```
data = pd.merge(CustomerDemographic,CustomerAddress, on="customer_id")
data = pd.merge(Transactions,data, on="customer_id")
data.to_csv("customerData.csv")
```

In [15]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19968 entries, 0 to 19967
Data columns (total 30 columns):
 #   Column                               Non-Null Count  Dtype         
---  ------                               --------------  -----         
 0   transaction_id                       19968 non-null  int64         
 1   product_id                           19968 non-null  int64         
 2   customer_id                          19968 non-null  int64         
 3   transaction_date                     19968 non-null  datetime64[ns]
 4   online_order                         19609 non-null  float64       
 5   order_status                         19968 non-null  object        
 6   brand                                19773 non-null  object        
 7   product_line                         19773 non-null  object        
 8   product_class                        19773 non-null  object        
 9   product_size                         19773 non-null  object        
 10  list_price                           19968 non-null  float64       
 11  standard_cost                        19773 non-null  float64       
 12  product_first_sold_date              19773 non-null  float64       
 13  first_name                           19968 non-null  object        
 14  last_name                            19326 non-null  object        
 15  gender                               19968 non-null  object        
 16  past_3_years_bike_related_purchases  19968 non-null  int64         
 17  DOB                                  19522 non-null  datetime64[ns]
 18  job_title                            17589 non-null  object        
 19  job_industry_category                16746 non-null  object        
 20  wealth_segment                       19968 non-null  object        
 21  deceased_indicator                   19968 non-null  object        
 22  default                              18517 non-null  object        
 23  owns_car                             19968 non-null  object        
 24  tenure                               19522 non-null  float64       
 25  address                              19968 non-null  object        
 26  postcode                             19968 non-null  int64         
 27  state                                19968 non-null  object        
 28  country                              19968 non-null  object        
 29  property_valuation                   19968 non-null  int64         
dtypes: datetime64[ns](2), float64(5), int64(6), object(17)
memory usage: 4.7+ MB
```

In [59]:

```
# checking what type of values do each of the columns in the dataset take
print(" Size of the data set",data.shape,"\n\n","Number of Unique values per column \n"
)
for column in data.columns:
    print("\n")
    if(data[column].unique().shape[0] ==1 ):
        print("column " ,column, " has zero variance")
    elif(data[column].unique().shape[0] > 1 and data[column].unique().shape[0] < 100):
        print(column," : ",data[column].unique().shape[0])
        print(data[column].unique())
    else:
        print("column " ,column, " has high variance")
```

```
 Size of the data set (19968, 30)

 Number of Unique values per column



column  transaction_id  has high variance


column  product_id  has high variance


column  customer_id  has high variance


column  transaction_date  has high variance


online_order  :  3
[ 0.  1. nan]


order_status  :  2
['Approved' 'Cancelled']


brand  :  7
['Solex' 'Giant Bicycles' 'Trek Bicycles' 'WeareA2B' 'OHM Cycles'
 'Norco Bicycles' nan]


product_line  :  5
['Standard' 'Road' 'Touring' 'Mountain' nan]


product_class  :  4
['medium' 'high' 'low' nan]


product_size  :  4
['medium' 'large' 'small' nan]


column  list_price  has high variance


column  standard_cost  has high variance


column  product_first_sold_date  has high variance


column  first_name  has high variance


column  last_name  has high variance


gender  :  6
['Male' 'Female' 'U' 'F' 'M' 'Femal']
```

```
column  past_3_years_bike_related_purchases  has high variance


column  DOB  has high variance


column  job_title  has high variance


job_industry_category  :  10
['Financial Services' 'Health' 'Retail' 'Property' 'Manufacturing'
 'Entertainment' 'IT' nan 'Argiculture' 'Telecommunications']


wealth_segment  :  3
['Mass Customer' 'Affluent Customer' 'High Net Worth']


deceased_indicator  :  2
['N' 'Y']


default  :  91
['ã»(ï¿£âï¿£)ã»:*:' 'ZÌ®ÌÍÌ\xa0ÍÍAÌ¥ÌÌÍÌ»ÌLÌ£ÍÍÌ¯Ì¹ÌÍGÌ»OÌ\xadÌÌ®'
 'testâ\xa0testâ«' 'âºâ´âµâââ'
 'Ì¡ÍÍÌIÍÌÌÌ¦nÍÍÍvÌ®Ì«okÌ²Ì«ÌÍiÌÍÌ\xadÌ¹Ì\xa0ÌnÌ¡Ì»Ì®Ì£ÌºgÌ²ÍÍÌ\xadÍÌ¬Í Ìº
tÍÌ¦hÌÌ²eÌ¢Ì¤ ÍÌ¬Ì²ÍfÌ´ÌÍÌ£eÍÍeÌ£Ì¥Ì©lÍÍÍiÍ\xa0ÍÍÌ¦nÍÍÌÍÌ³Ì®gÍ Ì¨oÍ¡ÍÌªfÌÌ
£Ì¬ ÌÌÍÌÍÌ®cÒÍÌ«ÍÍÍÍhÌµÌ¤Ì£ÍÍaÍÌÌ¼ÍÍoÌ¼Ì£Ì¥sÍ¢Ì±ÍÌºÌÌ¦Ì».ÌÌ'
 'á' 'Ù¡Ù¢Ù£' nan '1/0' '\'"\'\'\'\'"' datetime.datetime(2018, 2, 1, 0, 0)
 '"\'' '(ï½¡â â âï½¡)' 'ï½ï½¨(Â´âï½â©' 'âºâ´âµ' 100 'á\xa0' 'é¨è½æ\xa0¼'
 'ð¾ ð ð ð ð ð ð ð§' 'ãã¼ãã£ã¼ã¸è¡ããªãã' '(ï¾à²¥çà²¥ï¼ï¾ï»¿ â»ââ»' 'â¢'
 -100 "'" '\'\'\'\'"' 'â©testâ©'
 '() { 0; }; touch /tmp/blns.shellshock1.fail;' 'ðµ ð ð ð'
 'TÌÌ\xadÌºÌºoÍ Ì·iÌ²Ì¬ÍÌªÍnÌÌÍvÍÌÌÌÌ¦oÌ¶ÌÌºÌ\xa0keÍÍÌ®ÌºÌªÌ¹Ì±Ì¤ ÌtÍÌÍÌ³Ì
£Ì»ÌªhÌ¼ÍÌ²Ì¦Ì³ÌÌ²eÍÌ£ÌºÌ¦Ì¬Í Ì¢Ì¼Ì»Ì±ÌhÍÍÍÍÌÌ£Ì²iÌ¦Ì²Ì£ÌºÌ¤vÌ»ÍeÌºÌ\xadÌ³
ÌªÌº-mÌ¢iÍnÌÌºÌÌ²Ì¯ÌºdÌµÌ¼ÌÍÌ©Ì¼ÌÌ³ ÌÌ¥Ì±Ì³Ì\xadrÌÌÌeÍpÍ\xa0rÌ¼ÌÌ»Ì\xadÌeÍ
ÌºÌ\xa0Ì£sÌ'
 'Ì¦HÍÌ¬Ì¤ÌÌ¤eÍ ÍÌÌ¥ÌÌ»ÍÌwÌhÌÌ¯ÍoÌÍÌÍÌ±Ì® ÒÌºÌÌÌÍWÌ·Ì¼Ì\xadaÌºÌªÍiÌ¨ÍÍÌ\xa
dÍÌ¯ÌtÌ¶Ì¼Ì®sÌÌÍÍ Ì\xa0Ì«Ì\xa0BÌ»ÍÍÍÍÌ³eÌµhÌµÌ¬ÍÌ«ÍiÌÌ¹ÍÌ³Ì³Ì®ÍÌ«nÍdÌ´ÌªÌÌ
ÍÌºÍÌ©ÍÍÍÌ²TÍ¢ÍÌ¼ÍÌªhÍÍÌ®Ì»eÌ¬ÌÍÌ Ì¤Ì¹ÌWÍÍÍÌÌÍÍaÍÍÍÌ¹Ì¼'
 -1 'ð©ð½' '__ï¾(,_,*)' 'ï»¿'
 1000000000000000049861653971908893017010268485438462151574892930611988399
099305815384459015356416
 'âð¿ ðªð¿ ðð¿ ðð¿ ðð¿ ðð¿' '/dev/null; touch /tmp/blns.fail ; echo'
 'Î©âÃ§ââ«ËÂµâ¤â¥Ã·' 'ï¼ï¼ï¼' '(â¯Â°â¡Â°ï¼â¯ï¸µ â»ââ»)' '0/0'
 ',ãã»:*:ã»ãâ( â» Ï â» )ãã»:*:ã»ãâ'
 '×Ö¼Ö°×¨Öµ××©×Ö´××ª, ×Ö¼Ö¸×¨Ö¸× ×Ö±×Ö¹×Ö´××, ×Öµ×ª ×Ö·×©Ö¼×Ö¸×Ö·×Ö´×, ×Ö°
×Öµ×ª ×Ö¸×Ö¸×¨Ö¶×¥'
 '../../../../../../../../../../../etc/passwd%00' 'âââ'
 '×Ö¸×Ö°×ªÖ¸×testØ§ÙØµÙØ\xadØ§Øª Ø§ÙØªÙØ\xadÙÙ'
 "ËÉnbá´lÉ ÉuÆÉÉ¯ ÇÉ¹olop ÊÇ ÇÉ¹oqÉl Ên Êunpá´pá´Éuá´ É¹odÉ¯ÇÊ poÉ¯sná´Ç o
p pÇs 'Êá´lÇ Æuá´Ésá´dá´pÉ É¹nÊÇÊÉÇsuoÉ 'ÊÇÉ¯É Êá´s É¹olop É¯nsdá´ É¯ÇÉ¹oË
¥"
 ",./;'[]\\-=" '00ËÆ$-' "1'; DROP TABLE users--" 'Â¸ËÃâÄ±ËÃÂ¯ËÂ¿'
 'ð\xa0ð\xa0±ð\xa0¹ð\xa0±ð\xa0±¸ð\xa0²ð\xa0³'
 '() { _; } >_[$($())] { touch /tmp/blns.shellshock2.fail; }'
 'ÅâÂ´Â®â\xa0Â¥Â¨ËÃ¸Ïââ' 'ã½à¼¼àºÙÍàºà¼½ï¾ ã½à¼¼àºÙÍàºà¼½ï¾'
 'â¤ï¸ ð ð ð ð ð ð ð ð ð ð ð ð ð ð' 'â¦testâ§' -0.5
 '0ï¸â£ 1ï¸â£ 2ï¸â£ 3ï¸â£ 4ï¸â£ 5ï¸â£ 6ï¸â£ 7ï¸â£ 8ï¸â£ 9ï¸â£ ð'
 "<svg><script>0<1>alert('XSS')</script>"
 'ÌÌºÍÌ¹Ì¯ÍTÌ±Ì¤ÍÌ¥ÍÍhÍÌ²eÍÍÌ¼ÌÌÌ¼Ì£Í ÍÌÌ±Ì\xa0ÍÍÍNÍ\xa0ÍeÌÌ±zÌÌÌÌºÍpÌ¤ÌºÌ
¹ÍÌ¯ÍeÍÌ\xa0Ì»Ì\xa0rÌ¨Ì¤ÍÌºÌÍÌÌdÍÌ\xa0ÌÌ\xadÌ¬ÌiÌ¦ÍÌ©ÍÍÌ¤aÌ\xa0ÌÌ¬ÍÌnÍÍ
Ì»ÌÌ°ÍÍhÌµÍiÌ³ÌvÌ¢ÍeÍÌ\xadÍ-ÒÌ\xadÌ©Ì¼ÍmÌ¤Ì\xadÌ«iÍÍÌÌ¦nÌÍdÌ£Ì ÍÌ¯Ì²ÍoÌ¨ÌÌ
¯Ì°Ì²'
 'Â¡â¢Â£Â¢âÂ§Â¶â¢ÂªÂºââ\xa0' "<img src=x onerror=alert('hi') />" 1
 'ï¼ï½¥â¿ã¾â²(ï½¡ââ¿âï½¡)â±â¿ï½¥ï¾' 0 'â«testâ«' 'ã'
 'ç¤¾æç§å\xad¸é¢èªå\xad¸ç\xa0ç©¶æ'
 '../../../../../../../../../../../etc/hosts' '<>?:"{}|_+' 'â' 'åè£½æ¼¢èª'
 'ÅâÂ´â°ËÃÂ¨ËÃâââ' 'ð' 'âªâªtestâª' 'ç°ä¸\xadãããã«ããã¦ä¸ãã'
 'ì¬íê³¼íì ì´íì°êµ¬ì' 'Ã¥ÃâÆÂ©ËâËÂ¬â¦Ã¦' 'â£' '`ââ¬â¹âºï¬ï¬â¡Â°Â·ââÂ±'
 '1;DROP TABLE users' '!@#$%^&*()' 'â¡'
 'Ø«Ù ÙÙØ³ Ø³ÙØ·Øª ÙØ¨Ø§ÙØªØ\xadØ¯ÙØ¯Ø, Ø¬Ø²ÙØ±ØªÙ Ø¨Ø§Ø³ØªØ®Ø¯Ø§Ù Ø£Ù Ø¯Ù
Ù. Ø¥Ø° ÙÙØ§Ø Ø§ÙØ³ØªØ§Ø± ÙØªÙØµÙØ¨ ÙØ§Ù. Ø£ÙÙÙ Ø§ÙØ·Ø§ÙÙØ§Ø Ø¨Ø±ÙØ·Ø§ÙÙØ§
-ÙØ±ÙØ³Ø§ ÙØ¯ Ø£Ø®Ø°. Ø³ÙÙÙØ§ÙØ Ø¥ØªÙØ§ÙÙØ© Ø¨ÙÙ ÙØ§, ÙØ°ÙØ± Ø'
 'ÃÃÃÃËÃÃï£¿ÃÃÃâ' 'nil' 'NIL' 'ð¾ ð ð ð ð ð ð ð'
 "<script>alert('hi')</script>" 'ì¸ëë°í\xa0ë¥´']


owns_car  :  2
['Yes' 'No']


tenure  :  23
[10. 22. 16.  2. 12. 18.  6.  7.  8. 13. nan 19.  4.  3.  9. 15. 17.  1.
 20. 11. 21.  5. 14.]


column  address  has high variance


column  postcode  has high variance


state  :  5
['VIC' 'NSW' 'QLD' 'Victoria' 'New South Wales']


column  country  has zero variance


property_valuation  :  12
[ 6  5  1 10  7  4  8  9 11  2 12  3]
```

In [60]:

```python
for column in Transactions:
    num_missing = Transactions[column].isnull().sum()
    if(num_missing > 0):
        print(column," : ",num_missing)
```

```
online_order  :  360
brand  :  197
product_line  :  197
product_class  :  197
product_size  :  197
standard_cost  :  197
product_first_sold_date  :  197
```

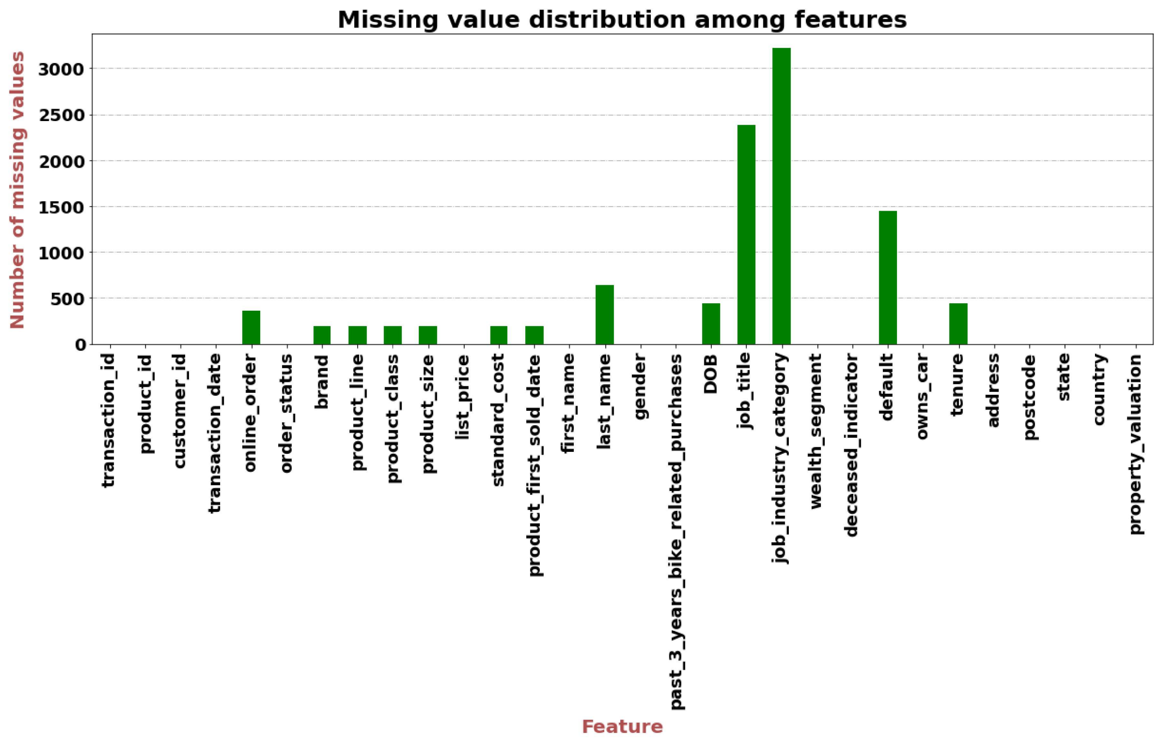
In [12]:

```
#  missing values per column
for column in data:
    num_missing = data[column].isnull().sum()
    if(num_missing > 0):
        print(column," : ",num_missing)
```

```
online_order  :  359
brand  :  195
product_line  :  195
product_class  :  195
product_size  :  195
standard_cost  :  195
product_first_sold_date  :  195
last_name  :  642
DOB  :  446
job_title  :  2379
job_industry_category  :  3222
default  :  1451
tenure  :  446
```

In [48]:

```
#sns.set_context('talk')
plt.title('Missing value distribution among features', fontsize=25, weight = 'bold')
plt.xlabel('Feature', color='#AF5050', labelpad=10, fontsize=20, weight = 'bold')
plt.ylabel('Number of missing values', color='#af5050', labelpad=10, fontsize=20, weigh
t = 'bold')
plt.rcParams['axes.axisbelow'] = True
data.isnull().sum().plot(figsize=(20, 6), color='green', rot=90,kind ='bar')
plt.xticks(fontsize=18, rotation=90,weight = 'bold')
plt.yticks(fontsize=18, weight = 'bold')
matplotlib.pyplot.grid(axis = 'y', linestyle='-.')
```



In [17]:

```
data.dtypes
```

Out[17]:

```
transaction_id                                  int64
product_id                                      int64
customer_id                                     int64
transaction_date                       datetime64[ns]
online_order                                  float64
order_status                                   object
brand                                          object
product_line                                   object
product_class                                  object
product_size                                   object
list_price                                    float64
standard_cost                                 float64
product_first_sold_date                       float64
first_name                                     object
last_name                                      object
gender                                         object
past_3_years_bike_related_purchases             int64
DOB                                    datetime64[ns]
job_title                                      object
job_industry_category                          object
wealth_segment                                 object
deceased_indicator                             object
default                                        object
owns_car                                       object
tenure                                        float64
address                                        object
postcode                                        int64
state                                          object
country                                        object
property_valuation                              int64
dtype: object
```

In [33]:

```
# checking the maximum and minimum values of numerical columns looking for possible out
liers
for column in list(data.select_dtypes(include = ["int64","float64"]).columns):
    maximum = max(data[column])
    minimum = min(data[column])
    print(column, "          max =",maximum, "          min =",minimum)
```

```
transaction_id           max = 20000           min = 1
product_id           max = 100           min = 0
customer_id           max = 3500           min = 1
online_order           max = 1.0           min = 0.0
list_price           max = 2091.47           min = 12.01
standard_cost           max = 1759.85           min = 7.21
product_first_sold_date           max = 42710.0           min = 33259.0
past_3_years_bike_related_purchases           max = 99           min = 0
tenure           max = 22.0           min = 1.0
postcode           max = 4883           min = 2000
property_valuation           max = 12           min = 1
```

In [49]:

```
for column in list(data.select_dtypes(include = ["datetime64[ns]"]).columns):
    maximum = max(data[column])
    minimum = min(data[column])
    print(column, "         max =",maximum, "         min =",minimum)
```

```
transaction_date          max = 2017-12-30 00:00:00          min = 2017-01
-01 00:00:00
DOB          max = 2002-03-11 00:00:00          min = 1843-12-21 00:00:00
```

In [58]:

```
# The date of bitth values range from 12-Dec-1843 to 11-3-2003.
data.sort_values(by="DOB").head(2)
```

Out[58]:

	transaction_id	product_id	customer_id	transaction_date	online_order	order_status
5895	1107	15	34	2017-08-22	0.0	Approved
5894	1039	8	34	2017-07-01	1.0	Approved