**To,**

**Sprocket Central Pvt. Ltd**

**Subject:** Data Quality issues and strategy to mitigate these issues.

**Respectd Sir/Ma'am**

**Below are the information of the data quality issues and strategy to mitigate these problems**

## Sprocket Central Pty Ltd


### Task 1: Data Quality Assessment

**We have Three datasets**

- Customer Demographic
- Customer Addresses
- Transaction data in the past three months

## Customer Demographic

**Null Values in the columns**

- last_name has 3.125 % data is null
- job_title has 2.175 % data is null
- DOB has 3.125 % data is null
- job_title has 12.650 % data is null
- job_industry_category has 16.400 % data is null
- default has 7.550 % data is null
- tenure has 2.175 % data is null

**mitigate these issues**

- We have to clean or fill the null values as per the dependencies that it can be fill or drop
- After cleaning or fill the null value we will select the important featrure that make give as better insights

### Categorical columns

- gender
- job_title
- job_industry_category
- wealth_segment
- deceased_indicator
- owns_car

- **Shape of the dataset is (4000, 13)**

**Note:**

- We can use the categorical columns to find the most interested categoery that will be helpfull to take the decesion

## Transactions datasets

**Null Values in the columns**

- online_order 1.800
- brand 0.985
- product_line 0.985
- product_class 0.985

- product_size 0.985
- standard_cost 0.985
- product_first_sold_date 0.985

- **Shape of the dataset is (20000, 13)**

### Numerical columns

### Categorical columns

- online order
- brand
- order status
- product line
- prodcut class
- product size

**Note:**

- Same Here categorical data can be use to find the most interested category to target them

### Note:

- Here Transaction_id, product_id and customer_id is unique
- From the transcation date i can make onw new columns moths so then i could find the monthly transactions

## Customer Address datasets

**Categorical columns**

- Postalcode
- state

- **This two columns are important in this datasets**

- **Shape of the dataset is (3999, 6)**

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

Out[72]:
```
(20000, 13)
```

In [73]:
```
ca.shape
```

Out[73]:
```
(3999, 6)
```

In [74]:
```
cd.shape
```

Out[74]:
```
(4000, 13)
```

In [68]:
```
ca.columns
```

Out[68]:
```
Index(['customer_id', 'address', 'postcode', 'state', 'country',
       'property_valuation'],
      dtype='object')
```

In [60]:
```
trans['product_class'].value_counts()
```

Out[60]:
```
medium    13826
high       3013
low        2964
Name: product_class, dtype: int64
```

In [64]:
```
trans['product_size'].value_counts()
```

Out[64]:
```
medium    12990
large      3976
small      2837
Name: product_size, dtype: int64
```

In [59]:
```
trans['product_line'].value_counts()
```

Out[59]:
```
Standard    14176
Road         3970
Touring      1234
Mountain      423
Name: product_line, dtype: int64
```

In [51]:
```
# categorical columns
trans['order_status'].value_counts()
```

Out[51]:
```
Approved     19821
Cancelled      179
Name: order_status, dtype: int64
```

In [55]:
```
trans['online_order'].value_counts()
```

Out[55]:
```
1.0    9829
0.0    9811
Name: online_order, dtype: int64
```

In [57]:
```
# branst types
trans['brand'].value_counts()
```

Out[57]:
```
Solex             4253
Giant Bicycles    3312
WeareA2B          3295
OHM Cycles        3043
Trek Bicycles     2990
Norco Bicycles    2910
Name: brand, dtype: int64
```

In [1]:
```
import pandas as  pd
import numpy as np
```

In [11]:
```
xls = pd.ExcelFile('KPMG_VI_New_raw_data_update_final.xlsx')
cd = pd.read_excel(xls, 'CustomerDemographic', header=1)
```

...

In [47]:
```
cd.head()
```

Out[47]:

| | customer_id | first_name | last_name | gender | past_3_years_bike_related_purchases | DOB | job_title | job_industry_category | wealth_segment | deceased_inc |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | Laraine | Medendorp | F | 93 | 1953-10-12 | Executive Secretary | Health | Mass Customer | |
| 1 | 2 | Eli | Bockman | Male | 81 | 1980-12-16 | Administrative Officer | Financial Services | Mass Customer | |
| 2 | 3 | Arlin | Dearle | Male | 61 | 1954-01-20 | Recruiting Manager | Property | Mass Customer | |
| 3 | 4 | Talbot | NaN | Male | 33 | 1961-10-03 | NaN | IT | Mass Customer | |
| 4 | 5 | Sheila-kathryn | Calton | Female | 56 | 1977-05-13 | Senior Editor | NaN | Affluent Customer | |

In [48]:
```
cd.columns
```

Out[48]:
```
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 [36]:
```
# Shape of the datasets
cd.shape
```

Out[36]:
```
(4000, 13)
```

In [35]:
```
cd.isnull().sum()/len(cd)*100
```

Out[35]:
```
customer_id                          0.000
first_name                           0.000
last_name                            3.125
gender                               0.000
past_3_years_bike_related_purchases  0.000
DOB                                  2.175
job_title                           12.650
job_industry_category               16.400
wealth_segment                       0.000
deceased_indicator                   0.000
default                              7.550
owns_car                             0.000
tenure                               2.175
dtype: float64
```

In [42]:
```
cd.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 [22]:
```
trans = pd.read_excel(xls, 'Transactions', header=1)
```

In [38]:
```
trans.isnull().sum()/len(trans)*100
```

Out[38]:
```
transaction_id             0.000
product_id                 0.000
customer_id                0.000
transaction_date           0.000
online_order               1.800
order_status               0.000
brand                      0.985
product_line               0.985
product_class              0.985
product_size               0.985
list_price                 0.000
standard_cost              0.985
product_first_sold_date    0.985
dtype: float64
```

In [43]:
```
trans.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 [76]:
```
!pip install pandoc
```

```
Requirement already satisfied: pandoc in c:\users\kamal\anaconda3\lib\site-packages (2.3)
Requirement already satisfied: plumbum in c:\users\kamal\anaconda3\lib\site-packages (from pandoc) (1.8.2)
Requirement already satisfied: ply in c:\users\kamal\anaconda3\lib\site-packages (from pandoc) (3.11)
Requirement already satisfied: pywin32 in c:\users\kamal\anaconda3\lib\site-packages (from plumbum->pandoc) (305.1)
```

In [49]:
```
trans.columns
```

Out[49]:
```
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 [26]:
```
ca = pd.read_excel(xls, 'CustomerAddress', header=1)
```

In [39]:
```
ca.isnull().sum()/len(ca)*100
```

Out[39]:
```
customer_id           0.0
address               0.0
postcode              0.0
state                 0.0
country               0.0
property_valuation    0.0
dtype: float64
```

In [44]:
```
ca.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 [45]:
```
ca.describe()
```

Out[45]:

| | customer_id | postcode | property_valuation |
|---|---|---|---|
| **count** | 3999.000000 | 3999.000000 | 3999.000000 |
| **mean** | 2003.987997 | 2985.755939 | 7.514379 |
| **std** | 1154.576912 | 844.878364 | 2.824663 |
| **min** | 1.000000 | 2000.000000 | 1.000000 |
| **25%** | 1004.500000 | 2200.000000 | 6.000000 |
| **50%** | 2004.000000 | 2768.000000 | 8.000000 |
| **75%** | 3003.500000 | 3750.000000 | 10.000000 |
| **max** | 4003.000000 | 4883.000000 | 12.000000 |