KPMG TASK-1

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Exploring Transactions Sheet

In [2]:
```
path=r'C:\Users\JAYADEVA JAVALI\Downloads\KPMG_VI_New_raw_data_update_final.xlsx'
df=pd.read_excel(path,sheet_name='Transactions')
```

In [3]:
```
df.head()
```

Out[3]:

	transaction_id	product_id	customer_id	transaction_date	online_order	order_status	brand	pr
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 [4]:
```
df.shape
```

Out[4]:
```
(20000, 13)
```

In [5]:
```
df.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]:
```
df.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 [10]:
```
#convert date columns from integer to datetime
df['transaction_date'] = pd.to_datetime(df['transaction_date'], unit='s')
df['transaction_date'].head()
```

Out[10]:
```
0   2017-02-25
1   2017-05-21
2   2017-10-16
3   2017-08-31
4   2017-10-01
Name: transaction_date, dtype: datetime64[ns]
```

In [11]:
```
#convert date columns from integer to datetime
df['product_first_sold_date'] = pd.to_datetime(df['product_first_sold_date'], unit='
df['product_first_sold_date'].head()
```

Out[11]:
```
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 [7]:
```
df.describe()
```

Out[7]:

	transaction_id	product_id	customer_id	online_order	list_price	standard_cost	produc
count	20000.000000	20000.00000	20000.000000	19640.000000	20000.000000	19803.000000	
mean	10000.500000	45.36465	1738.246050	0.500458	1107.829449	556.046951	
std	5773.647028	30.75359	1011.951046	0.500013	582.825242	405.955660	
min	1.000000	0.00000	1.000000	0.000000	12.010000	7.210000	
25%	5000.750000	18.00000	857.750000	0.000000	575.270000	215.140000	
50%	10000.500000	44.00000	1736.000000	1.000000	1163.890000	507.580000	
75%	15000.250000	72.00000	2613.000000	1.000000	1635.300000	795.100000	
max	20000.000000	100.00000	5034.000000	1.000000	2091.470000	1759.850000	

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
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
```

To Treat Missing Values, The options we have are:

- **Drop Missing Values**
- **Impute Missing Values based on type of variable**

We can decide on this during analysis based on objective.

In [12]:
```
dups = df.duplicated()
dups.sum()
```

Out[12]:
```
0
```

Exploring Columns

In [13]:
```
df['order_status'].value_counts()
```

Out[13]:
```
Approved     19821
Cancelled      179
Name: order_status, dtype: int64
```

In [14]:
```
df['brand'].value_counts()
```

Out[14]:
```
Solex             4253
Giant Bicycles    3312
WeareA2B          3295
OHM Cycles        3043
Trek Bicycles     2990
Norco Bicycles    2910
Name: brand, dtype: int64
```

In [15]:
```
df['product_line'].value_counts()
```

Out[15]:
```
Standard    14176
Road         3970
Touring      1234
Mountain      423
Name: product_line, dtype: int64
```

In [16]:
```
df['product_class'].value_counts()
```

Out[16]:
```
medium    13826
high       3013
low        2964
Name: product_class, dtype: int64
```

In [17]:
```
df['product_size'].value_counts()
```

Out[17]:
```
medium    12990
large      3976
small      2837
Name: product_size, dtype: int64
```

In [54]:
```
df['transaction_id'].nunique()
```

Out[54]:
```
20000
```

In [58]:
```
df['customer_id'].nunique()
```

Out[58]:
```
3494
```

In [18]:
```
df.corr().T
```

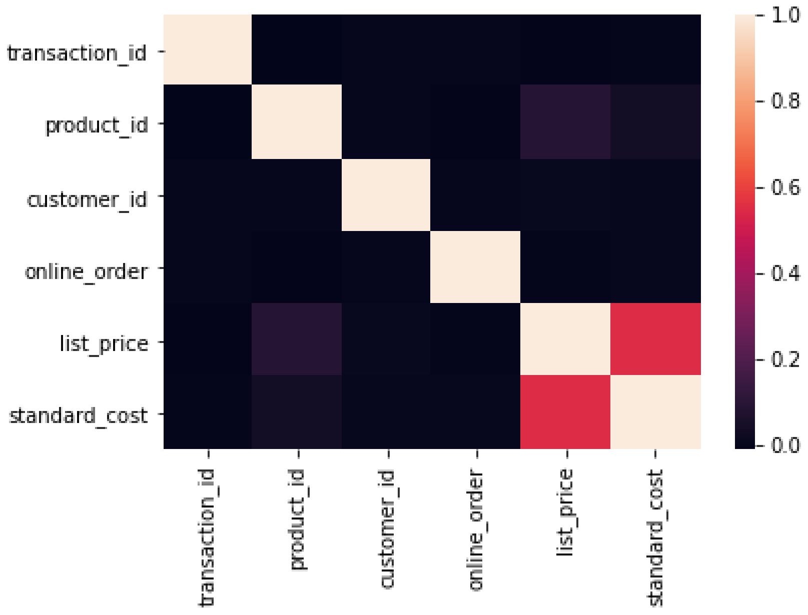
Out[18]:

	transaction_id	product_id	customer_id	online_order	list_price	standard_cost

	transaction_id	product_id	customer_id	online_order	list_price	standard_cost
transaction_id	1.000000	-0.011486	0.001753	0.003394	-0.006154	-0.003291
product_id	-0.011486	1.000000	0.004278	-0.004233	0.090066	0.038765
customer_id	0.001753	0.004278	1.000000	0.001616	0.009306	0.005365
online_order	0.003394	-0.004233	0.001616	1.000000	-0.000295	0.006934
list_price	-0.006154	0.090066	0.009306	-0.000295	1.000000	0.551539
standard_cost	-0.003291	0.038765	0.005365	0.006934	0.551539	1.000000

In [20]:
```
sns.heatmap(df.corr())
```

Out[20]:
```
<AxesSubplot:>
```



Exploring Customer Demographic

In [22]:
```
df2=pd.read_excel(path,sheet_name='CustomerDemographic')
```

In [23]:
```
df2.head()
```

Out[23]:

	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 [24]: df2.shape
```

```
Out[24]: (4000, 13)
```

```
In [25]: df2.columns
```

```
Out[25]: 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 [26]: df2.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 [27]: df2.describe()
```

Out[27]:

	customer_id	past_3_years_bike_related_purchases	tenure
count	4000.000000	4000.000000	3913.000000
mean	2000.500000	48.890000	10.657041
std	1154.844867	28.715005	5.660146
min	1.000000	0.000000	1.000000
25%	1000.750000	24.000000	6.000000
50%	2000.500000	48.000000	11.000000
75%	3000.250000	73.000000	15.000000
max	4000.000000	99.000000	22.000000

```
In [55]: df2['customer_id'].nunique()
```

```
Out[55]: 4000
```

```
In [28]: df2.isnull().sum()
```

```
Out[28]: 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
```

To Treat Missing Values, The options we have are:

- **Drop Missing Values**
- **Impute Missing Values based on type of variable** We can decide on this during analysis based on objective.

In [31]:
```
df2['gender'].value_counts()
```

Out[31]:
```
Female    2037
Male      1872
U           88
Femal        1
F            1
M            1
Name: gender, dtype: int64
```

In [33]:
```
# Replace inconsistent values with appropriate value
df2['gender'] = df2['gender'].replace('F','Female').replace('M','Male').replace('Fem
```

In [34]:
```
df2['gender'].value_counts()
```

Out[34]:
```
Female         2039
Male           1873
Unspecified      88
Name: gender, dtype: int64
```

In [35]:
```
df2['default'].value_counts()
```

Out[35]:
```
100                                         113
1                                           112
-1                                          111
-100                                         99
â°â´âµâââ                                    53
                                           ...
ç¤¾æç§å¸é¢èªå¸ç ç©¶æ                      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
```

The data doesnt seem to be right to process,so lets drop this column

In [37]:
```
df2.drop('default',axis=1)
df2.head()
```

Out[37]:

	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

	customer_id	first_name	last_name	gender	past_3_years_bike_related_purchases	DOB	job
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

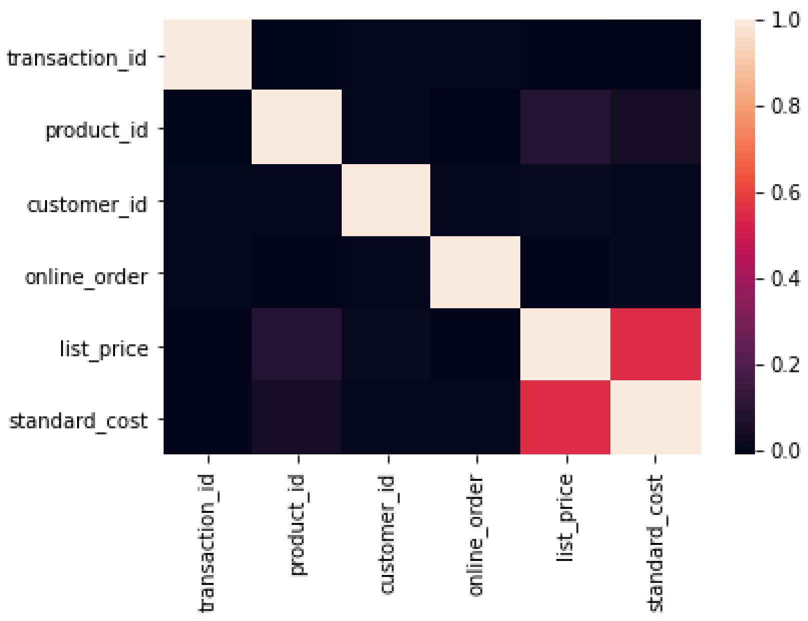
```
In [29]: df2.corr()
```

Out[29]:

	customer_id	past_3_years_bike_related_purchases	tenure
customer_id	1.000000	-0.002529	-0.019947
past_3_years_bike_related_purchases	-0.002529	1.000000	-0.009508
tenure	-0.019947	-0.009508	1.000000

```
In [30]: sns.heatmap(df.corr())
```

```
Out[30]: <AxesSubplot:>
```



Exploring Customer Address

```
In [61]: df3=pd.read_excel(path,sheet_name='CustomerAddress')
         df3.head()
```

Out[61]:

	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]:
```
df3.shape
```

Out[40]:
```
(3999, 6)
```

In [41]:
```
df3.columns
```

Out[41]:
```
Index(['customer_id', 'address', 'postcode', 'state', 'country',
       'property_valuation'],
      dtype='object')
```

In [42]:
```
df3.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 [43]:
```
df3.describe()
```

Out[43]:

	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

In [57]:
```
df3['customer_id'].nunique()
```

Out[57]:
```
3999
```

In [53]:
```
df3.isnull().sum()
```

Out[53]:
```
customer_id           0
address               0
postcode              0
state                 0
country               0
property_valuation    0
dtype: int64
```

There are no missing values here

In [48]:
```
df3['address'].value_counts()
```

Out[48]:
```
3 Talisman Place              2
```

```
3 Mariners Cove Terrace      2
64 Macpherson Junction       2
205 Melody Circle            1
376 Buena Vista Street       1
                            ..
590 Hayes Court              1
4365 Basil Junction          1
34748 Charing Cross Point    1
61 Kim Avenue                1
1 Cordelia Alley             1
Name: address, Length: 3996, dtype: int64
```

In [49]:
```
df3['postcode'].value_counts()
```

Out[49]:
```
2170    31
2155    30
2145    30
2153    29
2770    26
        ..
4552     1
4555     1
2485     1
3580     1
4421     1
Name: postcode, Length: 873, dtype: int64
```

In [50]:
```
df3['state'].value_counts()
```

Out[50]:
```
NSW                2054
VIC                 939
QLD                 838
New South Wales      86
Victoria             82
Name: state, dtype: int64
```

In [51]:
```
df3['country'].value_counts()
```

Out[51]:
```
Australia    3999
Name: country, dtype: int64
```

In [52]:
```
df3['property_valuation'].value_counts()
```

Out[52]:
```
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
```

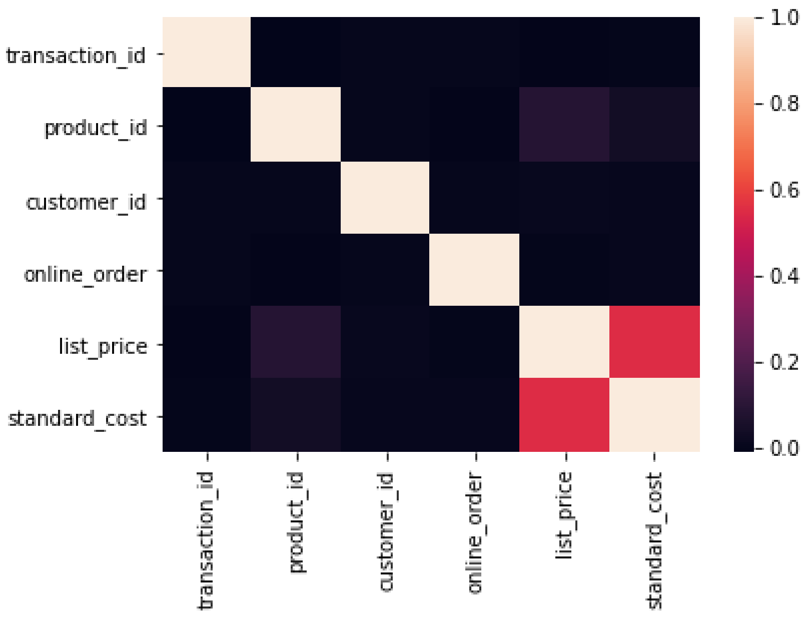
In [44]:
```
df3.corr().T
```

Out[44]:

	customer_id	postcode	property_valuation
customer_id	1.000000	0.011396	-0.012073
postcode	0.011396	1.000000	-0.508392
property_valuation	-0.012073	-0.508392	1.000000

```
In [47]: sns.heatmap(df.corr())
```

```
Out[47]: <AxesSubplot:>
```


transaction_id
product_id
customer_id
online_order
list_price
standard_cost