

Sprocket Central Pty Ltd Market Analysis

Background Information

Sprocket Central Pty Ltd , a medium size bikes & cycling accessories organisation.

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.

Strategy is believed to be the key to future growth. The following steps will be followed: Ask, Prepare, Process, Analyze, Construct, Share, Act.

```
In [1]: 1 ### 1. Ask Phase
```

Project aim

The aim of this project is to identify top customers Sprocket Central Pty Ltd should target.

2. Prepare phase

Provided with three datasets: Customer Demographic,Customer Addresses and Transactions data, here i made sure the datasets are reliable, original, comprehensive,current and cited to feel confident in them.

```
In [2]: 1 # importing necessary python packages
2 import numpy as np
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 import datetime
6 import matplotlib as mpl
7 import seaborn as sns
8 from scipy import stats
```

```
In [3]: 1 # import datasets
2 Customer_address_df0 = pd.read_excel("Customeraddress.xlsx")
3 Customer_demographic_df0 = pd.read_excel("Customerdemographic.xlsx")
4 Transactions_df0 = pd.read_excel("Transactions.xlsx")
```

3. Process Phase

```
In [4]: 1 Customer_address_df0.head(10) # Display first ten row of customer address dataframe
```

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5
0	customer_id	address	postcode	state	country	property_valuation
1	1	060 Morning Avenue	2016	New South Wales	Australia	10
2	2	6 Meadow Vale Court	2153	New South Wales	Australia	10
3	4	0 Holy Cross Court	4211	QLD	Australia	9
4	5	17979 Del Mar Point	2448	New South Wales	Australia	4
5	6	9 Oakridge Court	3216	VIC	Australia	9
6	7	4 Delaware Trail	2210	New South Wales	Australia	9
7	8	49 Londonderry Lane	2650	New South Wales	Australia	4
8	9	97736 7th Trail	2023	New South Wales	Australia	12
9	11	93405 Ludington Park	3044	VIC	Australia	8

In [5]: 1 Customer_address_df0.tail(10) # Display last ten rows of customer address dataframe

Out[5]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5
3990	3994	2918 Summer Ridge Hill		3030	VIC	Australia
3991	3995	613 Erie Lane		2088	NSW	Australia
3992	3996	0 Transport Center		3977	VIC	Australia
3993	3997	4 Dovetail Crossing		2350	NSW	Australia
3994	3998	736 Roxbury Junction		2540	NSW	Australia
3995	3999	1482 Hauk Trail		3064	VIC	Australia
3996	4000	57042 Village Green Point		4511	QLD	Australia
3997	4001	87 Crescent Oaks Alley		2756	NSW	Australia
3998	4002	8194 Lien Street		4032	QLD	Australia
3999	4003	320 Acker Drive		2251	NSW	Australia

In [6]: 1 Customer_address_df0.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 0 to 3999
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Unnamed: 0    4000 non-null   object 
 1   Unnamed: 1    4000 non-null   object 
 2   Unnamed: 2    4000 non-null   object 
 3   Unnamed: 3    4000 non-null   object 
 4   Unnamed: 4    4000 non-null   object 
 5   Unnamed: 5    4000 non-null   object 
dtypes: object(6)
memory usage: 187.6+ KB
```

In [7]: 1 Customer_address_df0.iloc[:,4].unique() # Display country

Out[7]: array(['country', 'Australia'], dtype=object)

In [8]: 1 Customer_address_df0.iloc[:,3].unique()# Display state

Out[8]: array(['state', 'New South Wales', 'QLD', 'VIC', 'NSW', 'Victoria'],
 dtype=object)

In [9]: 1 Customer_address_df0.iloc[:,5].describe(include = "all") # property_valuation Statistics

Out[9]:

	count	unique	top	freq
Name: Unnamed: 5	4000	13	9	647

Name: Unnamed: 5, dtype: int64

In [10]: 1 Customer_address_df0.to_csv("Customer_address_df0", index = False)

In [11]: 1 Customer_address_df = pd.read_csv("Customer_address_df0", header = None)

In [12]: 1 Customer_address_df.head()

Out[12]:

	0	1	2	3	4	5
0	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5
1	customer_id	address	postcode	state	country	property_valuation
2	1	060 Morning Avenue	2016	New South Wales	Australia	10
3	2	6 Meadow Vale Court	2153	New South Wales	Australia	10
4	4	0 Holy Cross Court	4211	QLD	Australia	9

In [13]: 1 Customer_address_df = Customer_address_df.drop([Customer_address_df.index[0]])

```
In [14]: 1 Customer_address_df.head()
```

	0	1	2	3	4	5
1	customer_id	address	postcode	state	country	property_valuation
2	1	060 Morning Avenue	2016	New South Wales	Australia	10
3	2	6 Meadow Vale Court	2153	New South Wales	Australia	10
4	4	0 Holy Cross Court	4211	QLD	Australia	9
5	5	17979 Del Mar Point	2448	New South Wales	Australia	4


```
In [15]: 1 Customer_address_df = Customer_address_df.rename(columns={0: 'customer_id', 1: 'address', 2: "postcode", 3:"state", 4:"count"} )
```



```
In [16]: 1 Customer_address_df
```

	customer_id	address	postcode	state	country	property_valuation
1	customer_id	address	postcode	state	country	property_valuation
2	1	060 Morning Avenue	2016	New South Wales	Australia	10
3	2	6 Meadow Vale Court	2153	New South Wales	Australia	10
4	4	0 Holy Cross Court	4211	QLD	Australia	9
5	5	17979 Del Mar Point	2448	New South Wales	Australia	4
...
3996	3999	1482 Hauk Trail	3064	VIC	Australia	3
3997	4000	57042 Village Green Point	4511	QLD	Australia	6
3998	4001	87 Crescent Oaks Alley	2756	NSW	Australia	10
3999	4002	8194 Lien Street	4032	QLD	Australia	7
4000	4003	320 Acker Drive	2251	NSW	Australia	7

4000 rows × 6 columns


```
In [17]: 1 Customer_address_df = Customer_address_df.drop(1, axis = 0).reset_index(drop =True) # drop second row
```



```
In [18]: 1 Customer_address_df.head() # validate
```

	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 [19]: 1 Customer_address_df.property_valuation.describe()
```

count	unique	top	freq
3999	12	9	647
			Name: property_valuation, dtype: object


```
In [20]: 1 max(Customer_address_df.property_valuation)
```

```
Out[20]: '9'
```



```
In [21]: 1 min(Customer_address_df.property_valuation)
```

```
Out[21]: '1'
```



```
In [22]: 1 Customer_address_df.property_valuation.unique()
```

```
Out[22]: array(['10', '9', '4', '12', '8', '6', '7', '3', '5', '11', '1', '2'],  
           dtype=object)
```

In [23]: 1 Customer_address_df.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    object  
 1   address          3999 non-null    object  
 2   postcode         3999 non-null    object  
 3   state            3999 non-null    object  
 4   country          3999 non-null    object  
 5   property_valuation 3999 non-null    object  
dtypes: object(6)
memory usage: 187.6+ KB
```

In [24]: 1 Customer_demographic_df0.head() # first five rows

Out[24]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnam
0	customer_id	first_name	last_name	gender	past_3_years_bike_related_purchases	DOB	job_title	job_industry_category	wealth_segment	deceased_in
1	1	Laraine	Medendorp	F		1953-10-12 00:00:00	Executive Secretary	Health	Mass Customer	
2	2	Eli	Bockman	Male		1980-12-16 00:00:00	Administrative Officer	Financial Services	Mass Customer	
3	3	Arlin	Dearle	Male		1954-01-20 00:00:00	Recruiting Manager	Property	Mass Customer	
4	4	Talbot	NaN	Male		1961-10-03 00:00:00	NaN	IT	Mass Customer	

In [25]: 1 Customer_demographic_df0.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4001 entries, 0 to 4000
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype  
 ---  --  
 0   Unnamed: 0      4001 non-null    object  
 1   Unnamed: 1      4001 non-null    object  
 2   Unnamed: 2      3876 non-null    object  
 3   Unnamed: 3      4001 non-null    object  
 4   Unnamed: 4      4001 non-null    object  
 5   Unnamed: 5      3914 non-null    object  
 6   Unnamed: 6      3495 non-null    object  
 7   Unnamed: 7      3345 non-null    object  
 8   Unnamed: 8      4001 non-null    object  
 9   Unnamed: 9      4001 non-null    object  
 10  Unnamed: 10     3699 non-null    object  
 11  Unnamed: 11     4001 non-null    object  
 12  Unnamed: 12     3914 non-null    object  
dtypes: object(13)
memory usage: 406.5+ KB
```

In [26]: 1 Customer_demographic_df0.to_csv("Customer_demographic_df0", index = False) # save dataframe to csv

In [27]: 1 Customer_demographic_df1 = pd.read_csv("Customer_demographic_df0", header = None) # read csv file

In [28]: 1 Customer_demographic_df1.head()

Out[28]:

	0	1	2	3	4	5	6	7	8	Unnam
0	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnam
1	customer_id	first_name	last_name	gender	past_3_years_bike_related_purchases	DOB	job_title	job_industry_category	wealth_segment	deceased_in
2	1	Laraine	Medendorp	F		1953-10-12 00:00:00	Executive Secretary	Health	Mass Customer	
3	2	Eli	Bockman	Male		1980-12-16 00:00:00	Administrative Officer	Financial Services	Mass Customer	
4	3	Arlin	Dearle	Male		1954-01-20 00:00:00	Recruiting Manager	Property	Mass Customer	

```
In [29]: 1 Customer_demographic_df1 = Customer_demographic_df1.rename(columns={0: 'customer_id', 1: 'first_name', 2: "last_name", 3:"ge
          ↵
```

```
In [30]: 1 Customer_demographic_df1.head() # validate
```

```
Out[30]:   customer_id first_name last_name gender past_3_years_bike_related_purchases date_of_birth job_title job_industry_category wealth_segment deceased
          0 Unnamed: 0 Unnamed: 1 Unnamed: 2 Unnamed: 3 Unnamed: 4 Unnamed: 5 Unnamed: 6 Unnamed: 7 Unnamed: 8
          1 customer_id first_name last_name gender past_3_years_bike_related_purchases date_of_birth job_title job_industry_category wealth_segment deceased
          2 1 Laraine Medendorp F 93 1953-10-12 00:00:00 Executive Secretary Health Mass Customer
          3 2 Eli Bockman Male 81 1980-12-16 00:00:00 Administrative Officer Financial Services Mass Customer
          4 3 Arlin Dearle Male 61 1954-01-20 00:00:00 Recruiting Manager Property Mass Customer
```

```
In [31]: 1 Customer_demographic_df1 = Customer_demographic_df1.drop(0, axis = 0).reset_index(drop =True) # drop first row
          2 Customer_demographic_df1.head() # validate
          ↵
```

```
Out[31]:   customer_id first_name last_name gender past_3_years_bike_related_purchases date_of_birth job_title job_industry_category wealth_segment deceased
          0 customer_id first_name last_name gender past_3_years_bike_related_purchases date_of_birth job_title job_industry_category wealth_segment deceased
          1 1 Laraine Medendorp F 93 1953-10-12 00:00:00 Executive Secretary Health Mass Customer
          2 2 Eli Bockman Male 81 1980-12-16 00:00:00 Administrative Officer Financial Services Mass Customer
          3 3 Arlin Dearle Male 61 1954-01-20 00:00:00 Recruiting Manager Property Mass Customer
          4 4 Talbot NaN Male 33 1961-10-03 00:00:00 NaN IT Mass Customer
```

```
In [32]: 1 Customer_demographic_df1 = Customer_demographic_df1.drop(0, axis = 0).reset_index(drop =True) # drop first row
          ↵
```

```
In [33]: 1 Customer_demographic_df1.head() # validate
```

```
Out[33]:   customer_id first_name last_name gender past_3_years_bike_related_purchases date_of_birth job_title job_industry_category wealth_segment deceased
          0 1 Laraine Medendorp F 93 1953-10-12 00:00:00 Executive Secretary Health Mass Customer
          1 2 Eli Bockman Male 81 1980-12-16 00:00:00 Administrative Officer Financial Services Mass Customer
          2 3 Arlin Dearle Male 61 1954-01-20 00:00:00 Recruiting Manager Property Mass Customer
          3 4 Talbot NaN Male 33 1961-10-03 00:00:00 NaN IT Mass Customer
          4 5 Sheila-kathryn Calton Female 56 1977-05-13 00:00:00 Senior Editor NaN Affluent Customer
```

```
In [34]: 1 Customer_demographic_df1.past_3_years_bike_related_purchases.describe()
```

```
Out[34]: count    4000
unique   100
top      16
freq     56
Name: past_3_years_bike_related_purchases, dtype: object
```

```
In [35]: 1 Customer_demographic_df1.deceased_indicator.value_counts()
```

```
Out[35]: N    3998
Y      2
Name: deceased_indicator, dtype: int64
```

```
In [36]: 1 Customer_demographic_df1.owns_car.unique()
```

```
Out[36]: array(['Yes', 'No'], dtype=object)
```

```
In [37]: 1 Customer_demographic_df1.tenure.describe()
```

```
Out[37]: count    3913
unique     22
top        7
freq     235
Name: tenure, dtype: object
```

```
In [38]: 1 Transactions_df0.head() # first five row
```

```
Out[38]: Unnamed: 0 Unnamed: 1 Unnamed: 2 Unnamed: 3 Unnamed: 4 Unnamed: 5 Unnamed: 6 Unnamed: 7 Unnamed: 8 Unnamed: 9 Unnamed: 10 Unnamed: 11
0 transaction_id product_id customer_id transaction_date online_order order_status brand product_line product_class product_size list_price standard_cost
1 1 2 2950 2017-02-25 00:00:00 False Approved Solex Standard medium medium 71.49 53.62
2 2 3 3120 2017-05-21 00:00:00 True Approved Trek Bicycles Standard medium large 2091.47 388.92
3 3 37 402 2017-10-16 00:00:00 False Approved OHM Cycles Standard low medium 1793.43 248.82
4 4 88 3135 2017-08-31 00:00:00 False Approved Norco Bicycles Standard medium medium 1198.46 381.1
```

```
In [39]: 1 Transactions_df0.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20001 entries, 0 to 20000
Data columns (total 13 columns):
 #   Column      Non-Null Count  Dtype  
 --- 
 0   Unnamed: 0   20001 non-null   object 
 1   Unnamed: 1   20001 non-null   object 
 2   Unnamed: 2   20001 non-null   object 
 3   Unnamed: 3   20001 non-null   object 
 4   Unnamed: 4   19641 non-null   object 
 5   Unnamed: 5   20001 non-null   object 
 6   Unnamed: 6   19804 non-null   object 
 7   Unnamed: 7   19804 non-null   object 
 8   Unnamed: 8   19804 non-null   object 
 9   Unnamed: 9   19804 non-null   object 
 10  Unnamed: 10  20001 non-null   object 
 11  Unnamed: 11  19804 non-null   object 
 12  Unnamed: 12  19804 non-null   object 
dtypes: object(13)
memory usage: 2.0+ MB
```

```
In [40]: 1 Transactions_df0.iloc[0:,5].unique() # order_status categories
```

```
Out[40]: array(['order_status', 'Approved', 'Cancelled'], dtype=object)
```

```
In [41]: 1 Transactions_df0.iloc[0:,6].unique() # brand categories
```

```
Out[41]: array(['brand', 'Solex', 'Trek Bicycles', 'OHM Cycles', 'Norco Bicycles',
 'Giant Bicycles', 'WeareA2B', nan], dtype=object)
```

```
In [42]: 1 Transactions_df0.iloc[0:,7].unique() # product_class
```

```
Out[42]: array(['product_line', 'Standard', 'Road', 'Mountain', 'Touring', nan],
 dtype=object)
```

```
In [43]: 1 Transactions_df0.iloc[0:,8].unique() # product_size
```

```
Out[43]: array(['product_class', 'medium', 'low', 'high', nan], dtype=object)
```

```
In [44]: 1 Transactions_df0.head(3)
```

```
Out[44]: Unnamed: 0 Unnamed: 1 Unnamed: 2 Unnamed: 3 Unnamed: 4 Unnamed: 5 Unnamed: 6 Unnamed: 7 Unnamed: 8 Unnamed: 9 Unnamed: 10 Unnamed: 11
0 transaction_id product_id customer_id transaction_date online_order order_status brand product_line product_class product_size list_price standard_cost
1 1 2 2950 2017-02-25 00:00:00 False Approved Solex Standard medium medium 71.49 53.62
2 2 3 3120 2017-05-21 00:00:00 True Approved Trek Bicycles Standard medium large 2091.47 388.92
```

```
In [45]: 1 Transactions_df0.to_csv("Transactions_df0", index = False) # save datafram to csv
2 Transactions_df1 = pd.read_csv("Transactions_df0", header = None)
3
```

```
In [46]: 1 Transactions_df1.head()
```

```
Out[46]:
```

	0	1	2	3	4	5	6	7	8	9	10	11
0	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	Unnamed: 10	Unnamed: 11
1	transaction_id	product_id	customer_id	transaction_date	online_order	order_status	brand	product_line	product_class	product_size	list_price	standard_cost
2	1	2	2950	2017-02-25 00:00:00	False	Approved	Solex	Standard	medium	medium	71.49	53.62
3	2	3	3120	2017-05-21 00:00:00	True	Approved	Trek Bicycles	Standard	medium	large	2091.47	388.92
4	3	37	402	2017-10-16 00:00:00	False	Approved	OHM Cycles	Standard	low	medium	1793.43	248.82

```
In [47]: 1 Transactions_df1 = Transactions_df1.rename(columns={0: 'transaction_id', 1: 'product_id', 2: "customer_id", 3:"transaction_d
2 Transactions_df1.head(3) # validate
```

```
Out[47]:
```

	transaction_id	product_id	customer_id	transaction_date	online_order	order_status	brand	product_line	product_class	product_size	list_price	standa
0	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	Unnamed: 10	Unna
1	transaction_id	product_id	customer_id	transaction_date	online_order	order_status	brand	product_line	product_class	product_size	list_price	standa
2	1	2	2950	2017-02-25 00:00:00	False	Approved	Solex	Standard	medium	medium	71.49	

```
In [48]: 1 Transactions_df1 = Transactions_df1.drop(0, axis = 0).reset_index(drop =True) # drop first row
```

```
In [49]: 1 Transactions_df1.head() # validate
```

```
Out[49]:
```

	transaction_id	product_id	customer_id	transaction_date	online_order	order_status	brand	product_line	product_class	product_size	list_price	standard
0	transaction_id	product_id	customer_id	transaction_date	online_order	order_status	brand	product_line	product_class	product_size	list_price	standard
1	1	2	2950	2017-02-25 00:00:00	False	Approved	Solex	Standard	medium	medium	71.49	
2	2	3	3120	2017-05-21 00:00:00	True	Approved	Trek Bicycles	Standard	medium	large	2091.47	3
3	3	37	402	2017-10-16 00:00:00	False	Approved	OHM Cycles	Standard	low	medium	1793.43	2
4	4	88	3135	2017-08-31 00:00:00	False	Approved	Norco Bicycles	Standard	medium	medium	1198.46	

```
In [50]: 1 Transactions_df1 = Transactions_df1.drop(0, axis = 0).reset_index(drop =True) # drop first row
```

```
In [51]: 1 Transactions_df1.head(3) # validate
```

```
Out[51]:
```

	transaction_id	product_id	customer_id	transaction_date	online_order	order_status	brand	product_line	product_class	product_size	list_price	standard
0	1	2	2950	2017-02-25 00:00:00	False	Approved	Solex	Standard	medium	medium	71.49	
1	2	3	3120	2017-05-21 00:00:00	True	Approved	Trek Bicycles	Standard	medium	large	2091.47	3
2	3	37	402	2017-10-16 00:00:00	False	Approved	OHM Cycles	Standard	low	medium	1793.43	2

4. Analyze phase

```
In [52]: 1 Sprocket_df0 = Transactions_df1.merge(Customer_demographic_df1, on = "customer_id", how = "inner")
```

In [53]: 1 Sprocket_df0.head(3)

	transaction_id	product_id	customer_id	transaction_date	online_order	order_status	brand	product_line	product_class	product_size	...	gender	past_3
0	1	2	2950	2017-02-25 00:00:00	False	Approved	Solex	Standard	medium	medium	...	Male	
1	11065	1	2950	2017-10-16 00:00:00	False	Approved	Giant Bicycles	Standard	medium	medium	...	Male	
2	18923	62	2950	2017-04-26 00:00:00	False	Approved	Solex	Standard	medium	medium	...	Male	

3 rows × 25 columns

In [54]: 1 Sprocket_df0.shape

Out[54]: (19997, 25)

In [55]: 1 Sprocket_df = Sprocket_df0.merge(Customer_address_df, on = "customer_id", how = "inner")

In [56]: 1 Sprocket_df.head(3)

	transaction_id	product_id	customer_id	transaction_date	online_order	order_status	brand	product_line	product_class	product_size	...	wealth_segment
0	1	2	2950	2017-02-25 00:00:00	False	Approved	Solex	Standard	medium	medium	...	Mass Customer
1	11065	1	2950	2017-10-16 00:00:00	False	Approved	Giant Bicycles	Standard	medium	medium	...	Mass Customer
2	18923	62	2950	2017-04-26 00:00:00	False	Approved	Solex	Standard	medium	medium	...	Mass Customer

3 rows × 30 columns

In [57]: 1 Sprocket_df.shape

Out[57]: (19968, 30)

In [58]: 1 Sprocket_df.dtypes

```
Out[58]: transaction_id          object
product_id            object
customer_id          object
transaction_date      object
online_order          object
order_status          object
brand                object
product_line          object
product_class         object
product_size          object
list_price            object
standard_cost         object
product_first_sold_date  object
first_name             object
last_name              object
gender                object
past_3_years_bike_related_purchases  object
date_of_birth          object
job_title              object
job_industry_category object
wealth_segment         object
deceased_indicator    object
default               object
owns_car              object
tenure                object
address               object
postcode              object
state                 object
country               object
property_valuation    object
dtype: object
```

```
In [59]: 1 missing_values_count = Sprocket_df.isnull().sum()
2 missing_values_count
```

```
Out[59]: transaction_id          0
product_id            0
customer_id           0
transaction_date      0
online_order          359
order_status          0
brand                 195
product_line          195
product_class         195
product_size          195
list_price             0
standard_cost         195
product_first_sold_date 195
first_name             0
last_name              642
gender                 0
past_3_years_bike_related_purchases 0
date_of_birth          446
job_title              2379
job_industry_category 3222
wealth_segment          0
deceased_indicator      0
default                1451
owns_car               0
tenure                  446
address                 0
postcode                 0
state                   0
country                 0
property_valuation       0
dtype: int64
```

```
In [60]: 1 # how many total missing values do we have?
2 total_cells = np.product(Sprocket_df.shape)
3 total_missing = missing_values_count.sum()
4
5 # percent of data that is missing
6 percent_missing = (total_missing/total_cells) * 100
7 print(percent_missing)
```

1.6885349893162391

```
In [61]: 1 # remove all the rows that contain a missing value since percentage missing is relatively small
2 Sprocket_df = Sprocket_df.dropna()
```

```
In [62]: 1 Sprocket_df.shape
```

```
Out[62]: (12970, 30)
```

```
In [63]: 1 Sprocket_df = Sprocket_df.drop_duplicates()
```

```
In [64]: 1 Sprocket_df.shape
```

```
Out[64]: (12970, 30)
```

```
In [65]: 1 Sprocket_df.gender.unique()
```

```
Out[65]: array(['Male', 'Female', 'F', 'Femal'], dtype=object)
```

```
In [66]: 1 # convert to Lower case
2 Sprocket_df['gender'] = Sprocket_df['gender'].str.lower()
```

```
In [67]: 1 Sprocket_df.gender.unique()
```

```
Out[67]: array(['male', 'female', 'f', 'femal'], dtype=object)
```

```
In [68]: 1 d = {'f':'female','femal':'female'}
2 Sprocket_df['gender'] = Sprocket_df['gender'].replace(d)
3 Sprocket_df.gender.unique()
```

```
Out[68]: array(['male', 'female'], dtype=object)
```

```
In [69]: 1 Sprocket_df.state.unique()
```

```
Out[69]: array(['VIC', 'NSW', 'QLD', 'Victoria', 'New South Wales'], dtype=object)
```

```
In [70]: 1 # convert to lower case
2 Sprocket_df['state'] = Sprocket_df['state'].str.lower()
3 # remove trailing white spaces
4 Sprocket_df['state'] = Sprocket_df['state'].str.strip()
```

```
In [71]: 1 Sprocket_df.state.unique()
```

```
Out[71]: array(['vic', 'nsw', 'qld', 'victoria', 'new south wales'], dtype=object)
```

```
In [72]: 1 d = {'vic':'victoria','nsw':'new south wales','qld':'queensland'}
2 Sprocket_df['state'] = Sprocket_df['state'].replace(d)
3 Sprocket_df.state.unique()
```

```
Out[72]: array(['victoria', 'new south wales', 'queensland'], dtype=object)
```

```
In [73]: 1 Sprocket_df[["date_of_birth","product_first_sold_date","transaction_date"]]
```

	date_of_birth	product_first_sold_date	transaction_date
0	1955-01-11 00:00:00	41245	2017-02-25 00:00:00
1	1955-01-11 00:00:00	37659	2017-10-16 00:00:00
2	1955-01-11 00:00:00	40487	2017-04-26 00:00:00
3	1979-02-04 00:00:00	41701	2017-05-21 00:00:00
4	1979-02-04 00:00:00	40649	2017-10-05 00:00:00
...
19960	1975-08-31 00:00:00	38991	2017-11-13 00:00:00
19962	1997-07-18 00:00:00	40649	2017-10-15 00:00:00
19963	1997-07-18 00:00:00	40410	2017-02-02 00:00:00
19964	1992-11-30 00:00:00	35378	2017-12-06 00:00:00
19965	1992-11-30 00:00:00	42226	2017-06-20 00:00:00

12970 rows × 3 columns

```
In [74]: 1 Sprocket_df["list_price"] = Sprocket_df[["list_price"]].astype('float64') # format to float datatype
2 Sprocket_df["standard_cost"] = Sprocket_df[["standard_cost"]].astype('float64') # format to float datatype
3 Sprocket_df["property_valuation"] = Sprocket_df[["property_valuation"]].astype('int') # format to int datatype
4 Sprocket_df['date_of_birth'] = pd.to_datetime(Sprocket_df['date_of_birth'], format="%Y/%m/%d") # format to datetime datatype
5 Sprocket_df['transaction_date'] = pd.to_datetime(Sprocket_df['transaction_date'], format="%Y/%m/%d") # format to datetime datatype
6 Sprocket_df["profit"] = Sprocket_df.list_price - Sprocket_df.standard_cost # create profit column
```

```
In [75]: 1 Sprocket_df['date_of_birth'].head(3) # first few row of date of birth
```

```
Out[75]: 0 1955-01-11
1 1955-01-11
2 1955-01-11
Name: date_of_birth, dtype: datetime64[ns]
```

```
In [76]: 1 Sprocket_df['transaction_date'].head(3) # first few row of transaction date
```

```
Out[76]: 0 2017-02-25
1 2017-10-16
2 2017-04-26
Name: transaction_date, dtype: datetime64[ns]
```

```
In [77]: 1 # create year column
2 Sprocket_df['year_of_birth'] = Sprocket_df['date_of_birth'].dt.year
3 Sprocket_df['transaction_year'] = Sprocket_df['transaction_date'].dt.year
```

```
In [78]: 1 Sprocket_df['transaction_year'].head(4)
```

```
Out[78]: 0 2017
1 2017
2 2017
3 2017
Name: transaction_year, dtype: int64
```

```
In [79]: 1 Sprocket_df['transaction_year'].unique() # transaction year category
```

```
Out[79]: array([2017], dtype=int64)
```

```
In [80]: 1 Sprocket_df['year_of_birth'].head(4)
2
```

```
Out[80]: 0    1955
1    1955
2    1955
3    1979
Name: year_of_birth, dtype: int64
```

```
In [81]: 1 Sprocket_df['age'] = Sprocket_df['transaction_year'] - Sprocket_df['year_of_birth']
2 # create age column
```

```
In [82]: 1 Sprocket_df.dtypes # validate
```

```
Out[82]: transaction_id          object
product_id            object
customer_id          object
transaction_date      datetime64[ns]
online_order          object
order_status          object
brand                object
product_line          object
product_class         object
product_size          object
list_price            float64
standard_cost         float64
product_first_sold_date   object
first_name             object
last_name              object
gender                object
past_3_years_bike_related_purchases  object
date_of_birth         datetime64[ns]
job_title              object
job_industry_category   object
wealth_segment         object
deceased_indicator     object
default                object
owns_car               object
tenure                 object
address                object
postcode               object
state                  object
country                object
property_valuation     int32
profit                 float64
year_of_birth           int64
transaction_year        int64
age                    int64
dtype: object
```

```
In [83]: 1 def group(age):
2     ''' group age ...
3     if age > 80:
4         age_group = "old"
5     elif age > 60:
6         age_group = "old_adult"
7     elif age > 40:
8         age_group = "middle_adult"
9     elif age > 20:
10        age_group = "early_adult"
11    else:
12        age_group = "adolescent"
13    return age_group
14
```

```
In [84]: 1 Sprocket_df["age_group"] = Sprocket_df["age"].apply(group)
```

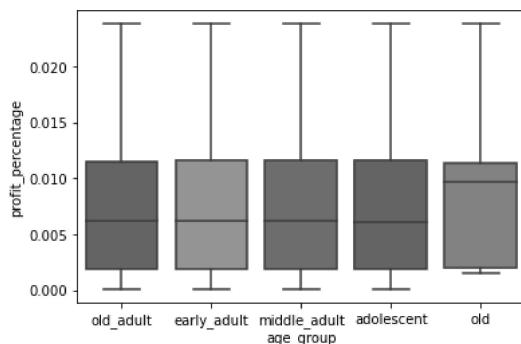
```
In [85]: 1 Sprocket_df['age_group'].unique()
```

```
Out[85]: array(['old_adult', 'early_adult', 'middle_adult', 'adolescent', 'old'],
              dtype=object)
```

```
In [86]: 1 total_profit = Sprocket_df.profit.sum()
```

```
In [87]: 1 Sprocket_df["profit_percentage"] = (Sprocket_df.profit/total_profit) * 100 # create profit percentage column
2 # Create boxplot to show distribution of profit percentage by age group
3 sns.boxplot(x = "age_group", y = "profit_percentage", data = Sprocket_df)
```

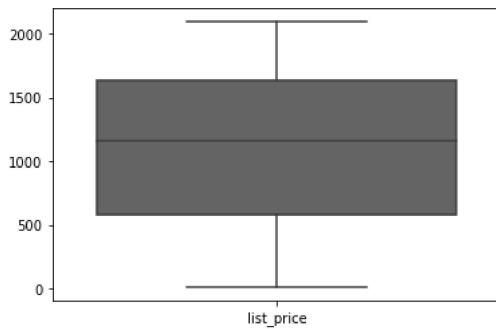
Out[87]: <AxesSubplot:xlabel='age_group', ylabel='profit_percentage'>



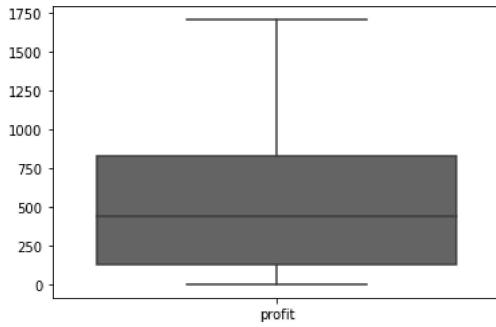
Apart from the old category, the means of other categories are nearly equal

Check for outliers

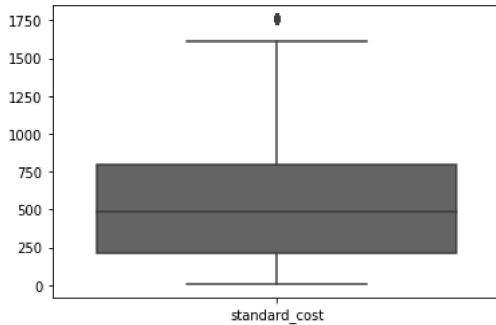
```
In [88]: 1 sns.boxplot(data = Sprocket_df[["list_price"]], showfliers = True);
```



```
In [89]: 1 sns.boxplot(data = Sprocket_df[["profit"]], showfliers = True);
```



```
In [90]: 1 sns.boxplot(data = Sprocket_df[["standard_cost"]], showfliers = True);
```



```
In [91]: 1 Sprocket_df["z_score"] = stats.zscore(Sprocket_df["standard_cost"]) # z_score column
```

```
In [92]: 1 # select rows where corresponding z score is less than 3 or greater than -3
2 Sprocket_df = Sprocket_df[(Sprocket_df["z_score"] < 3) | (Sprocket_df["z_score"] > -3)]
```

5. Construct

Run two-way ANOVA

```
In [93]: 1 # Import statsmodels and ols function
2 import statsmodels.api as sm
3 from statsmodels.formula.api import ols
```

```
In [94]: 1 Sprocket_test_df = Sprocket_df[["profit", "state", "age_group"]]
```

```
In [95]: 1 Sprocket_test_df.head()
```

```
Out[95]:
```

	profit	state	age_group
0	17.87	victoria	old_adult
1	448.68	victoria	old_adult
2	179.44	victoria	old_adult
3	1702.55	new south wales	early_adult
4	451.65	new south wales	early_adult

```
In [96]: 1 # Construct a multiple linear regression with an interaction term between color and cut
2 model = ols(formula = "profit ~ C(age_group) + C(state) + C(age_group):C(state)", data = Sprocket_test_df).fit()
   ▲
```

```
In [97]: 1 # Get summary statistics
          2 model.summary()
```

Out[97]: OLS Regression Results

Dep. Variable:	profit	R-squared:	0.001			
Model:	OLS	Adj. R-squared:	-0.000			
Method:	Least Squares	F-statistic:	0.7446			
Date:	Sun, 20 Aug 2023	Prob (F-statistic):	0.720			
Time:	11:40:16	Log-Likelihood:	-98864.			
No. Observations:	12970	AIC:	1.978e+05			
Df Residuals:	12956	BIC:	1.979e+05			
Df Model:	13					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
Intercept	526.8216	21.372	24.650	0.000	484.930	568.714
C(age_group)[T.early_adult]	32.8963	23.108	1.424	0.155	-12.399	78.192
C(age_group)[T.middle_adult]	20.3765	23.258	0.876	0.381	-25.213	65.966
C(age_group)[T.old]	194.3954	157.920	1.231	0.218	-115.151	503.942
C(age_group)[T.old_adult]	0.6418	35.128	0.018	0.985	-68.214	69.497
C(state)[T.queensland]	17.9801	40.777	0.441	0.659	-61.949	97.909
C(state)[T.victoria]	55.7610	39.746	1.403	0.161	-22.148	133.670
C(age_group)[T.early_adult]:C(state)[T.queensland]	-20.4787	43.871	-0.467	0.641	-106.471	65.514
C(age_group)[T.middle_adult]:C(state)[T.queensland]	-27.5258	44.287	-0.622	0.534	-114.335	59.283
C(age_group)[T.old]:C(state)[T.queensland]	9.489e-13	4.85e-13	1.958	0.050	-1.1e-15	1.9e-12
C(age_group)[T.old_adult]:C(state)[T.queensland]	32.8144	64.162	0.511	0.609	-92.953	158.582
C(age_group)[T.early_adult]:C(state)[T.victoria]	-65.4712	42.767	-1.531	0.126	-149.302	18.359
C(age_group)[T.middle_adult]:C(state)[T.victoria]	-38.5845	42.907	-0.899	0.369	-122.688	45.519
C(age_group)[T.old]:C(state)[T.victoria]	-527.9240	273.908	-1.927	0.054	-1064.825	8.977
C(age_group)[T.old_adult]:C(state)[T.victoria]	-45.1069	60.666	-0.744	0.457	-164.021	73.807
Omnibus:	1336.549	Durbin-Watson:	1.957			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1661.050			
Skew:	0.854	Prob(JB):	0.00			
Kurtosis:	2.606	Cond. No.	2.91e+17			

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The smallest eigenvalue is 2.41e-31. This might indicate that there are strong multicollinearity problems or that the design matrix is singular.

Null Hypothesis (age group)

$H_0: \text{profit adolescent} = \text{profit early adult} = \text{profit middle adult} = \text{profit old adult} = \text{profit old}$

There is no difference in profit based on age group.

Alternative Hypothesis (age group)

There is a difference in profit age group.

Null Hypothesis (state)

There is no difference in profit based on state.

Alternative Hypothesis (state)

There is a difference in profit based on state.

Null Hypothesis (Interaction) H_0 : The effect of age group on profit is independent of the state, and vice versa.

Alternative Hypothesis (Interaction) H_1 : There is an interaction effect between age group and state on profit.

```
In [98]: 1 sm.stats.anova_lm(model, typ = 1)
```

```
Out[98]:
```

	df	sum_sq	mean_sq	F	PR(>F)
C(age_group)	4.0	3.085021e+05	77125.532344	0.315029	0.868099
C(state)	2.0	1.236038e+05	61801.884994	0.252438	0.776908
C(age_group):C(state)	8.0	2.242839e+06	280354.827687	1.145147	0.328974
Residual	12956.0	3.171889e+09	244820.052031	NaN	NaN

Since all of the p-values (column PR(>F)) are greater than 0.05, we can fail to reject all three null hypotheses.

```
In [99]: 1 # Import Tukey's HSD function
2 from statsmodels.stats.multicomp import pairwise_tukeyhsd
```

```
In [100]: 1 # Run Tukey's HSD post hoc test for one-way ANOVA
2 tukey_oneway = pairwise_tukeyhsd(endog = Sprocket_test_df["profit"], groups = Sprocket_test_df["age_group"], alpha = 0.05)
3 # Run Tukey's HSD post hoc test for one-way ANOVA
4 tukey_onewaystate = pairwise_tukeyhsd(endog = Sprocket_test_df["profit"], groups = Sprocket_test_df["state"], alpha = 0.05)
5
```

```
In [101]: 1 # Get results (pairwise comparisons)
2 tukey_oneway.summary()
```

```
Out[101]: Multiple Comparison of Means - Tukey HSD, FWER=0.05
```

group1	group2	meandiff	p-adj	lower	upper	reject
adolescent	early_adult	13.486	0.9358	-33.5436	60.5156	False
adolescent	middle_adult	6.2263	0.9965	-41.0951	53.5478	False
adolescent	old	20.4917	0.9999	-330.7572	371.7405	False
adolescent	old_adult	-1.3433	1.0	-70.0783	67.3917	False
early_adult	middle_adult	-7.2597	0.9362	-32.6212	18.1018	False
early_adult	old	7.0056	1.0	-341.9638	355.975	False
early_adult	old_adult	-14.8293	0.9512	-70.7613	41.1027	False
middle_adult	old	14.2653	1.0	-334.7435	363.2742	False
middle_adult	old_adult	-7.5696	0.9961	-63.7473	48.608	False
old	old_adult	-21.835	0.9998	-374.3862	330.7163	False

```
In [102]: 1 # Get results (pairwise comparisons)
2 tukey_onewaystate.summary()
```

```
Out[102]: Multiple Comparison of Means - Tukey HSD, FWER=0.05
```

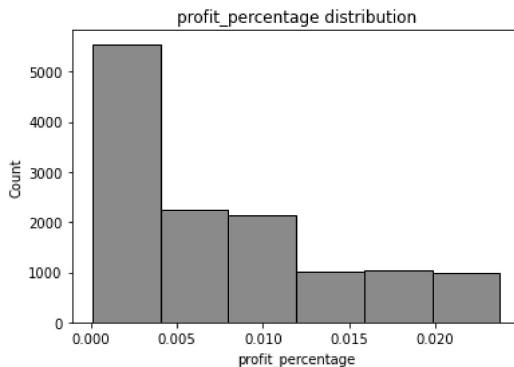
group1	group2	meandiff	p-adj	lower	upper	reject
new south wales	queensland	-1.2161	0.9933	-27.1244	24.6923	False
new south wales	victoria	6.5482	0.809	-18.1959	31.2922	False
queensland	victoria	7.7642	0.8157	-22.1669	37.6954	False

The observed difference in profit by state and age group is statistically insignificant.

6. Share phase

```
In [103]: 1 sns.histplot(data=Sprocket_df, x= "profit_percentage", bins = 6)
2
3 # Add title
4 plt.title("profit_percentage distribution")
```

Out[103]: Text(0.5, 1.0, 'profit_percentage distribution')



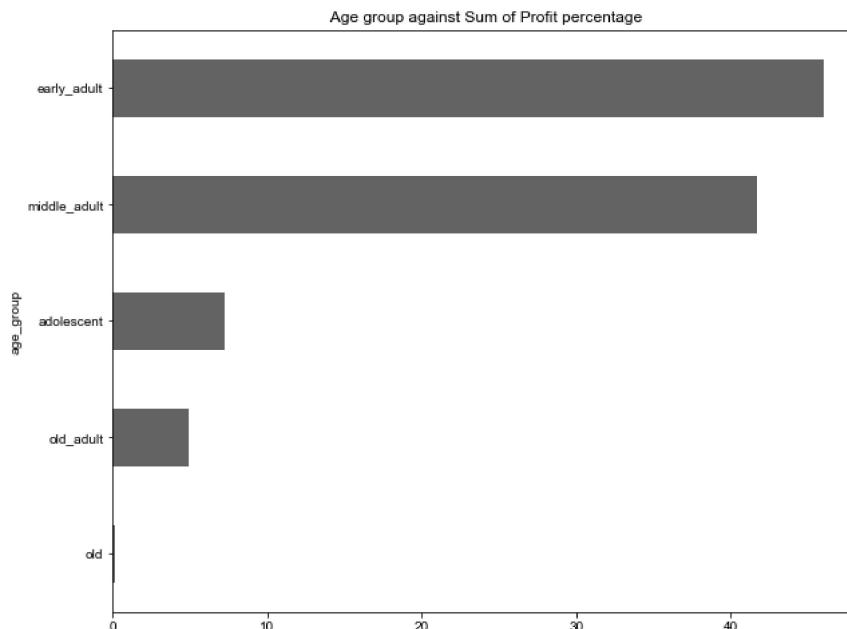
The distribution of the histogram is skewed to the right i.e count decreases as the profit percentage increases

```
In [104]: 1 Sprocket_df.groupby('age_group').profit_percentage.sum().sort_values()
```

```
Out[104]: age_group
old           0.118118
old_adult     4.889938
adolescent    7.262026
middle_adult  41.699803
early_adult   46.030116
Name: profit_percentage, dtype: float64
```

middle_adult and early_adult contribute about 92% of the profit

```
In [105]: 1 Sprocket_df.groupby('age_group').profit_percentage.sum().sort_values().plot(kind='barh',figsize=(10,8));
2 sns.set_style("darkgrid")
3 plt.title(" Age group against Sum of Profit percentage ");
4
```



```
In [106]: 1 Sprocket_df.groupby('online_order').profit_percentage.sum().sort_values()
```

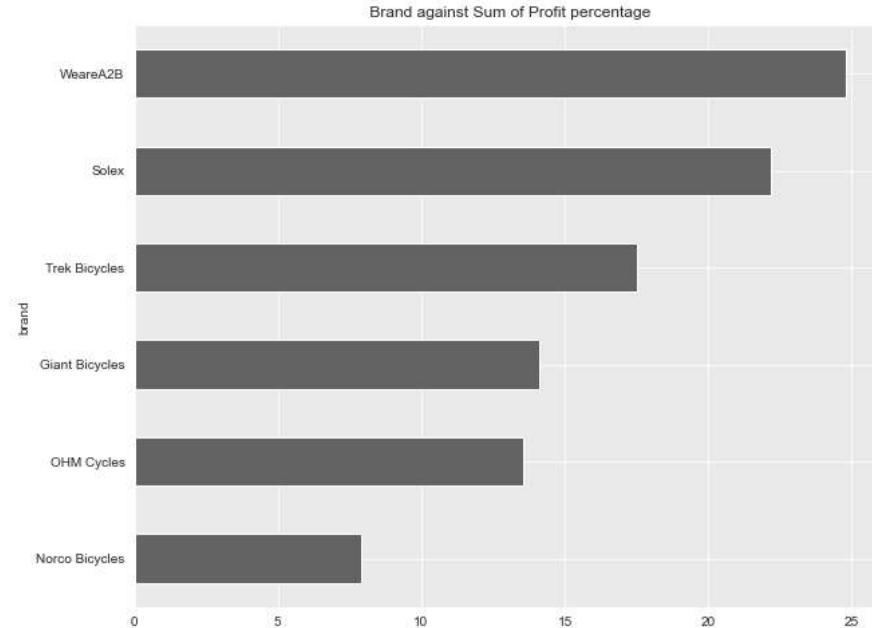
```
Out[106]: online_order
True      49.336272
False     50.663728
Name: profit_percentage, dtype: float64
```

Online order and non online order are nearly equal

```
In [107]: 1 Sprocket_df.groupby('brand').profit_percentage.sum().sort_values()
2 # How product brand contribute to profit
```

```
Out[107]: brand
Norco Bicycles    7.919267
OHM Cycles        13.542690
Giant Bicycles   14.088296
Trek Bicycles    17.497087
Solex             22.160707
WeareA2B          24.791953
Name: profit_percentage, dtype: float64
```

```
In [108]: 1 Sprocket_df.groupby('brand').profit_percentage.sum().sort_values().plot(kind='barh', figsize=(10,8));
2 sns.set_style("darkgrid")
3 plt.title(" Brand against Sum of Profit percentage");
```

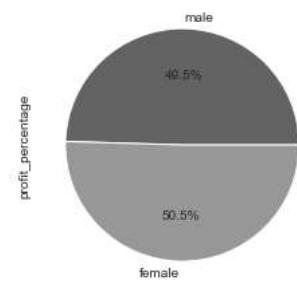


```
In [109]: 1 Sprocket_df.groupby('gender').profit_percentage.sum().sort_values()
2 # How product gender contribute to profit
```

```
Out[109]: gender
male      49.524477
female    50.475523
Name: profit_percentage, dtype: float64
```

The difference in profit generates from gender can be ignored

```
In [110]: 1 plt.figure(figsize=(4,4))
2 Sprocket_df.groupby('gender').profit_percentage.sum().sort_values().plot.pie(autopct="%1.1f%%")
3 plt.show()
```

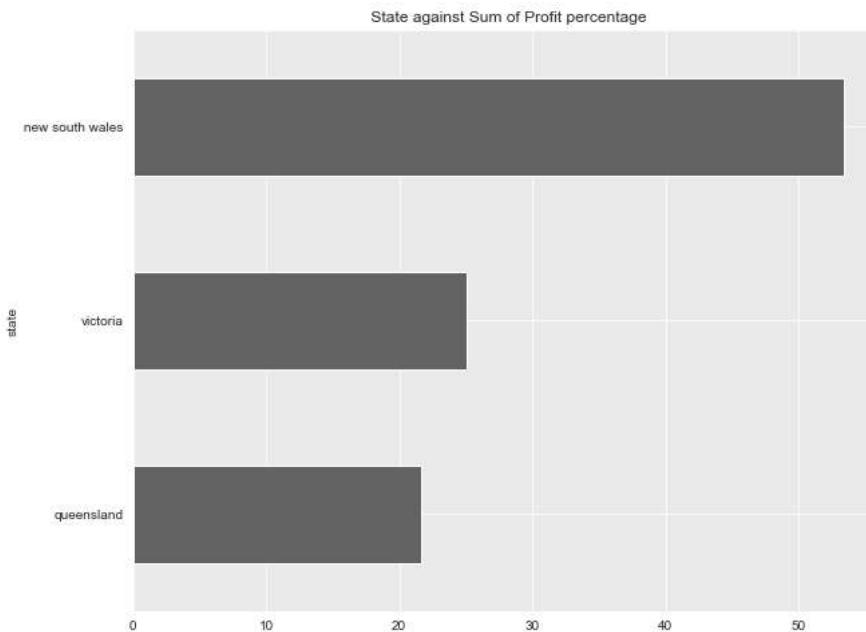


```
In [111]: 1 Sprocket_df.groupby('state').profit_percentage.sum().sort_values()
```

```
Out[111]: state
queensland      21.618061
victoria        25.013210
new south wales 53.368729
Name: profit_percentage, dtype: float64
```

About 54% of the profit being generated is from customers from new south wales

```
In [112]: 1 Sprocket_df.groupby('state').profit_percentage.sum().sort_values().plot(kind='barh', figsize=(10,8));
2 sns.set_style("darkgrid")
3 plt.title(" State against Sum of Profit percentage");
```

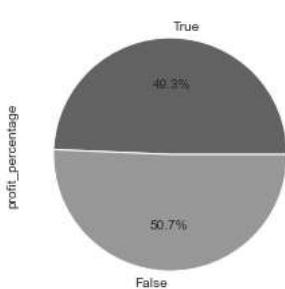


```
In [113]: 1 Sprocket_df.groupby('online_order').profit_percentage.sum().sort_values()
```

```
Out[113]: online_order
True      49.336272
False     50.663728
Name: profit_percentage, dtype: float64
```

The difference in profit generates from mode order can be ignored

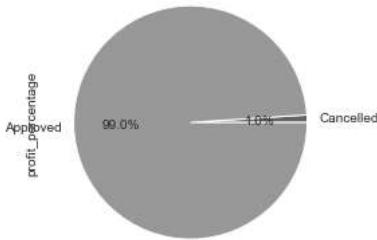
```
In [114]: 1 plt.figure(figsize=(4,4))
2 Sprocket_df.groupby('online_order').profit_percentage.sum().sort_values().plot.pie(autopct="%1.1f%%")
3 plt.show()
```



```
In [115]: 1 Sprocket_df.groupby('order_status').profit_percentage.sum().sort_values()
```

```
Out[115]: order_status
Cancelled      1.037798
Approved       98.962202
Name: profit_percentage, dtype: float64
```

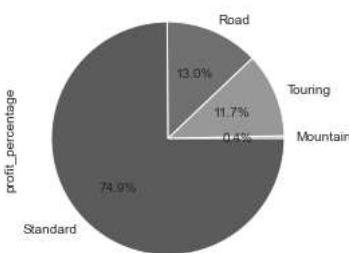
```
In [116]: 1 plt.figure(figsize=(4,4))
2 Sprocket_df.groupby('order_status').profit_percentage.sum().sort_values().plot.pie(autopct="%1.1f%%")
3 plt.show()
```



```
In [117]: 1 Sprocket_df.groupby('product_line').profit_percentage.sum().sort_values()
```

```
Out[117]: product_line
Mountain      0.361666
Touring       11.731323
Road          13.000500
Standard      74.906511
Name: profit_percentage, dtype: float64
```

```
In [118]: 1 plt.figure(figsize=(4,4))
2 Sprocket_df.groupby('product_line').profit_percentage.sum().sort_values().plot.pie(autopct="%1.1f%%")
3 plt.show()
```



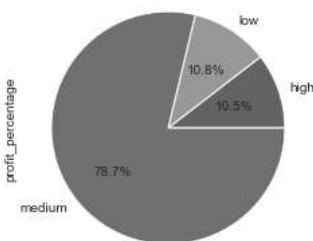
```
In [119]: 1 Sprocket_df.groupby('product_class').profit_percentage.sum().sort_values()
```

```
Out[119]: product_class
high        10.470744
low         10.783299
medium      78.745958
Name: profit_percentage, dtype: float64
```

```
In [120]: 1 Sprocket_df.groupby('product_size').profit_percentage.sum().sort_values()
```

```
Out[120]: product_size
small        3.747644
large        33.302844
medium       62.949512
Name: profit_percentage, dtype: float64
```

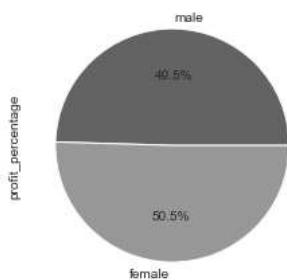
```
In [121]: 1 plt.figure(figsize=(4,4))
2 Sprocket_df.groupby('product_class').profit_percentage.sum().sort_values().plot.pie(autopct="%1.1f%%")
3 plt.show()
```



```
In [122]: 1 Sprocket_df.groupby('gender').profit_percentage.sum().sort_values()
```

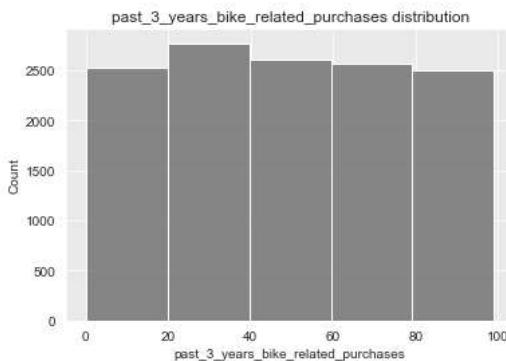
```
Out[122]: gender
male      49.524477
female    50.475523
Name: profit_percentage, dtype: float64
```

```
In [123]: 1 plt.figure(figsize=(4,4))
2 Sprocket_df.groupby('gender').profit_percentage.sum().sort_values().plot.pie(autopct="%1.1f%%")
3 plt.show()
```



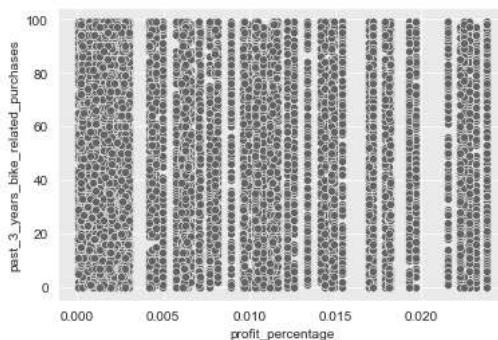
```
In [124]: 1 Sprocket_df["past_3_years_bike_related_purchases"] = Sprocket_df[["past_3_years_bike_related_purchases"]].astype('int64')
2 sns.histplot(data=Sprocket_df, x= "past_3_years_bike_related_purchases", bins = 5)
3
4 # Add title
5 plt.title("past_3_years_bike_related_purchases distribution")
```

```
Out[124]: Text(0.5, 1.0, 'past_3_years_bike_related_purchases distribution')
```



```
In [125]: 1 sns.scatterplot(data=Sprocket_df, x="profit_percentage", y="past_3_years_bike_related_purchases")
2 # No correlation
```

```
Out[125]: <AxesSubplot:xlabel='profit_percentage', ylabel='past_3_years_bike_related_purchases'>
```



```
In [126]: 1 Sprocket_df.groupby('job_title').profit_percentage.sum().sort_values(ascending = False).head(15)
```

```
Out[126]: job_title
Information Systems Manager    1.717418
Social Worker                  1.590804
Recruiting Manager             1.421926
Nuclear Power Engineer        1.356981
Internal Auditor               1.355069
General Manager                1.324777
Chemical Engineer              1.295316
Speech Pathologist              1.245013
Administrative Officer         1.207766
Registered Nurse               1.184907
Assistant Media Planner        1.180942
Food Chemist                   1.172929
Assistant Manager              1.163864
Marketing Manager              1.147413
Project Manager                1.144754
Name: profit_percentage, dtype: float64
```

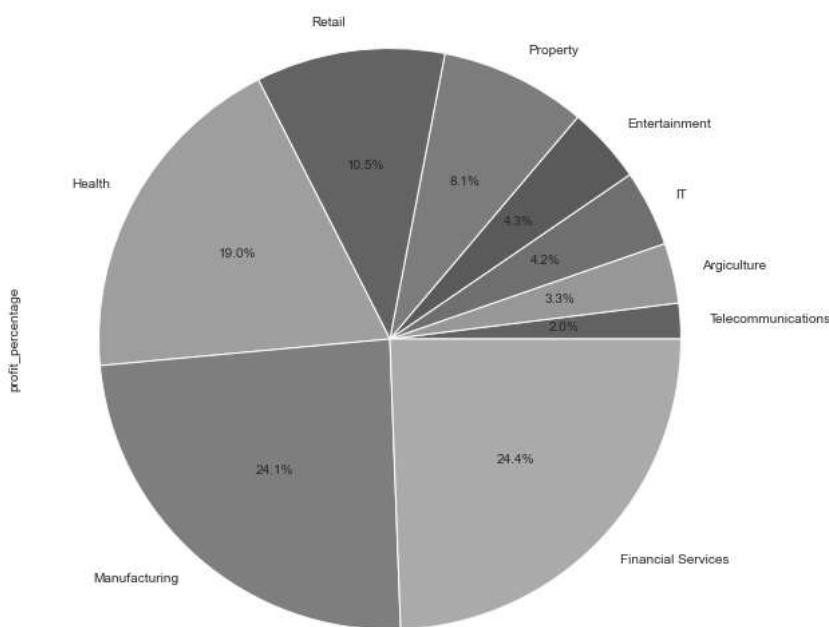
```
In [127]: 1 Sprocket_df.groupby('job_title').profit_percentage.sum().sort_values(ascending = False).tail(15)
```

```
Out[127]: job_title
Accounting Assistant II        0.089983
Systems Administrator IV        0.088001
Health Coach III                0.087474
Human Resources Assistant IV   0.076379
Web Developer II                 0.072734
Programmer Analyst I            0.071497
Geologist II                     0.065086
Geologist III                    0.064593
Automation Specialist IV       0.061198
Developer I                      0.060576
Office Assistant II              0.059511
Research Assistant III           0.055594
Administrative Assistant I      0.044153
Developer IV                      0.039761
Database Administrator II        0.037188
Name: profit_percentage, dtype: float64
```

```
In [128]: 1 Sprocket_df.groupby('job_industry_category').profit_percentage.sum().sort_values(ascending = False)
```

```
Out[128]: job_industry_category
Financial Services          24.409255
Manufacturing                  24.129211
Health                         18.996776
Retail                          10.499267
Property                        8.138326
Entertainment                   4.259811
IT                               4.242691
Agriculture                     3.342727
Telecommunications                1.981936
Name: profit_percentage, dtype: float64
```

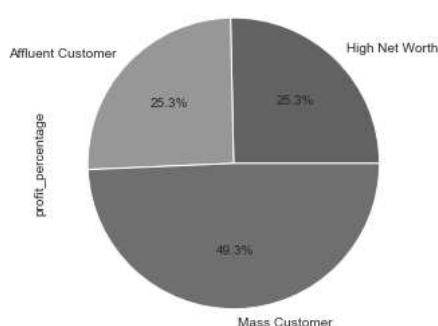
```
In [129]: 1 plt.figure(figsize=(10,10))
2 Sprocket_df.groupby('job_industry_category').profit_percentage.sum().sort_values().plot.pie(autopct="%1.1f%")
3 plt.show()
```



```
In [130]: 1 Sprocket_df.groupby('wealth_segment').profit_percentage.sum().sort_values()
```

```
Out[130]: wealth_segment
High Net Worth      25.340132
Affluent Customer  25.342887
Mass Customer       49.316981
Name: profit_percentage, dtype: float64
```

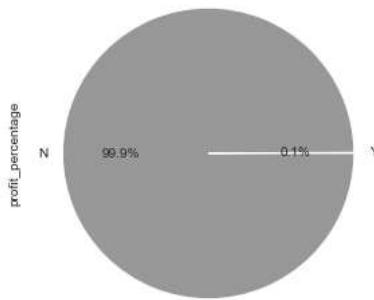
```
In [131]: 1 plt.figure(figsize=(5,5))
2 Sprocket_df.groupby('wealth_segment').profit_percentage.sum().sort_values().plot.pie(autopct="%1.1f%")
3 plt.show()
```



```
In [132]: 1 Sprocket_df.groupby('deceased_indicator').profit_percentage.sum().sort_values()
```

```
Out[132]: deceased_indicator
Y      0.107919
N     99.892081
Name: profit_percentage, dtype: float64
```

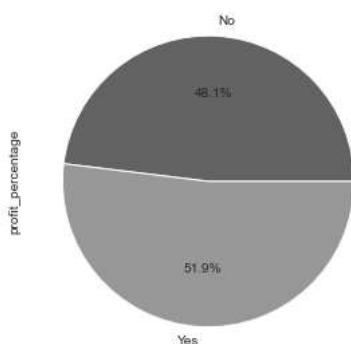
```
In [133]: 1 plt.figure(figsize=(5,5))
2 Sprocket_df.groupby('deceased_indicator').profit_percentage.sum().sort_values().plot.pie(autopct="%1.1f%%")
3 plt.show()
```



```
In [134]: 1 Sprocket_df.groupby('owns_car').profit_percentage.sum().sort_values()
```

```
Out[134]: owns_car
No      48.068036
Yes     51.931964
Name: profit_percentage, dtype: float64
```

```
In [135]: 1 plt.figure(figsize=(5,5))
2 Sprocket_df.groupby('owns_car').profit_percentage.sum().sort_values().plot.pie(autopct="%1.1f%%")
3 plt.show()
```



7. Act phase

Key findings

1. About 88% of profit is generated from customers within the age 21 to 60.
2. About 53% of profit is generated from customers residing in new south wales.

Recommendations

Customers residing in new south wales and within the age of 21 to 60 should be top target

```
In [136]: 1 Sprocket_df.to_csv("Sprocket_df")
```

```
1
```