KPMG VIRTUAL INTERNSHIP PROJECT

TASK: 1 - Data Quality Assessment

Assessment of data quality and completeness in preparation for analysis.

The client provided KPMG with 4 datasets:

1.Customer Demographic

2.Customer Addresses

3.Transactions data

4.NewCustomersList

In [1]:
```
# Importing the required libraries
import pandas as pd
```

Reading the data

In [2]:
```
data = pd.ExcelFile(r"C:\Users\91939\DATASCIENCE_ALL_COURSES\Projects\KPMG V
```

Reading each file separately

In [3]:
```
Transactions = pd.read_excel(data, 'Transactions')
NewCustomerList = pd.read_excel(data, 'NewCustomerList')
CustomerDemographic = pd.read_excel(data, 'CustomerDemographic')
CustomerAddress = pd.read_excel(data, 'CustomerAddress')
```

Exploring Transactions Data Set

In [4]:
```
1 Transactions.head(5)
```

Out[4]:

	transaction_id	product_id	customer_id	transaction_date	online_order	order_status	brand	p
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	

5 rows × 26 columns

In [5]:
```
1 Transactions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20000 entries, 0 to 19999
Data columns (total 26 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       
 13  Unnamed: 13              0 non-null      float64       
 14  Unnamed: 14              0 non-null      float64       
 15  Unnamed: 15              0 non-null      float64       
 16  Unnamed: 16              0 non-null      float64       
 17  Unnamed: 17              0 non-null      float64       
 18  Unnamed: 18              0 non-null      float64       
 19  Unnamed: 19              0 non-null      float64       
 20  Unnamed: 20              0 non-null      float64       
 21  Unnamed: 21              0 non-null      float64       
 22  Unnamed: 22              0 non-null      float64       
 23  Unnamed: 23              0 non-null      float64       
 24  Unnamed: 24              0 non-null      float64       
 25  Unnamed: 25              0 non-null      float64       
dtypes: datetime64[ns](1), float64(17), int64(3), object(5)
memory usage: 4.0+ MB
```

In [6]:
```
#Using only the required columns
Transactions = Transactions.iloc[:, 0:13]
Transactions.head()
```

Out[6]:

	transaction_id	product_id	customer_id	transaction_date	online_order	order_status	brand	p
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 [7]:
```
Transactions.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 [8]:
```
#Checking the shape of the data
Transactions.shape
```

Out[8]:
```
(20000, 13)
```

In [9]:
```
#Checking for null values
Transactions.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
```

There are missing values in 7 columns. They can be dropped or treated according to the nature of analysis

In [10]:
```
#Checking for duplicate values
Transactions.duplicated().sum()
```

Out[10]:
```
0
```

There are no duplicate values, so the data is unique.

In [11]:
```
#check for uniqueness of each column
Transactions.nunique()
```

Out[11]:
```
transaction_id             20000
product_id                   101
customer_id                 3494
transaction_date             364
online_order                   2
order_status                   2
brand                          6
product_line                   4
product_class                  3
product_size                   3
list_price                   296
standard_cost                103
product_first_sold_date      100
dtype: int64
```

Exploring the columns

In [12]:
```
1 Transactions.columns
```

Out[12]:
```
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 [13]:
```
1 Transactions['order_status'].value_counts()
```

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

In [14]:
```
1 Transactions['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]:
```
1 Transactions['product_line'].value_counts()
```

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

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

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

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

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

In [18]:
```
1 Transactions['product_first_sold_date']
```

Out[18]:
```
0        41245.0
1        41701.0
2        36361.0
3        36145.0
4        42226.0
          ...
19995    37823.0
19996    35560.0
19997    40410.0
19998    38216.0
19999    36334.0
Name: product_first_sold_date, Length: 20000, dtype: float64
```

In [19]:
```
1 #convert date column from integer to datetime
2 Transactions['product_first_sold_date'] = pd.to_datetime(Transactions['produ
3 Transactions['product_first_sold_date'].head()
```

Out[19]:
```
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 [20]:
```
1 Transactions['product_first_sold_date'].head(20)
```

Out[20]:
```
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
5    1970-01-01 10:50:31
6    1970-01-01 09:29:25
7    1970-01-01 11:05:15
8    1970-01-01 09:17:35
9    1970-01-01 10:36:56
10   1970-01-01 11:19:44
11   1970-01-01 11:42:52
12   1970-01-01 09:35:27
13   1970-01-01 09:36:26
14   1970-01-01 10:36:33
15   1970-01-01 10:31:13
16   1970-01-01 10:36:46
17   1970-01-01 09:24:48
18   1970-01-01 11:05:15
19   1970-01-01 10:22:17
Name: product_first_sold_date, dtype: datetime64[ns]
```

The values in the product_first_sold_date columns are not correct as it shows everything happening the same day at different times.

Exploring New Customer List Data Set

In [21]:
```
1 NewCustomerList.head(5)
```

Out[21]:

	first_name	last_name	gender	past_3_years_bike_related_purchases	DOB	job_title	job_i
0	Chickie	Brister	Male	86	1957-07-12	General Manager	
1	Morly	Genery	Male	69	1970-03-22	Structural Engineer	
2	Ardelis	Forrester	Female	10	1974-08-28	Senior Cost Accountant	
3	Lucine	Stutt	Female	64	1979-01-28	Account Representative III	
4	Melinda	Hadlee	Female	34	1965-09-21	Financial Analyst	

5 rows × 23 columns

In [22]:
```
1 NewCustomerList.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 23 columns):
 #   Column                               Non-Null Count  Dtype         
---  ------                               --------------  -----         
 0   first_name                           1000 non-null   object        
 1   last_name                            971 non-null    object        
 2   gender                               1000 non-null   object        
 3   past_3_years_bike_related_purchases  1000 non-null   int64         
 4   DOB                                  983 non-null    datetime64[ns]
 5   job_title                            894 non-null    object        
 6   job_industry_category                835 non-null    object        
 7   wealth_segment                       1000 non-null   object        
 8   deceased_indicator                   1000 non-null   object        
 9   owns_car                             1000 non-null   object        
 10  tenure                               1000 non-null   int64         
 11  address                              1000 non-null   object        
 12  postcode                             1000 non-null   int64         
 13  state                                1000 non-null   object        
 14  country                              1000 non-null   object        
 15  property_valuation                   1000 non-null   int64         
 16  Unnamed: 16                          1000 non-null   float64       
 17  Unnamed: 17                          1000 non-null   float64       
 18  Unnamed: 18                          1000 non-null   float64       
 19  Unnamed: 19                          1000 non-null   float64       
 20  Unnamed: 20                          1000 non-null   int64         
 21  Rank                                 1000 non-null   int64         
 22  Value                                1000 non-null   float64       
dtypes: datetime64[ns](1), float64(5), int64(6), object(11)
memory usage: 179.8+ KB
```

In [23]:
```
1 #Dropping the unnamed columns
2 NewCustomerList.drop(['Unnamed: 16', 'Unnamed: 17', 'Unnamed: 18',
3        'Unnamed: 19', 'Unnamed: 20'], axis=1, inplace=True)
```

In [24]:
```
1 #Checking the shape of the dataset
2 NewCustomerList.shape
```

Out[24]:
```
(1000, 18)
```

In [25]:
```
#Checking for null values
NewCustomerList.isnull().sum()
```

Out[25]:
```
first_name                             0
last_name                             29
gender                                 0
past_3_years_bike_related_purchases    0
DOB                                   17
job_title                            106
job_industry_category                165
wealth_segment                         0
deceased_indicator                     0
owns_car                               0
tenure                                 0
address                                0
postcode                               0
state                                  0
country                                0
property_valuation                     0
Rank                                   0
Value                                  0
dtype: int64
```

There are missing values in 4 columns. They can be dropped or treated according to the nature of analysis

In [26]:
```
#Checking for duplicate values
NewCustomerList.duplicated().sum()
```

Out[26]:
```
0
```

There are no duplicate values.

In [27]:
```
#Checking for uniquess of each column
NewCustomerList.nunique()
```

Out[27]:
```
first_name                             940
last_name                              961
gender                                   3
past_3_years_bike_related_purchases    100
DOB                                    958
job_title                              184
job_industry_category                    9
wealth_segment                           3
deceased_indicator                       1
owns_car                                 2
tenure                                  23
address                               1000
postcode                               522
state                                    3
country                                  1
property_valuation                      12
Rank                                   324
Value                                  324
dtype: int64
```

Exploring the columns

In [28]:
```
NewCustomerList.columns
```

Out[28]:
```
Index(['first_name', 'last_name', 'gender',
       'past_3_years_bike_related_purchases', 'DOB', 'job_title',
       'job_industry_category', 'wealth_segment', 'deceased_indicator',
       'owns_car', 'tenure', 'address', 'postcode', 'state', 'country',
       'property_valuation', 'Rank', 'Value'],
      dtype='object')
```

In [29]:
```
NewCustomerList['gender'].value_counts()
```

Out[29]:
```
Female    513
Male      470
U          17
Name: gender, dtype: int64
```

In [30]:
```
1 NewCustomerList[NewCustomerList.gender == "U"]
```

Out[30]:

	first_name	last_name	gender	past_3_years_bike_related_purchases	DOB	job_title	job_
59	Normy	Goodinge	U	5	NaT	Associate Professor	
226	Hatti	Carletti	U	35	NaT	Legal Assistant	
324	Rozamond	Turtle	U	69	NaT	Legal Assistant	
358	Tamas	Swatman	U	65	NaT	Assistant Media Planner	
360	Tracy	Andrejevic	U	71	NaT	Programmer II	
374	Agneta	McAmish	U	66	NaT	Structural Analysis Engineer	
434	Gregg	Aimeric	U	52	NaT	Internal Auditor	
439	Johna	Bunker	U	93	NaT	Tax Accountant	
574	Harlene	Nono	U	69	NaT	Human Resources Manager	
598	Gerianne	Kaysor	U	15	NaT	Project Manager	
664	Chicky	Sinclar	U	43	NaT	Operator	
751	Adriana	Saundercock	U	20	NaT	Nurse	
775	Dmitri	Viant	U	62	NaT	Paralegal	
835	Porty	Hansed	U	88	NaT	General Manager	
883	Shara	Bramhill	U	24	NaT	NaN	
904	Roth	Crum	U	0	NaT	Legal Assistant	
984	Pauline	Dallosso	U	82	NaT	Desktop Support Technician	

There are 17 columns with unknown/unspecified gender.

In [31]:
```
1 NewCustomerList['DOB'].value_counts()
```

Out[31]:
```
1993-11-02    2
1994-04-15    2
1963-08-25    2
1995-08-13    2
1987-01-15    2
             ..
1958-05-14    1
1977-12-08    1
1993-12-19    1
1954-10-06    1
1995-10-19    1
Name: DOB, Length: 958, dtype: int64
```

In [32]:
```
1 NewCustomerList['job_industry_category'].value_counts()
```

Out[32]:
```
Financial Services    203
Manufacturing         199
Health                152
Retail                 78
Property               64
IT                     51
Entertainment          37
Argiculture            26
Telecommunications     25
Name: job_industry_category, dtype: int64
```

In [33]:
```
1 NewCustomerList['wealth_segment'].value_counts()
```

Out[33]:
```
Mass Customer        508
High Net Worth       251
Affluent Customer    241
Name: wealth_segment, dtype: int64
```

In [34]:
```
1 NewCustomerList['state'].value_counts()
```

Out[34]:
```
NSW    506
VIC    266
QLD    228
Name: state, dtype: int64
```

In [35]:
```
1 NewCustomerList['owns_car'].value_counts()
```

Out[35]:
```
No     507
Yes    493
Name: owns_car, dtype: int64
```

In [36]:
```
1 NewCustomerList['deceased_indicator'].value_counts()
```

Out[36]:
```
N    1000
Name: deceased_indicator, dtype: int64
```

Exploring Customer Demographic Data Set

In [37]:
```
1 CustomerDemographic.head()
```

Out[37]:

	customer_id	first_name	last_name	gender	past_3_years_bike_related_purchases	DOB	jo
0	1	Laraine	Medendorp	F	93	1953-10-12	Ex Se
1	2	Eli	Bockman	Male	81	1980-12-16	Admini
2	3	Arlin	Dearle	Male	61	1954-01-20	Re M
3	4	Talbot	NaN	Male	33	1961-10-03	
4	5	Sheila-kathryn	Calton	Female	56	1977-05-13	Senio

In [38]:
```
1 CustomerDemographic.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 [39]:
```
#Checking for null values
CustomerDemographic.isnull().sum()
```

Out[39]:
```
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
```

There are missing values in 5 columns. They can be dropped or treated according to the nature of analysis

In [40]:
```
#Checking for duplicate data
CustomerDemographic.duplicated().sum()
```

Out[40]:
```
0
```

There are no duplicate values.

In [41]:
```
#Checking for uniqueness of each column
CustomerDemographic.nunique()
```

Out[41]:
```
customer_id                          4000
first_name                           3139
last_name                            3725
gender                                  6
past_3_years_bike_related_purchases   100
DOB                                  3448
job_title                             195
job_industry_category                   9
wealth_segment                          3
deceased_indicator                      2
default                                90
owns_car                                2
tenure                                 22
dtype: int64
```

Exploring the columns

In [42]:
```
1 CustomerDemographic.columns
```

Out[42]:
```
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 [43]:
```
1 CustomerDemographic['gender'].value_counts()
```

Out[43]:
```
Female    2037
Male      1872
U           88
Femal        1
M            1
F            1
Name: gender, dtype: int64
```

Certain categories are not correctly titled. The names in these categories are re-named.

In [44]:
```
1 #Re-naming the categories
2 CustomerDemographic['gender'] = CustomerDemographic['gender'].replace('F','F
```

In [45]:
```
1 CustomerDemographic['gender'].value_counts()
```

Out[45]:
```
Female         2039
Male           1873
Unspecified      88
Name: gender, dtype: int64
```

In [46]:
```
1 CustomerDemographic['past_3_years_bike_related_purchases'].value_counts()
```

Out[46]:
```
19    56
16    56
67    54
20    54
2     50
      ..
8     28
85    27
86    27
95    27
92    24
Name: past_3_years_bike_related_purchases, Length: 100, dtype: int64
```

In [47]:
```
1 CustomerDemographic['DOB'].value_counts()
```

Out[47]:
```
1978-01-30    7
1978-08-19    4
1964-07-08    4
1976-09-25    4
1976-07-16    4
             ..
2001-01-22    1
1955-03-06    1
1966-08-05    1
1968-11-16    1
1958-08-02    1
Name: DOB, Length: 3448, dtype: int64
```

In [48]:
```
1 CustomerDemographic['job_title'].value_counts()
```

Out[48]:
```
Business Systems Development Analyst    45
Social Worker                           44
Tax Accountant                          44
Internal Auditor                        42
Legal Assistant                         41
                                        ..
Account Representative II                4
Research Assistant III                   3
Health Coach I                           3
Health Coach III                         3
Developer I                              1
Name: job_title, Length: 195, dtype: int64
```

In [49]:
```
1 CustomerDemographic['job_industry_category'].value_counts()
```

Out[49]:
```
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
```

In [50]:
```
1 CustomerDemographic['wealth_segment'].value_counts()
```

Out[50]:
```
Mass Customer        2000
High Net Worth       1021
Affluent Customer     979
Name: wealth_segment, dtype: int64
```

In [51]:
```
1 CustomerDemographic['deceased_indicator'].value_counts()
```

Out[51]:
```
N    3998
Y       2
Name: deceased_indicator, dtype: int64
```

In [52]:
```
1 CustomerDemographic['default'].value_counts()
```

Out[52]:
```
100                                          113
1                                            112
-1                                           111
-100                                          99
Ù¡Ù¢Ù£                                        53
                                            ...
testâ testâ«                                  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
```

In [53]:
```
1 CustomerDemographic = CustomerDemographic.drop('default', axis=1)
```

The values are inconsistent, hence dropping the column.

In [54]:
```
1 CustomerDemographic.head(5)
```

Out[54]:

	customer_id	first_name	last_name	gender	past_3_years_bike_related_purchases	DOB	jo
0	1	Laraine	Medendorp	Female	93	1953-10-12	Ex Se
1	2	Eli	Bockman	Male	81	1980-12-16	Admini
2	3	Arlin	Dearle	Male	61	1954-01-20	Re M
3	4	Talbot	NaN	Male	33	1961-10-03	
4	5	Sheila-kathryn	Calton	Female	56	1977-05-13	Senio

In [55]:
```
1 CustomerDemographic['owns_car'].value_counts()
```

Out[55]:
```
Yes    2024
No     1976
Name: owns_car, dtype: int64
```

In [56]:
```
1 CustomerDemographic['tenure'].value_counts()
```

Out[56]:
```
7.0     235
5.0     228
11.0    221
10.0    218
16.0    215
8.0     211
18.0    208
12.0    202
14.0    200
9.0     200
6.0     192
4.0     191
13.0    191
17.0    182
15.0    179
1.0     166
3.0     160
19.0    159
2.0     150
20.0     96
22.0     55
21.0     54
Name: tenure, dtype: int64
```

Exploring Customer Address Data Set

In [57]:
```
1 CustomerAddress.head(5)
```

Out[57]:

	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 [58]:
```
1 CustomerAddress.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 [59]:
```
1 #Checking for null values.
2 CustomerAddress.isnull().sum()
```

Out[59]:
```
customer_id           0
address               0
postcode              0
state                 0
country               0
property_valuation    0
dtype: int64
```

There are no null values.

In [60]:
```
1 #Checking for duplicate values
2 CustomerAddress.duplicated().sum()
```

Out[60]:
```
0
```

There are no duplicate values.

In [61]:
```
1 #Checking for uniqueness of each column
2 CustomerAddress.nunique()
```

Out[61]:
```
customer_id           3999
address               3996
postcode               873
state                    5
country                  1
property_valuation      12
dtype: int64
```

Exploring the columns

In [62]:
```
1 CustomerAddress['postcode'].value_counts()
```

Out[62]:
```
2170    31
2145    30
2155    30
2153    29
3977    26
        ..
3331     1
3036     1
3321     1
3305     1
2143     1
Name: postcode, Length: 873, dtype: int64
```

In [63]:
```
1 CustomerAddress['state'].value_counts()
```

Out[63]:
```
NSW                2054
VIC                 939
QLD                 838
New South Wales      86
Victoria             82
Name: state, dtype: int64
```

In [64]:
```
1 CustomerAddress['country'].value_counts()
```

Out[64]:
```
Australia    3999
Name: country, dtype: int64
```

In [65]:
```
1 CustomerAddress['property_valuation'].value_counts()
```

Out[65]:
```
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
```

All the columns appear to have consistent and correct information.

Data Analysis of New Customer List

Extracting the age of Customers

In [66]:
```
today = pd.to_datetime('today')
NewCustomerList['Age'] = today.year - NewCustomerList['DOB'].dt.year
```

In [67]:
```
NewCustomerList['Age']
```

Out[67]:
```
0      65.0
1      52.0
2      48.0
3      43.0
4      57.0
       ...
995    63.0
996    21.0
997    68.0
998    70.0
999    67.0
Name: Age, Length: 1000, dtype: float64
```

In [68]:
```
NewCustomerList.dtypes
```

Out[68]:
```
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
owns_car                               object
tenure                                  int64
address                                object
postcode                                int64
state                                  object
country                                object
property_valuation                      int64
Rank                                    int64
Value                                 float64
Age                                   float64
dtype: object
```

In [69]:
```
1 NewCustomerList['Age']
```

Out[69]:
```
0      65.0
1      52.0
2      48.0
3      43.0
4      57.0
       ...
995    63.0
996    21.0
997    68.0
998    70.0
999    67.0
Name: Age, Length: 1000, dtype: float64
```

Categorisation of age into different sections

In [70]:
```
1  def Age_Category(age):
2      if age>10 and age<20:
3          return '10-19'
4      elif age>=20 and age<30:
5          return '20-29'
6      elif age>=30and age<40:
7          return '30-39'
8      elif age>=40 and age<50:
9          return '40-49'
10     elif age>=50 and age<60:
11         return '50-59'
12     elif age>=60 and age<70:
13         return '60-69'
14     elif age>=70:
15         return 'Above 70'
16
17
18 NewCustomerList['Age_Category'] = NewCustomerList['Age'].apply(Age_Category)
```

In [71]:
```
1 NewCustomerList
```

Out[71]:

	first_name	last_name	gender	past_3_years_bike_related_purchases	DOB	job_title
0	Chickie	Brister	Male	86	1957-07-12	General Manager
1	Morly	Genery	Male	69	1970-03-22	Structural Engineer
2	Ardelis	Forrester	Female	10	1974-08-28	Senior Cost Accountant
3	Lucine	Stutt	Female	64	1979-01-28	Account Representative III
4	Melinda	Hadlee	Female	34	1965-09-21	Financial Analyst
...	...	...	...	...	...	...
995	Ferdinand	Romanetti	Male	60	1959-10-07	Paralegal
996	Burk	Wortley	Male	22	2001-10-17	Senior Sales Associate
997	Melloney	Temby	Female	17	1954-10-05	Budget/Accounting Analyst IV
998	Dickie	Cubbini	Male	30	1952-12-17	Financial Advisor
999	Sylas	Duffill	Male	56	1955-10-02	Staff Accountant IV

1000 rows × 20 columns

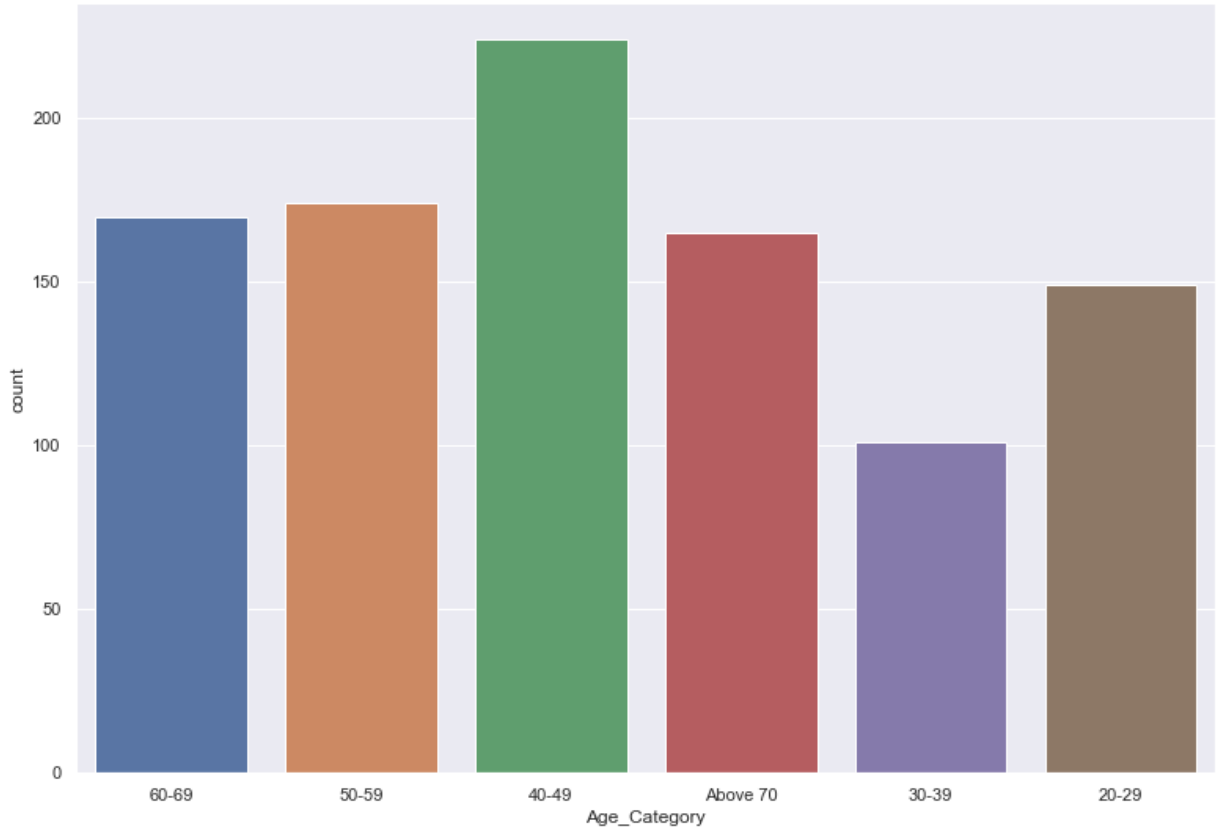
In [72]:
```
1 NewCustomerList['Age_Category']
```

Out[72]:
```
0         60-69
1         50-59
2         40-49
3         40-49
4         50-59
         ...
995       60-69
996       20-29
997       60-69
998    Above 70
999       60-69
Name: Age_Category, Length: 1000, dtype: object
```

Visualisation of Customers Basesd on Age Category

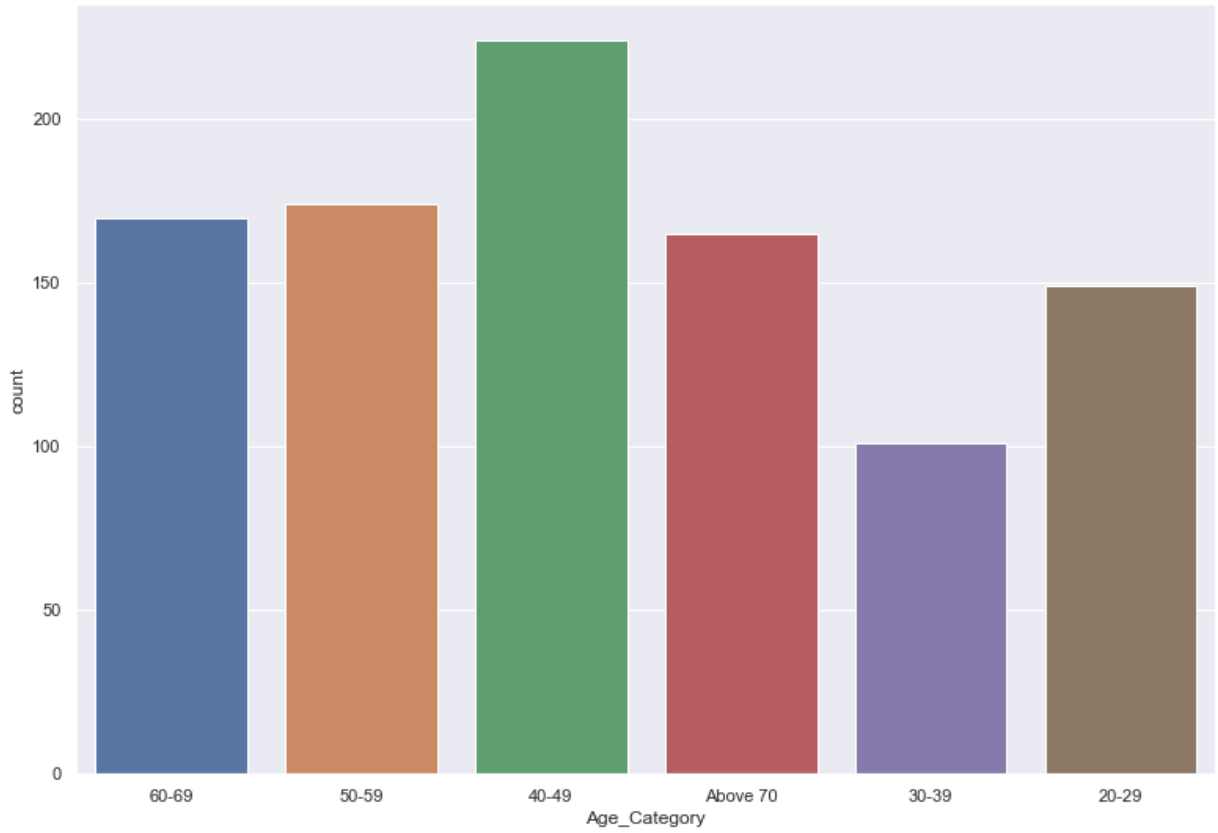
In [73]:
```
1 import seaborn as sns
2 import matplotlib.pyplot as plt
3 %matplotlib inline
4 sns.set(rc={'figure.figsize':(13,9)})
5 sns.countplot(x = 'Age_Category',data = NewCustomerList)
```

Out[73]:
```
<AxesSubplot:xlabel='Age_Category', ylabel='count'>
```



Saving the Image

In [74]:
```
Img1 = sns.countplot(x = 'Age_Category',data = NewCustomerList)
fig = Img1.get_figure()
fig.savefig("Img1.png")
```



Exploring the Number of sales on bike_related_purchases for past 3 years based on Gender wrt to Age Category

In [75]:
```
df = NewCustomerList[['Age_Category','past_3_years_bike_related_purchases','
```

In [76]:
```
1 df
```

Out[76]:

	Age_Category	past_3_years_bike_related_purchases	gender
0	60-69	86	Male
1	50-59	69	Male
2	40-49	10	Female
3	40-49	64	Female
4	50-59	34	Female
...	...	...	...
995	60-69	60	Male
996	20-29	22	Male
997	60-69	17	Female
998	Above 70	30	Male
999	60-69	56	Male

1000 rows × 3 columns

In [77]:
```
1 df1 = df.groupby(['Age_Category','gender']).sum()
```

In [78]:
```
1 df1
```

Out[78]:

		past_3_years_bike_related_purchases
Age_Category	**gender**	
20-29	**Female**	3184
	Male	4221
30-39	**Female**	2729
	Male	2590
40-49	**Female**	6019
	Male	4874
50-59	**Female**	4834
	Male	4283
60-69	**Female**	4253
	Male	4068
Above 70	**Female**	4193
	Male	3729

In [79]:
```
1 df2 = df1.reset_index()
```

In [80]:
```
df2
```

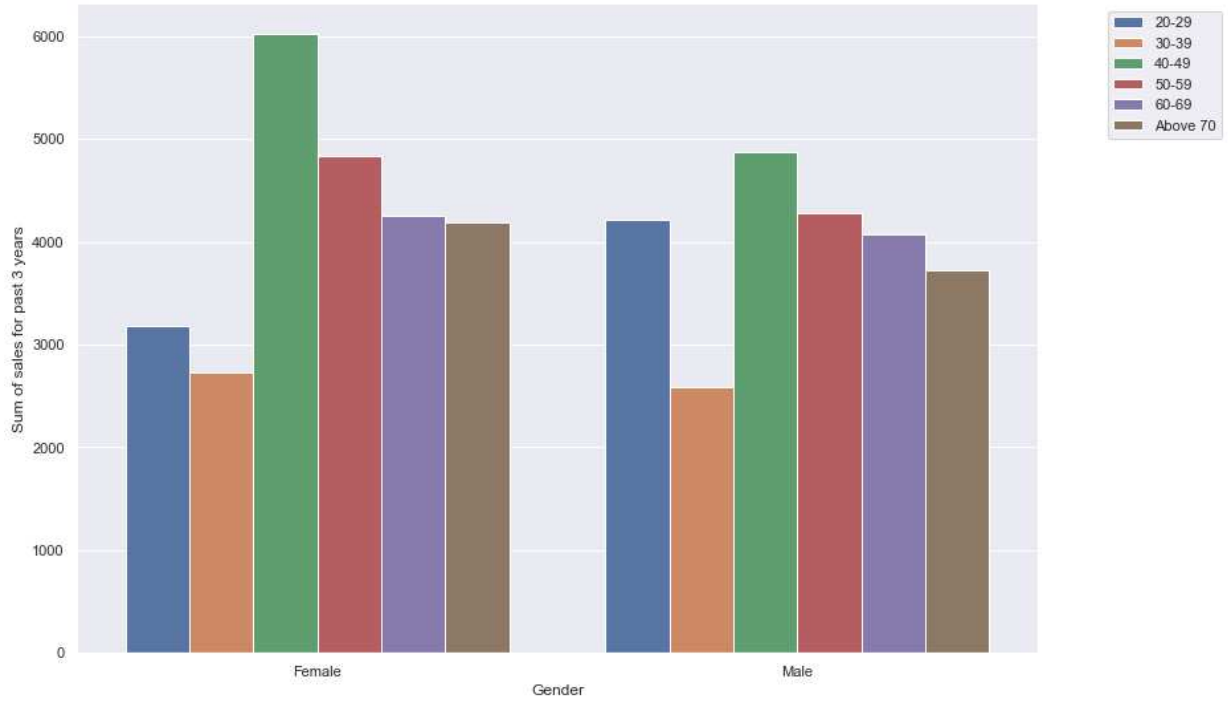
Out[80]:

	Age_Category	gender	past_3_years_bike_related_purchases
0	20-29	Female	3184
1	20-29	Male	4221
2	30-39	Female	2729
3	30-39	Male	2590
4	40-49	Female	6019
5	40-49	Male	4874
6	50-59	Female	4834
7	50-59	Male	4283
8	60-69	Female	4253
9	60-69	Male	4068
10	Above 70	Female	4193
11	Above 70	Male	3729

Visualisation of dataset

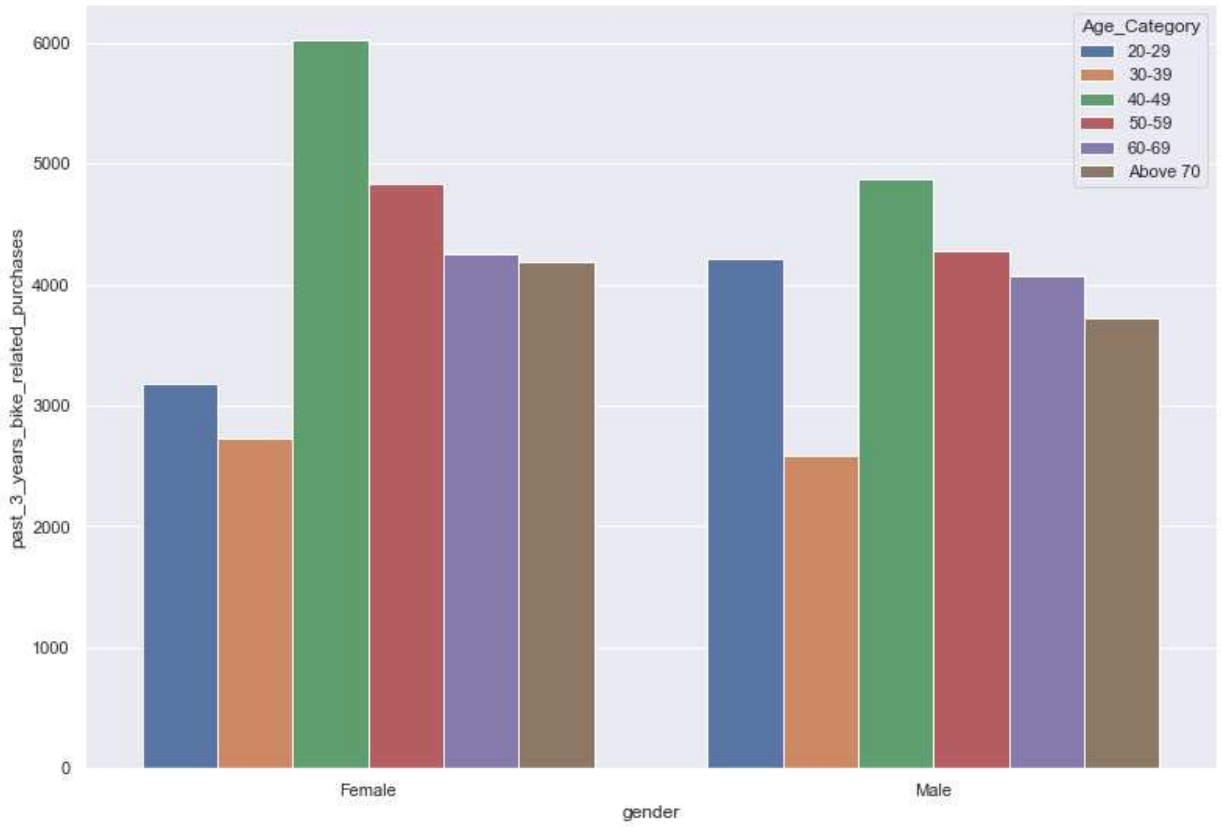
In [81]:
```
sns.set(rc={'figure.figsize':(13,9)})
fig = sns.barplot(x = 'gender',y = 'past_3_years_bike_related_purchases',dat
fig.set(xlabel = 'Gender',ylabel= 'Sum of sales for past 3 years')
fig.legend(bbox_to_anchor= (1.2,1))
```

Out[81]: `<matplotlib.legend.Legend at 0x1e1c392a2b0>`



Saving the Image

In [82]:
```
Img2 = sns.barplot(x = 'gender',y = 'past_3_years_bike_related_purchases',da
fig = Img2.get_figure()
fig.savefig("Img2.png")
```



In [83]:
```
1 NewCustomerList
```

Out[83]:

	first_name	last_name	gender	past_3_years_bike_related_purchases	DOB	job_title
0	Chickie	Brister	Male	86	1957-07-12	General Manager
1	Morly	Genery	Male	69	1970-03-22	Structural Engineer
2	Ardelis	Forrester	Female	10	1974-08-28	Senior Cost Accountant
3	Lucine	Stutt	Female	64	1979-01-28	Account Representative III
4	Melinda	Hadlee	Female	34	1965-09-21	Financial Analyst
...	...	...	...	...	...	...
995	Ferdinand	Romanetti	Male	60	1959-10-07	Paralegal
996	Burk	Wortley	Male	22	2001-10-17	Senior Sales Associate
997	Melloney	Temby	Female	17	1954-10-05	Budget/Accounting Analyst IV
998	Dickie	Cubbini	Male	30	1952-12-17	Financial Advisor
999	Sylas	Duffill	Male	56	1955-10-02	Staff Accountant IV

1000 rows × 20 columns

Exploring the data based on the job_industry_category wrt past_3_years_bike_purchases

In [84]:
```
1 df3  = NewCustomerList[['job_industry_category','past_3_years_bike_related_p
```

In [85]:
```
1 df3
```

Out[85]:

	job_industry_category	past_3_years_bike_related_purchases
0	Manufacturing	86
1	Property	69
2	Financial Services	10
3	Manufacturing	64
4	Financial Services	34
...	...	...
995	Financial Services	60
996	Health	22
997	Financial Services	17
998	Financial Services	30
999	Property	56

1000 rows × 2 columns

In [86]:
```
1 df4 = df3.groupby('job_industry_category').sum()
```

In [87]:
```
1 df5 = df4.reset_index()
```

In [88]:
```
1 df5
```

Out[88]:

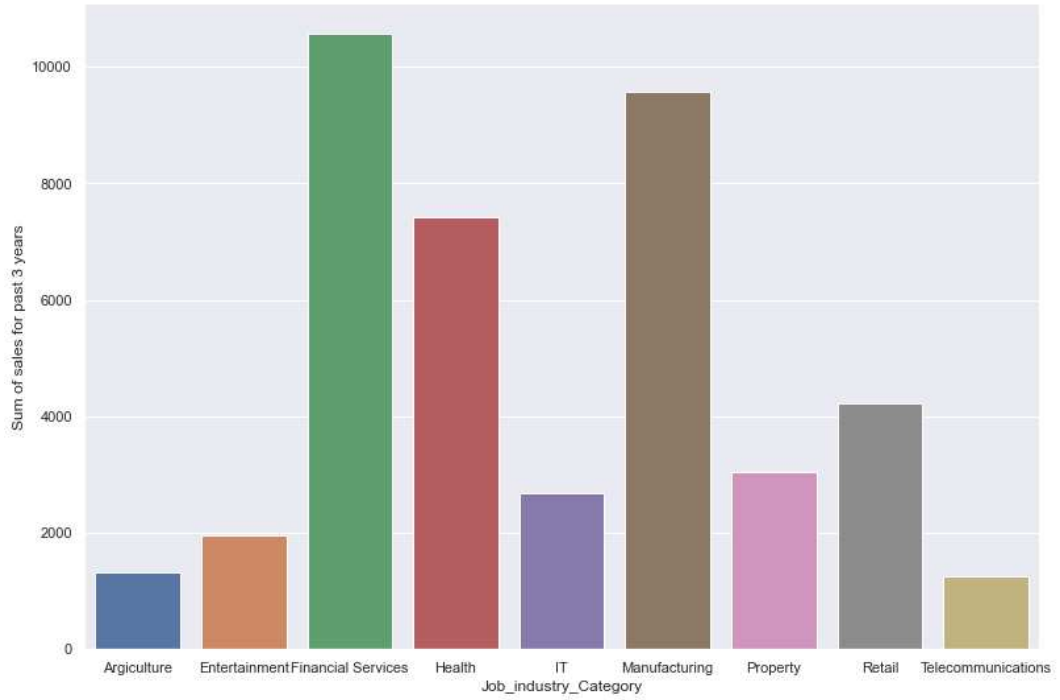
	job_industry_category	past_3_years_bike_related_purchases
0	Argiculture	1323
1	Entertainment	1953
2	Financial Services	10564
3	Health	7421
4	IT	2688
5	Manufacturing	9562
6	Property	3033
7	Retail	4225
8	Telecommunications	1250

Visualisation of data

In [89]:
```
sns.set(rc={'figure.figsize':(13,9)})
fig = sns.barplot(x = 'job_industry_category',y = 'past_3_years_bike_related
fig.set(xlabel = 'Job_industry_Category',ylabel= 'Sum of sales for past 3 ye
fig.legend(bbox_to_anchor= (1.2,1))
```

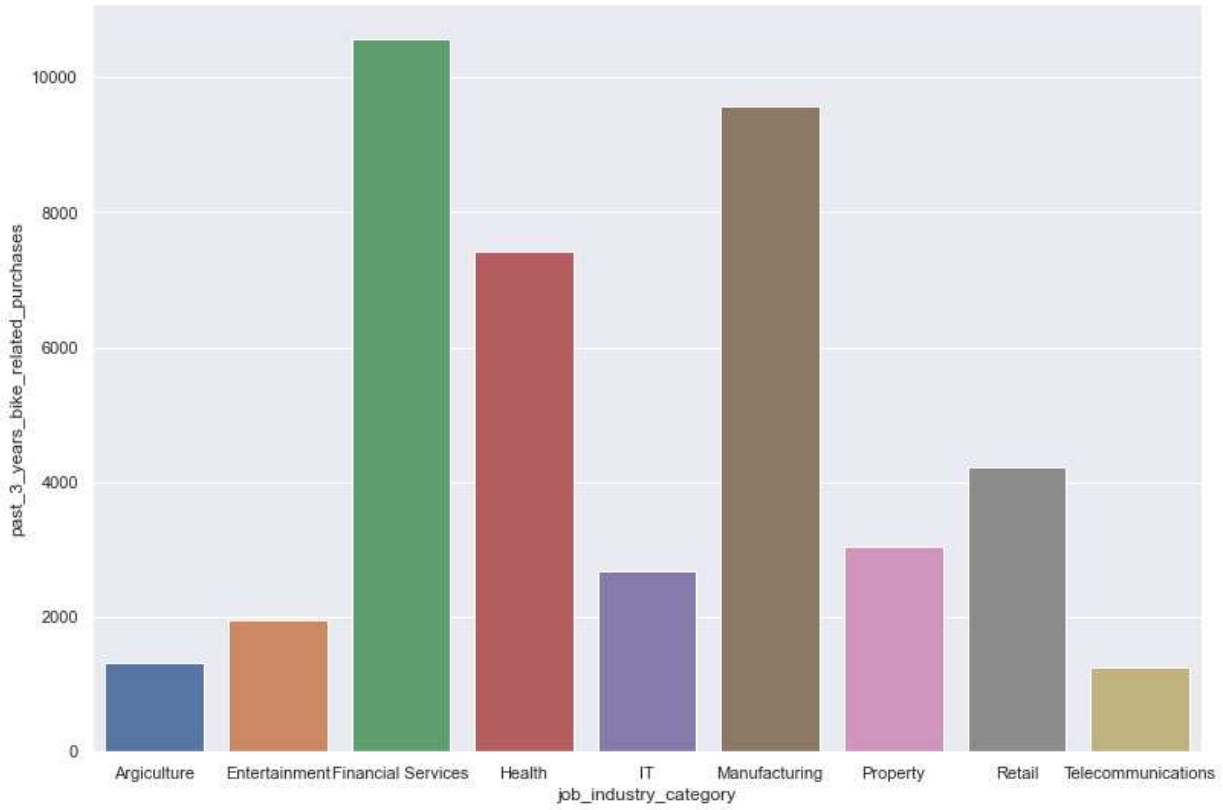
```
No handles with labels found to put in legend.
```

Out[89]:
```
<matplotlib.legend.Legend at 0x1e1c335adf0>
```



Saving the image

In [90]:
```
Img3 = sns.barplot(x = 'job_industry_category',y = 'past_3_years_bike_relate
fig = Img3.get_figure()
fig.savefig("Img3.png")
```



Exploring the data based on state living and who owns_car or not

In [91]:
```
df6 = NewCustomerList[['state','owns_car']]
```

In [92]:
```
def car(x):
    if x == 'Yes':
        return 1
    else:
        return 0;
```

In [93]:
```
df6['status'] = df6['owns_car'].apply(car)
```

```
<ipython-input-93-b127357906f3>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df6['status'] = df6['owns_car'].apply(car)
```

In [94]:
```
df6
```

Out[94]:

	state	owns_car	status
0	QLD	Yes	1
1	NSW	No	0
2	VIC	No	0
3	QLD	Yes	1
4	NSW	No	0
...	...	...	...
995	NSW	No	0
996	NSW	No	0
997	QLD	Yes	1
998	QLD	Yes	1
999	NSW	Yes	1

1000 rows × 3 columns

In [95]:
```
df7 = df6.groupby(['state','owns_car']).count().reset_index()
```

In [96]:
```
1 df7
```

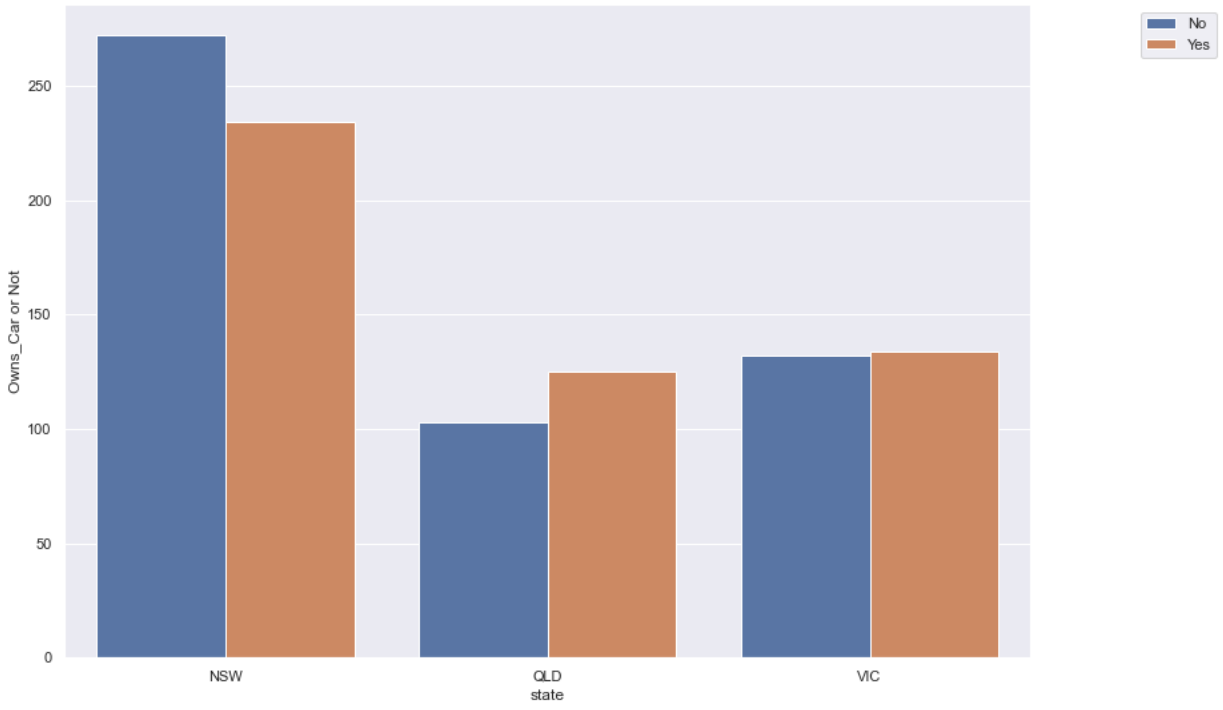
Out[96]:

	state	owns_car	status
0	NSW	No	272
1	NSW	Yes	234
2	QLD	No	103
3	QLD	Yes	125
4	VIC	No	132
5	VIC	Yes	134

Visualisation of data

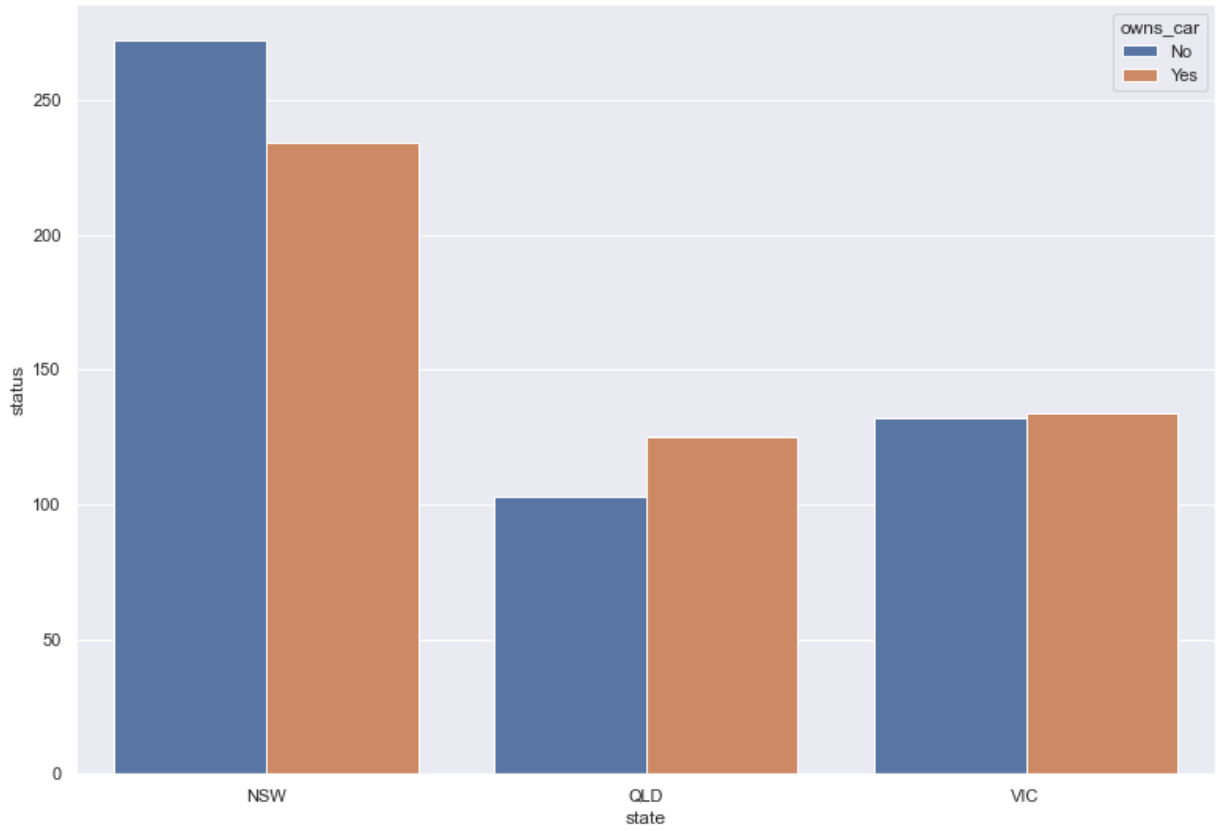
In [97]:
```
1 sns.set(rc={'figure.figsize':(13,9)})
2 fig = sns.barplot(x = 'state',y = 'status',hue = 'owns_car',data = df7)
3 fig.set(xlabel = 'state',ylabel= 'Owns_Car or Not')
4 fig.legend(bbox_to_anchor= (1.2,1))
```

Out[97]: `<matplotlib.legend.Legend at 0x1e1c399e6d0>`



Saving the Image

In [98]:
```
Img4 = sns.barplot(x = 'state',y = 'status',hue = 'owns_car',data = df7)
fig = Img4.get_figure()
fig.savefig("Img4.png")
```



In [120]:
```
1 NewCustomerList
```

Out[120]:

deceased_indicator	owns_car	tenure	address	postcode	state	country	property_valuation	Rar
N	Yes	14	45 Shopko Center	4500	QLD	Australia	6	
N	No	16	14 Mccormick Park	2113	NSW	Australia	11	
N	No	10	5 Colorado Crossing	3505	VIC	Australia	5	
N	Yes	5	207 Annamark Plaza	4814	QLD	Australia	1	
N	No	19	115 Montana Place	2093	NSW	Australia	9	
...	...	...	...	...	...	...	...	
N	No	9	2 Sloan Way	2200	NSW	Australia	7	99
N	No	6	04 Union Crossing	2196	NSW	Australia	10	99
N	Yes	15	33475 Fair Oaks Junction	4702	QLD	Australia	2	99
N	Yes	19	57666 Victoria Way	4215	QLD	Australia	2	99
N	Yes	14	21875 Grover Drive	2010	NSW	Australia	9	100

In [130]:
```
1 High_Value_Customers  = NewCustomerList[(NewCustomerList['job_industry_categ
```

In [131]:
```
1 High_Value_Customers
```

Out[131]:

	first_name	last_name	gender	job_industry_category	state	owns_car
82	Esther	Rooson	Female	Financial Services	NSW	No
166	Elvira	Kurten	Female	Financial Services	NSW	No
250	Sunny	Christescu	Female	Financial Services	NSW	No
542	Elvira	Darthe	Female	Financial Services	NSW	No
960	Sonia	Dunstall	Female	Financial Services	NSW	No

In [128]:
```
1 !pip install dataframe-image
```

```
Collecting dataframe-image
  Downloading dataframe_image-0.1.1-py3-none-any.whl (32 kB)
Collecting aiohttp
  Downloading aiohttp-3.8.1-cp38-cp38-win_amd64.whl (555 kB)
Requirement already satisfied: matplotlib>=3.1 in c:\users\91939\anaconda3\lib
\site-packages (from dataframe-image) (3.3.4)
Requirement already satisfied: nbconvert>=5 in c:\users\91939\anaconda3\lib\sit
e-packages (from dataframe-image) (6.0.7)
Requirement already satisfied: pandas>=0.24 in c:\users\91939\anaconda3\lib\sit
e-packages (from dataframe-image) (1.2.4)
Requirement already satisfied: requests in c:\users\91939\anaconda3\lib\site-pa
ckages (from dataframe-image) (2.27.1)
Requirement already satisfied: beautifulsoup4 in c:\users\91939\anaconda3\lib\s
ite-packages (from dataframe-image) (4.9.3)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\91939\anaconda3\li
b\site-packages (from matplotlib>=3.1->dataframe-image) (1.3.1)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\91939\anaconda3
\lib\site-packages (from matplotlib>=3.1->dataframe-image) (2.8.2)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:\u
sers\91939\anaconda3\lib\site-packages (from matplotlib>=3.1->dataframe-image)
(2.4.7)
Requirement already satisfied: numpy>=1.15 in c:\users\91939\anaconda3\lib\site
-packages (from matplotlib>=3.1->dataframe-image) (1.20.1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\91939\anaconda3\lib\si
te-packages (from matplotlib>=3.1->dataframe-image) (8.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\91939\anaconda3\lib\sit
e-packages (from matplotlib>=3.1->dataframe-image) (0.10.0)
Requirement already satisfied: six in c:\users\91939\anaconda3\lib\site-package
s (from cycler>=0.10->matplotlib>=3.1->dataframe-image) (1.16.0)
Requirement already satisfied: jupyterlab-pygments in c:\users\91939\anaconda3
\lib\site-packages (from nbconvert>=5->dataframe-image) (0.1.2)
Requirement already satisfied: bleach in c:\users\91939\anaconda3\lib\site-pack
ages (from nbconvert>=5->dataframe-image) (3.3.0)
Requirement already satisfied: jinja2>=2.4 in c:\users\91939\anaconda3\lib\site
-packages (from nbconvert>=5->dataframe-image) (3.0.3)
Requirement already satisfied: pygments>=2.4.1 in c:\users\91939\anaconda3\lib
\site-packages (from nbconvert>=5->dataframe-image) (2.8.1)
Requirement already satisfied: nbformat>=4.4 in c:\users\91939\anaconda3\lib\si
te-packages (from nbconvert>=5->dataframe-image) (5.1.3)
Requirement already satisfied: entrypoints>=0.2.2 in c:\users\91939\anaconda3\l
ib\site-packages (from nbconvert>=5->dataframe-image) (0.3)
Requirement already satisfied: defusedxml in c:\users\91939\anaconda3\lib\site-
packages (from nbconvert>=5->dataframe-image) (0.7.1)
Requirement already satisfied: nbclient<0.6.0,>=0.5.0 in c:\users\91939\anacond
a3\lib\site-packages (from nbconvert>=5->dataframe-image) (0.5.3)
Requirement already satisfied: traitlets>=4.2 in c:\users\91939\anaconda3\lib\s
ite-packages (from nbconvert>=5->dataframe-image) (5.0.5)
Requirement already satisfied: pandocfilters>=1.4.1 in c:\users\91939\anaconda3
\lib\site-packages (from nbconvert>=5->dataframe-image) (1.4.3)
Requirement already satisfied: mistune<2,>=0.8.1 in c:\users\91939\anaconda3\li
b\site-packages (from nbconvert>=5->dataframe-image) (0.8.4)
Requirement already satisfied: testpath in c:\users\91939\anaconda3\lib\site-pa
ckages (from nbconvert>=5->dataframe-image) (0.4.4)
Requirement already satisfied: jupyter-core in c:\users\91939\anaconda3\lib\sit
```

```
e-packages (from nbconvert>=5->dataframe-image) (4.7.1)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\91939\anaconda3\lib
\site-packages (from jinja2>=2.4->nbconvert>=5->dataframe-image) (2.0.1)
Requirement already satisfied: nest-asyncio in c:\users\91939\anaconda3\lib\sit
e-packages (from nbclient<0.6.0,>=0.5.0->nbconvert>=5->dataframe-image) (1.5.1)
Requirement already satisfied: async-generator in c:\users\91939\anaconda3\lib
\site-packages (from nbclient<0.6.0,>=0.5.0->nbconvert>=5->dataframe-image) (1.
10)
Requirement already satisfied: jupyter-client>=6.1.5 in c:\users\91939\anaconda
3\lib\site-packages (from nbclient<0.6.0,>=0.5.0->nbconvert>=5->dataframe-imag
e) (6.1.12)
Requirement already satisfied: pyzmq>=13 in c:\users\91939\anaconda3\lib\site-p
ackages (from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert>=5->data
frame-image) (20.0.0)
Requirement already satisfied: tornado>=4.1 in c:\users\91939\anaconda3\lib\sit
e-packages (from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert>=5->d
ataframe-image) (6.1)
Requirement already satisfied: pywin32>=1.0 in c:\users\91939\anaconda3\lib\sit
e-packages (from jupyter-core->nbconvert>=5->dataframe-image) (227)
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in c:\users\91939\anacon
da3\lib\site-packages (from nbformat>=4.4->nbconvert>=5->dataframe-image) (3.2.
0)
Requirement already satisfied: ipython-genutils in c:\users\91939\anaconda3\lib
\site-packages (from nbformat>=4.4->nbconvert>=5->dataframe-image) (0.2.0)
Requirement already satisfied: pyrsistent>=0.14.0 in c:\users\91939\anaconda3\l
ib\site-packages (from jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert>=5->da
taframe-image) (0.17.3)
Requirement already satisfied: setuptools in c:\users\91939\anaconda3\lib\site-
packages (from jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert>=5->dataframe-
image) (52.0.0.post20210125)
Requirement already satisfied: attrs>=17.4.0 in c:\users\91939\anaconda3\lib\si
te-packages (from jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert>=5->datafra
me-image) (20.3.0)
Requirement already satisfied: pytz>=2017.3 in c:\users\91939\anaconda3\lib\sit
e-packages (from pandas>=0.24->dataframe-image) (2021.1)
Collecting multidict<7.0,>=4.5
  Downloading multidict-6.0.2-cp38-cp38-win_amd64.whl (28 kB)
Collecting yarl<2.0,>=1.0
  Downloading yarl-1.7.2-cp38-cp38-win_amd64.whl (122 kB)
Collecting async-timeout<5.0,>=4.0.0a3
  Downloading async_timeout-4.0.2-py3-none-any.whl (5.8 kB)
Collecting aiosignal>=1.1.2
  Downloading aiosignal-1.2.0-py3-none-any.whl (8.2 kB)
Requirement already satisfied: charset-normalizer<3.0,>=2.0 in c:\users\91939\a
naconda3\lib\site-packages (from aiohttp->dataframe-image) (2.0.12)
Collecting frozenlist>=1.1.1
  Downloading frozenlist-1.3.0-cp38-cp38-win_amd64.whl (33 kB)
Requirement already satisfied: idna>=2.0 in c:\users\91939\anaconda3\lib\site-p
ackages (from yarl<2.0,>=1.0->aiohttp->dataframe-image) (2.6)
Requirement already satisfied: soupsieve>1.2 in c:\users\91939\anaconda3\lib\si
te-packages (from beautifulsoup4->dataframe-image) (2.2.1)
Requirement already satisfied: packaging in c:\users\91939\anaconda3\lib\site-p
ackages (from bleach->nbconvert>=5->dataframe-image) (20.9)
Requirement already satisfied: webencodings in c:\users\91939\anaconda3\lib\sit
e-packages (from bleach->nbconvert>=5->dataframe-image) (0.5.1)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\91939\anaconda3\l
ib\site-packages (from requests->dataframe-image) (2020.12.5)
```

```
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\91939\anaconda
3\lib\site-packages (from requests->dataframe-image) (1.22)
Installing collected packages: multidict, frozenlist, yarl, async-timeout, aios
ignal, aiohttp, dataframe-image
Successfully installed aiohttp-3.8.1 aiosignal-1.2.0 async-timeout-4.0.2 datafr
ame-image-0.1.1 frozenlist-1.3.0 multidict-6.0.2 yarl-1.7.2
```

In [129]:
```
1 import dataframe_image as dfi
```

In [132]:
```
1 dfi.export(High_Value_Customers, 'High_Value-Customers.png')
```