FEYNN LABS_ PROJECT - 1_EXPLORATORY DATA ANALYSIS

Created by : Parth shukla

Now here I am given with project 1 under **feynn labs Machine Learning Internship**.

In this perticular project I have to come up with a business idea where I will apply Machine Learning/Data Science in small or medium business and help them with their sales, business operations, marketing etc.

So as a part of my this project I have found one sales dataset of one small shop on **Kaggle** and I will be using Machine Learning or Data Science techniques to help small buissnesses grow using this freely available dataset.

Let's Start

In the first step here we will be downloading the **dataset (CSV Format)** in our local computer and transferring that into desired file to load it here using **Pandas** library.

➔ Getting touch with our data

1. Importing Numpy and Pandas

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

2. Defining our dataset "df" , and loading our csv file into that.

In [2]:
```python
df = pd.read_csv('Data/201904 sales reciepts.csv')
```

3. Exploring our dataset first time.

Having first look of our dataset using **df.head()**.

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

Out[3]:

	transaction_id	transaction_date	transaction_time	sales_outlet_id	staff_id	customer_id	instore_yn	order	line_item_id	product_id	quantity	line
0	7	2019-04-01	12:04:43	3	12	558	N	1	1	52	1	
1	11	2019-04-01	15:54:39	3	17	781	N	1	1	27	2	
2	19	2019-04-01	14:34:59	3	17	788	Y	1	1	46	2	
3	32	2019-04-01	16:06:04	3	12	683	N	1	1	23	2	
4	33	2019-04-01	19:18:37	3	17	99	Y	1	1	34	1	

Checking for datatypes of all indivisual columns of our dataset using **df.info()**.

```
In [4]:  df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49894 entries, 0 to 49893
Data columns (total 14 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   transaction_id      49894  non-null int64
 1   transaction_date    49894  non-null object
 2   transaction_time    49894  non-null object
 3   sales_outlet_id     49894  non-null int64
 4   staff_id            49894  non-null int64
 5   customer_id         49894  non-null int64
 6   instore_yn          49894  non-null object
 7   order               49894  non-null int64
 8   line_item_id        49894  non-null int64
 9   product_id          49894  non-null int64
 10  quantity            49894  non-null int64
 11  line_item_amount    49894  non-null float64
 12  unit_price          49894  non-null float64
 13  promo_item_yn       49894  non-null object
dtypes: float64(2), int64(8), object(4)
memory usage: 5.3+ MB
```

checking for some mathematical relations and behaviours of our dataset using **df.describe()**.

In [5]:
```
df.describe()
```

Out[5]:

	transaction_id	sales_outlet_id	staff_id	customer_id	order	line_item_id	product_id	quantity	line_item_amount	un
count	49894.000000	49894.000000	49894.000000	49894.000000	49894.000000	49894.000000	49894.000000	49894.000000	49894.000000	49894
mean	869.056059	5.351846	25.359582	2282.324468	1.173428	1.631860	47.878983	1.438209	4.682646	3
std	857.863149	2.074796	12.466490	3240.551757	1.025445	1.412881	17.928355	0.543039	4.436668	2
min	1.000000	3.000000	6.000000	0.000000	1.000000	1.000000	1.000000	1.000000	0.000000	0
25%	223.000000	3.000000	15.000000	0.000000	1.000000	1.000000	33.000000	1.000000	3.000000	2
50%	481.000000	5.000000	26.000000	0.000000	1.000000	1.000000	47.000000	1.000000	3.750000	3
75%	1401.000000	8.000000	41.000000	5412.000000	1.000000	1.000000	60.000000	2.000000	6.000000	3
max	4203.000000	8.000000	45.000000	8501.000000	9.000000	12.000000	87.000000	8.000000	360.000000	45

5. Checking for corelations in our dataset.

Going ahead , using df.corr() to get the correlations of every column with all other columns in our dataset.

In [6]:
```python
df.corr()
```

Out[6]:

	transaction_id	sales_outlet_id	staff_id	customer_id	order	line_item_id	product_id	quantity	line_item_amount	unit_price
transaction_id	1.000000	-0.134200	-0.050462	0.004820	-0.052610	-0.047631	-0.046251	0.015083	-0.010319	-0.033934
sales_outlet_id	-0.134200	1.000000	0.696921	0.429706	0.012392	0.004210	0.024360	-0.002860	0.004255	-0.001673
staff_id	-0.050462	0.696921	1.000000	0.294914	0.015983	-0.008372	0.010359	0.002996	0.003410	-0.000396
customer_id	0.004820	0.429706	0.294914	1.000000	-0.018909	-0.008114	0.001156	0.011265	-0.005202	-0.016218
order	-0.052610	0.012392	0.015983	-0.018909	1.000000	0.000616	-0.173570	-0.125321	0.452822	0.758723
line_item_id	-0.047631	0.004210	-0.008372	-0.008114	0.000616	1.000000	0.604757	-0.315383	-0.050380	0.074058
product_id	-0.046251	0.024360	0.010359	0.001156	-0.173570	0.604757	1.000000	-0.175536	-0.164309	-0.138539
quantity	0.015083	-0.002860	0.002996	0.011265	-0.125321	-0.315383	-0.175536	1.000000	0.353336	-0.119205
line_item_amount	-0.010319	0.004255	0.003410	-0.005202	0.452822	-0.050380	-0.164309	0.353336	1.000000	0.672168
unit_price	-0.033934	-0.001673	-0.000396	-0.016218	0.758723	0.074058	-0.138539	-0.119205	0.672168	1.000000

➔ EXPLORATORY DATA ANALYSIS

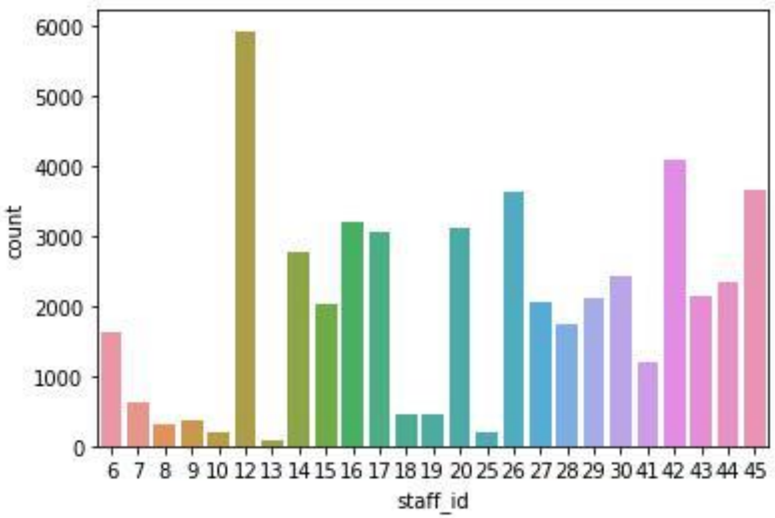
6. Univariate analysis on our dataset.

Performing **Univariate EDA** on our dataset.

In [7]:
```python
import seaborn as sns
```

```
In [8]: sns.countplot(df['staff_id'])
```

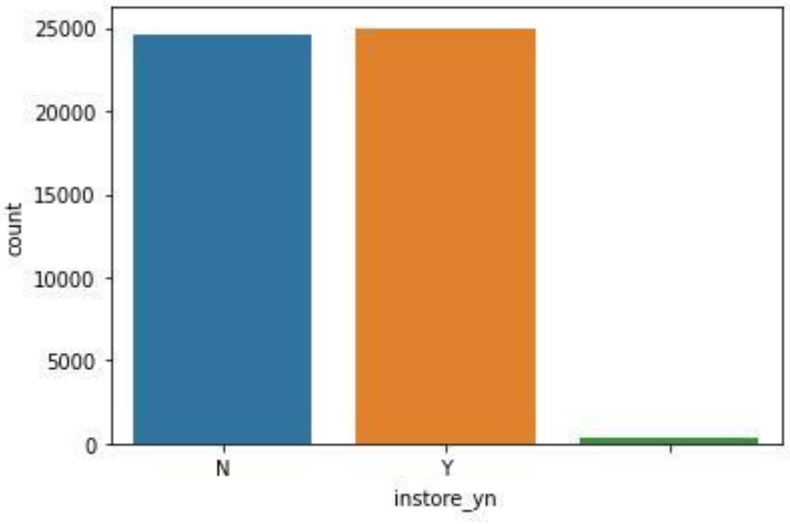
```
Out[8]: <AxesSubplot:xlabel='staff_id', ylabel='count'>
```



After seeing the countplot of staff_if , we can easily say that staff_id 12 is very often among all , *so we can conclide that the staff having id 12 might be very loyal to work or is having much pressure to work in perticular time frame*.

In [9]:
```
sns.countplot(df['instore_yn'])
```

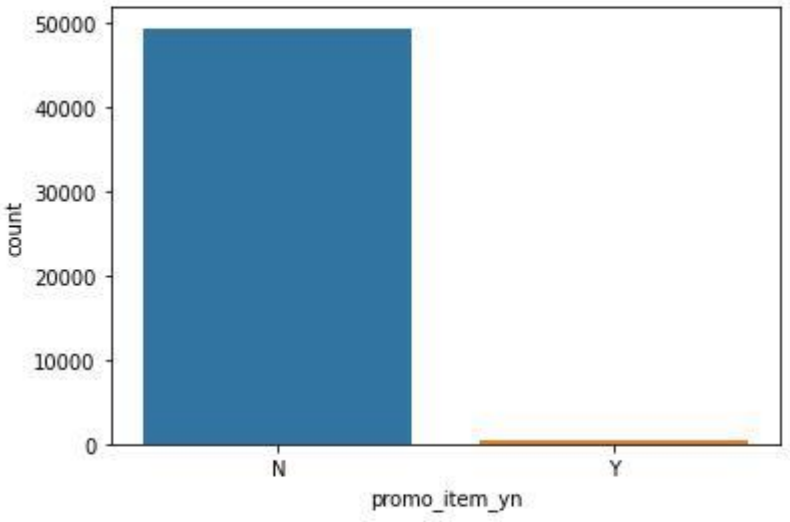
Out[9]: <AxesSubplot:xlabel='instore_yn', ylabel='count'>



Here Instore_yn has majorly two values Y and N. and it is having approximately same value count of Y and N , so it is **balanced**.

In [10]:
```
sns.countplot(df['promo_item_yn'])
```

Out[10]: <AxesSubplot:xlabel='promo_item_yn', ylabel='count'>



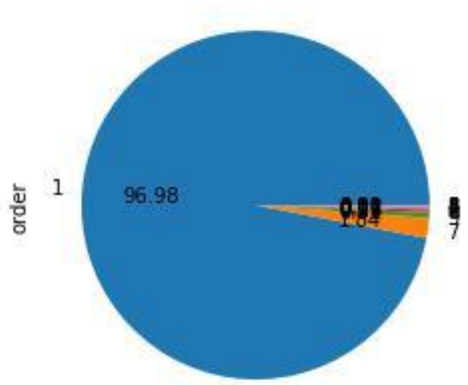
After plotting the count plot of promo_item_yn , we can clearly see that the dataset is **imbalanced** , so it will be better if we remove the column

In [11]:
```
df=df.drop(columns=['promo_item_yn'])
```

In [12]:
```
df.head()
```
Out[12]:

	transaction_id	transaction_date	transaction_time	sales_outlet_id	staff_id	customer_id	instore_yn	order	line_item_id	product_id	quantity	line
0	7	2019-04-01	12:04:43	3	12	558	N	1	1	52	1	
1	11	2019-04-01	15:54:39	3	17	781	N	1	1	27	2	
2	19	2019-04-01	14:34:59	3	17	788	Y	1	1	46	2	
3	32	2019-04-01	16:06:04	3	12	683	N	1	1	23	2	
4	33	2019-04-01	19:18:37	3	17	99	Y	1	1	34	1	

In [13]:
```
df['order'].value_counts().plot(kind='pie',autopct='%.2f')
```
Out[13]: <AxesSubplot:ylabel='order'>



After seeing the pie-chart we can say that the order 1 is most frequent amongst all. and it is also **imbalanced** so we will remove the column here.

In [14]:
```
df = df.drop(columns=['order'])
```

In [15]:
```python
df.head()
```

Out[15]:

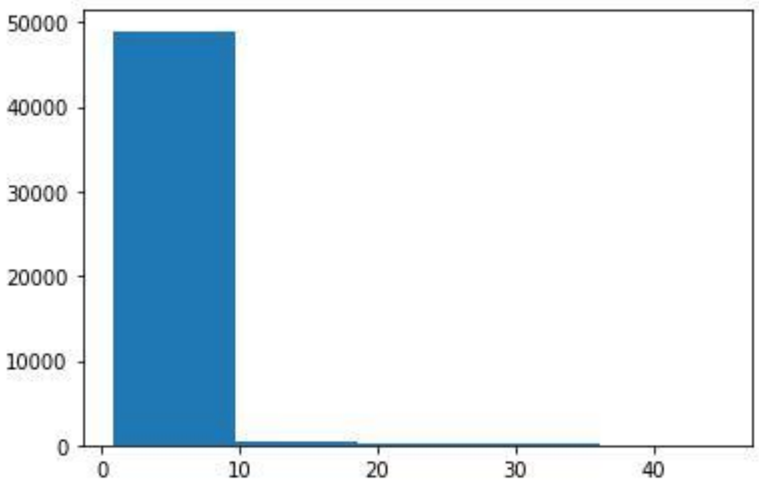
	transaction_id	transaction_date	transaction_time	sales_outlet_id	staff_id	customer_id	instore_yn	line_item_id	product_id	quantity	line_item_a
0	7	2019-04-01	12:04:43	3	12	558	N	1	52	1	
1	11	2019-04-01	15:54:39	3	17	781	N	1	27	2	
2	19	2019-04-01	14:34:59	3	17	788	Y	1	46	2	
3	32	2019-04-01	16:06:04	3	12	683	N	1	23	2	
4	33	2019-04-01	19:18:37	3	17	99	Y	1	34	1	

In [16]:
```python
import matplotlib.pyplot as plt
```

Plotting **Histograms** for columns in our dataset.

In [17]:
```python
plt.hist(df['unit_price'],bins=5)
```

Out[17]:
```
(array([48970.,   539.,   235.,    83.,    67.]),
 array([ 0.8 ,  9.64, 18.48, 27.32, 36.16, 45.  ]),
 <BarContainer object of 5 artists>)
```

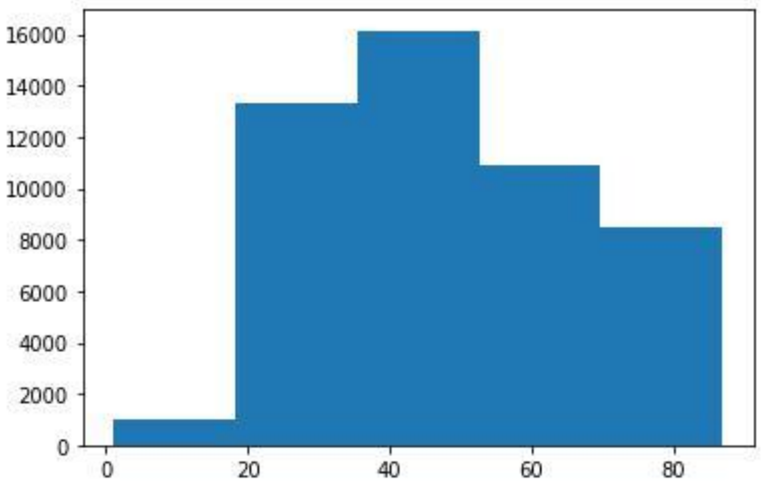


In [18]:
```
df['product_id'].unique()
```

Out[18]:
```
array([52, 27, 46, 23, 34, 32, 49, 60, 51, 35, 47, 25, 48, 53, 40, 37, 41,
38, 50, 59, 28, 77, 55, 54, 45, 79, 43, 61, 58, 42, 31, 39, 22, 76, 29,
33, 26, 30, 56, 74, 24, 71, 36, 69, 57, 70, 44, 78, 75, 73, 72, 87, 9, 84,
12, 6, 64, 63, 13, 65, 2, 7, 18, 20, 19, 10, 8, 15, 21, 4, 1, 17, 14, 82,
16, 3, 5, 81, 83, 11], dtype=int64)
```

In [19]:
```
plt.hist(df['product_id'],bins=5)
```

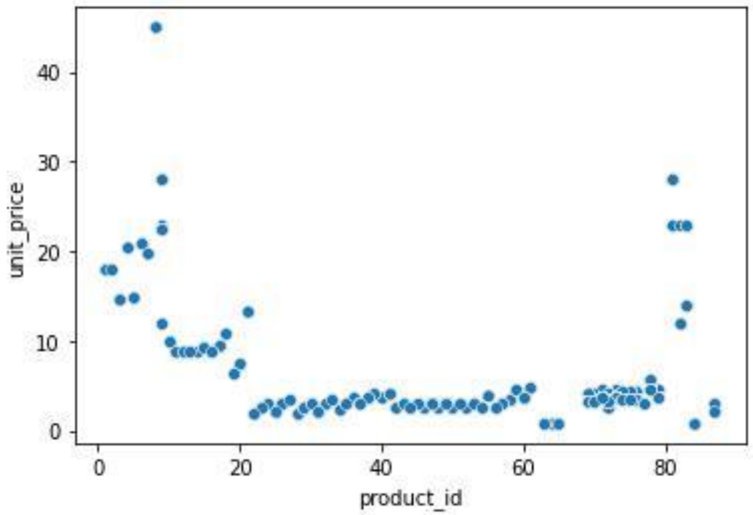
Out[19]:
```
(array([  988., 13318., 16150., 10923.,  8515.]),
 array([ 1. , 18.2, 35.4, 52.6, 69.8, 87. ]),
 <BarContainer object of 5 artists>)
```



7. Bi-variate analysis on our dataset.

In [20]:
```
sns.scatterplot(df['product_id'],df['unit_price'])
```

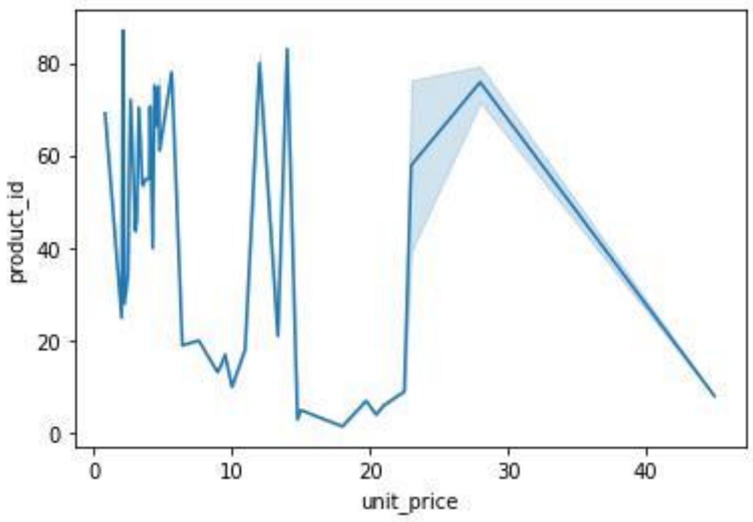
Out[20]:
```
<AxesSubplot:xlabel='product_id', ylabel='unit_price'>
```



So here in the scatterplot of **product_id vs unit_price** we can see that products having id between 0 to 20 is of high to medium of price and products having id between 20 to 80 is of low price , *it is so because it might possible that 0 to 20 product id is for some glossories and 20 to 80 product id id for some expensive products*.

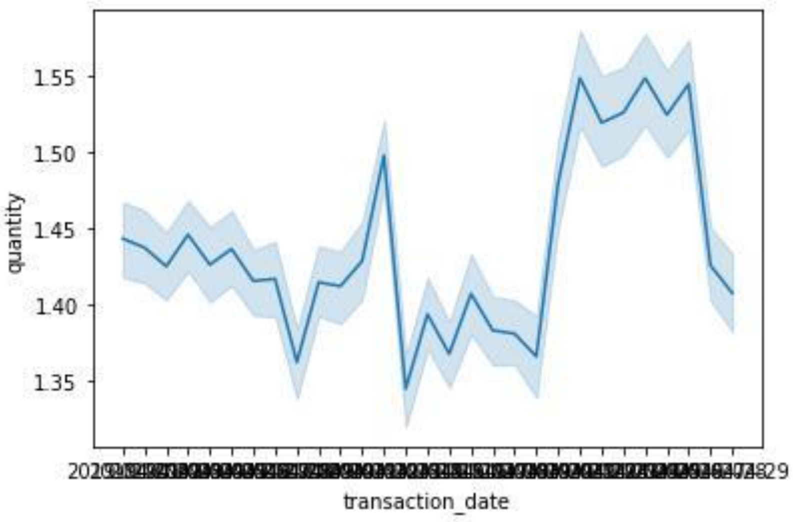
In [21]:
```
sns.lineplot(df['unit_price'],df['product_id'])
```

Out[21]: <AxesSubplot:xlabel='unit_price', ylabel='product_id'>



In [22]:
```python
sns.lineplot(df['transaction_date'],df['quantity'])
```

Out[22]: <AxesSubplot:xlabel='transaction_date', ylabel='quantity'>



=

➔ CUSTOMER DATA ANALYSIS

1. Reading the data.

In [23]:
```
df1=pd.read_csv('Data/customer.csv')
```

In [24]:
```
df1.head()
```

Out[24]:

	customer_id	home_store	customer_first-name	customer_email	customer_since	loyalty_card_number	birthdate	gender	birth_year
0	1	3	Kelly Key	Venus@adipiscing.edu	2017-01-04	908-424-2890	1950-05-29	M	1950
1	2	3	Clark Schroeder	Nora@fames.gov	2017-01-07	032-732-6308	1950-07-30	M	1950
2	3	3	Elvis Cardenas	Brianna@tellus.edu	2017-01-10	459-375-9187	1950-09-30	M	1950
3	4	3	Rafael Estes	Ina@non.gov	2017-01-13	576-640-9226	1950-12-01	M	1950
4	5	3	Colin Lynn	Dale@Integer.com	2017-01-15	344-674-6569	1951-02-01	M	1951

In [25]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2246 entries, 0 to 2245
Data columns (total 9 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   customer_id          2246 non-null   int64 
 1   home_store           2246 non-null   int64 
 2   customer_first-name  2246 non-null   object
 3   customer_email       2246 non-null   object
 4   customer_since       2246 non-null   object
 5   loyalty_card_number  2246 non-null   object
 6   birthdate            2246 non-null   object
 7   gender               2246 non-null   object
 8   birth_year           2246 non-null   int64 
dtypes: int64(3), object(6)
memory usage: 158.0+ KB
```

In [26]:
```
df1.describe()
```
Out[26]:

	customer_id	home_store	birth_year
count	2246.000000	2246.000000	2246.000000
mean	4285.902048	4.956812	1978.385574
std	3088.088265	1.852562	14.925503
min	1.000000	3.000000	1950.000000
25%	562.250000	3.000000	1965.000000
50%	5323.500000	5.000000	1981.000000
75%	5884.750000	5.000000	1991.000000
max	8501.000000	8.000000	2001.000000

2. Checking for correlations.

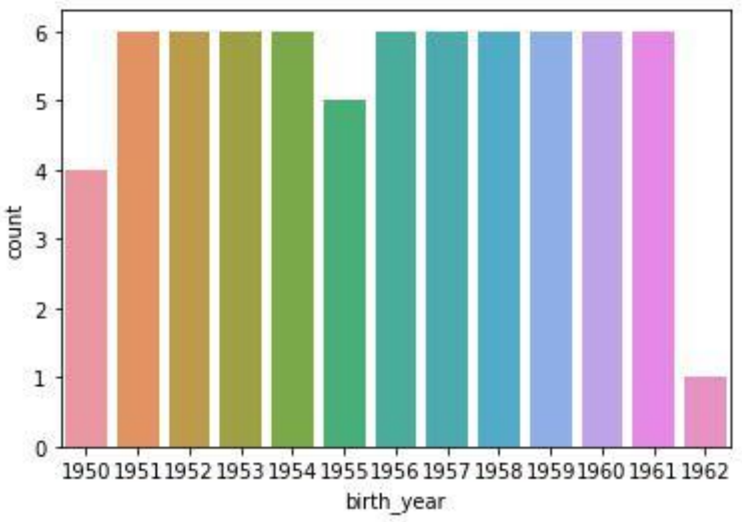
In [27]:
```
df1.corr()
```
Out[27]:

	customer_id	home_store	birth_year
customer_id	1.000000	0.948053	0.134341
home_store	0.948053	1.000000	0.084356
birth_year	0.134341	0.084356	1.000000

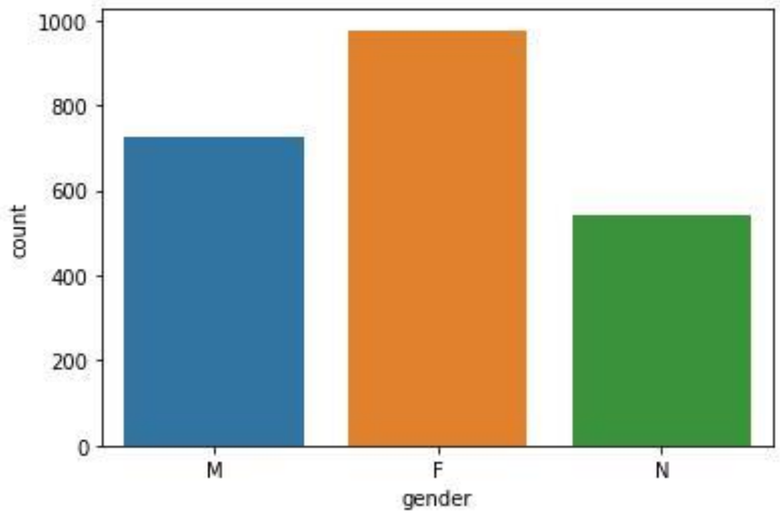
3. Performing Exploratory Data Analysis.

In [28]:
```
sns.countplot(df1['birth_year'][:70])
```

Out[28]: <AxesSubplot:xlabel='birth_year', ylabel='count'>



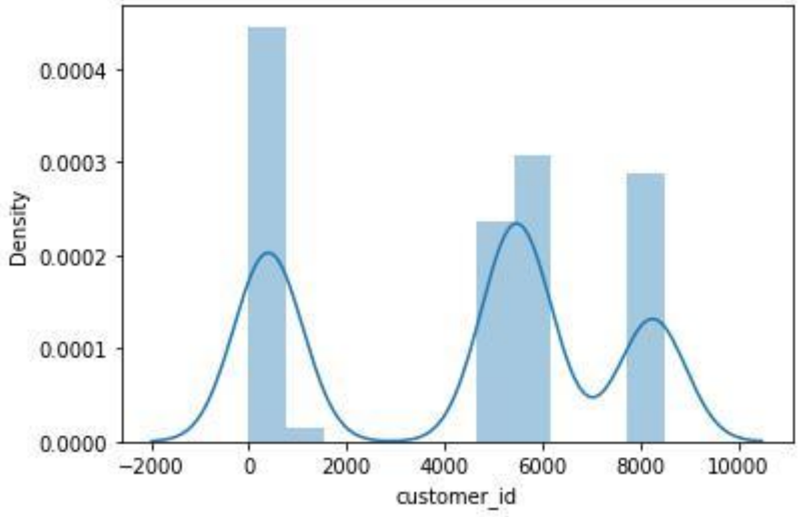
In [29]:
```
sns.countplot(df1['gender'])
```
Out[29]: <AxesSubplot:xlabel='gender', ylabel='count'>



Here we can clearly see that the store has more Female customers as compared to Male.

In [30]:
```
sns.distplot(df1[df1['customer_since']>'07-01-2017']['customer_id'],hist=True)
```

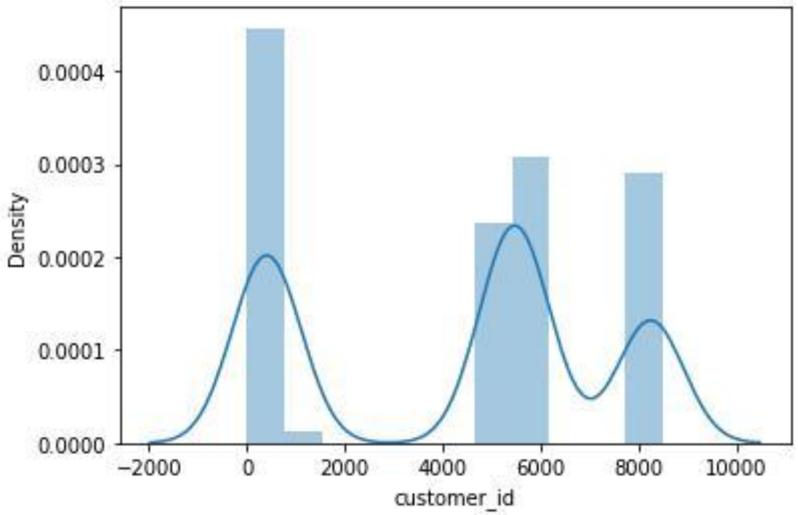
Out[30]: <AxesSubplot:xlabel='customer_id', ylabel='Density'>



From the distribution plot we can see that more older customers has `higher customers id , which indicates that customers ids are provided sequently.

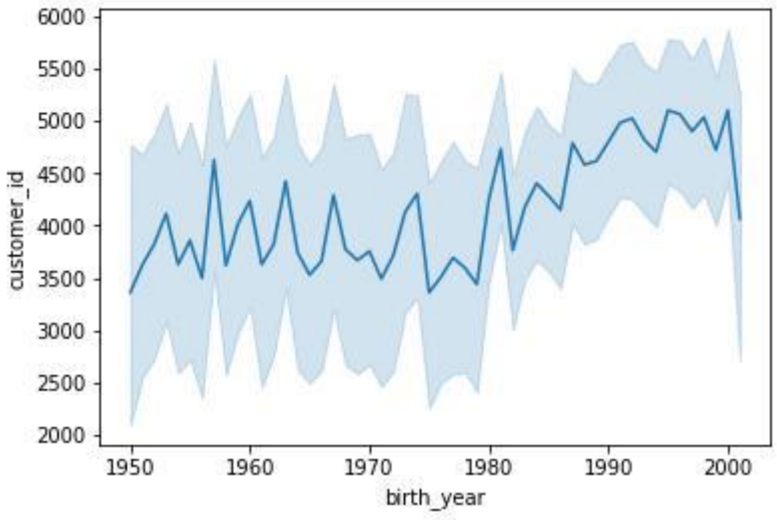
In [31]:
```
sns.distplot(df1[df1['birth_year']>1950]['customer_id'],hist=True)
```

Out[31]: <AxesSubplot:xlabel='customer_id', ylabel='Density'>



```
In [32]: sns.lineplot(df1['birth_year'],df1['customer_id'])
```

```
Out[32]: <AxesSubplot:xlabel='birth_year', ylabel='customer_id'>
```




```
============================================================
```

➔ PASTRY INVENTORY DATA ANALYSIS

1. Reading the data.

In [33]:
```
df2=pd.read_csv('Data/pastry inventory.csv')
```

In [34]:
```
df2.head()
```

Out[34]:

	sales_outlet_id	transaction_date	product_id	start_of_day	quantity_sold	waste	% waste
0	3	4/1/2019	69	18	8	10	56%
1	3	4/1/2019	70	18	12	6	33%
2	3	4/1/2019	71	18	8	10	56%
3	3	4/1/2019	72	48	9	39	81%
4	3	4/1/2019	73	18	9	9	50%

In [35]:
```
df2.tail()
```

Out[35]:

	sales_outlet_id	transaction_date	product_id	start_of_day	quantity_sold	waste	% waste
302	8	4/27/2019	69	18	1	17	94%
303	8	4/27/2019	70	18	4	14	78%
304	8	4/27/2019	71	18	2	16	89%
305	8	4/27/2019	72	48	19	29	60%
306	8	4/27/2019	73	18	4	14	78%

In [36]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307 entries, 0 to 306
Data columns (total 7 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   sales_outlet_id   307 non-null    int64 
 1   transaction_date  307 non-null    object
 2   product_id        307 non-null    int64 
 3   start_of_day      307 non-null    int64 
 4   quantity_sold     307 non-null    int64 
 5   waste             307 non-null    int64 
 6   % waste           307 non-null    object
dtypes: int64(5), object(2)
memory usage: 16.9+ KB
```

In [37]:
```
df2.describe()
```

Out[37]:

	sales_outlet_id	product_id	start_of_day	quantity_sold	waste
count	307.000000	307.000000	307.000000	307.000000	307.000000
mean	5.394137	70.983713	24.058632	9.296417	14.657980
std	2.049477	1.417582	12.063414	5.440115	11.202108
min	3.000000	69.000000	18.000000	0.000000	0.000000
25%	3.000000	70.000000	18.000000	6.000000	8.000000
50%	5.000000	71.000000	18.000000	8.000000	11.000000
75%	8.000000	72.000000	18.000000	11.000000	15.000000
max	8.000000	73.000000	48.000000	32.000000	47.000000

In [38]:
```
df2.corr()
```

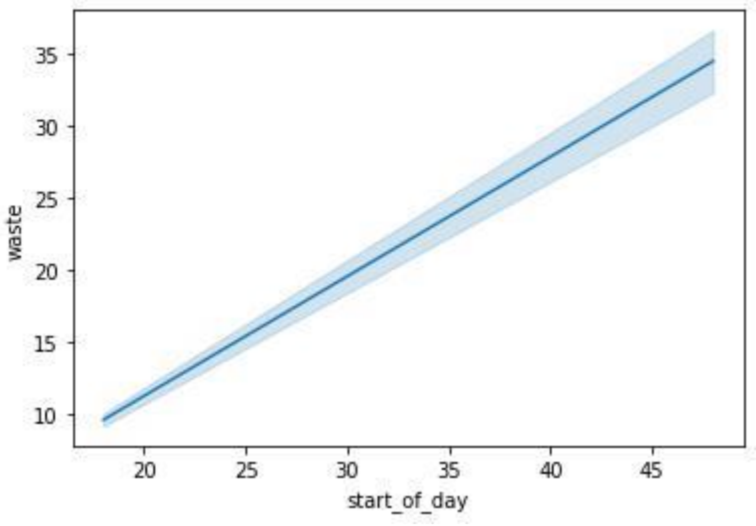
Out[38]:

	sales_outlet_id	product_id	start_of_day	quantity_sold	waste
sales_outlet_id	1.000000	0.013465	-0.005696	0.120800	-0.053893
product_id	0.013465	1.000000	0.361235	0.134961	0.339001
start_of_day	-0.005696	0.361235	1.000000	0.393825	0.893224
quantity_sold	0.120800	0.134961	0.393825	1.000000	-0.043859
waste	-0.053893	0.339001	0.893224	-0.043859	1.000000

2. Exploratory Data Analysis.

In [39]:
```
sns.lineplot(df2['start_of_day'],df2['waste'])
```

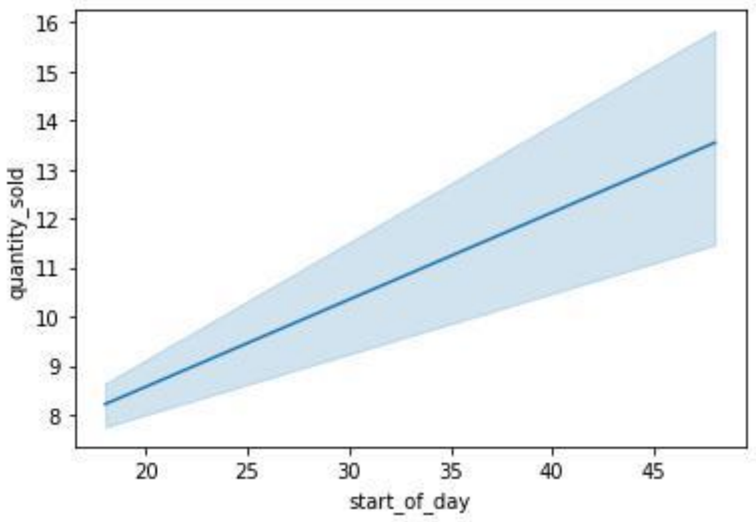
Out[39]: <AxesSubplot:xlabel='start_of_day', ylabel='waste'>



Here we can see from the lineplot that as start of the day increases , the waste is also increasing which is showing linear behaviour.

```
In [40]: sns.lineplot(df2['start_of_day'],df2['quantity_sold'])
```

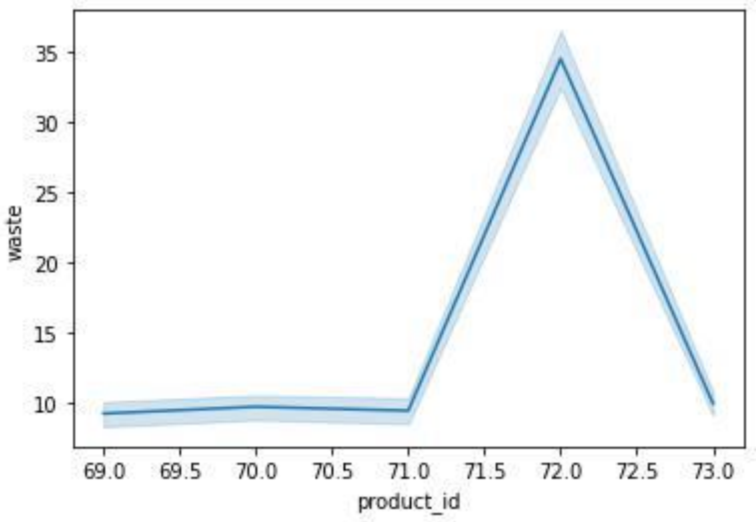
```
Out[40]: <AxesSubplot:xlabel='start_of_day', ylabel='quantity_sold'>
```



Here we can see from the lineplot that as start of the day increases , the waste is also increasing which is showing linear behaviour.

In [41]:
```
sns.lineplot(df2['product_id'],df2['waste'])
```

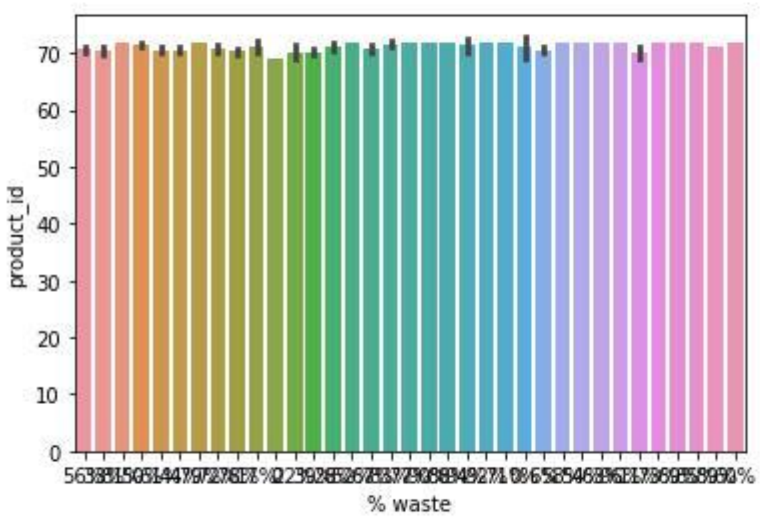
Out[41]: <AxesSubplot:xlabel='product_id', ylabel='waste'>



In [42]:

```
sns.barplot(df2['% waste'],df2['product_id'])
```

Out[42]: <AxesSubplot:xlabel='% waste', ylabel='product_id'>




```
==========================================================================================
```

➔ PRODUCT DATA ANALYSIS

1. Reading the dataset.

In [43]:
```
df3=pd.read_csv('Data/product.csv')
```

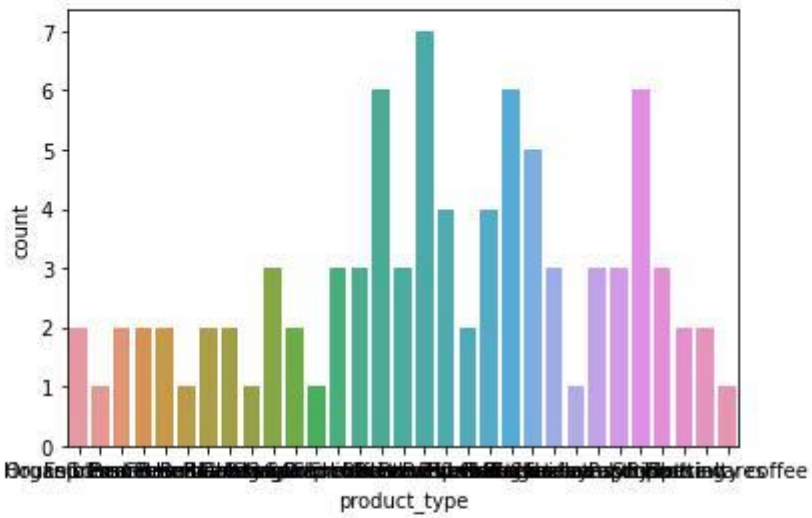
In [44]:
```
df3.head()
```

Out[44]:

	product_id	product_group	product_category	product_type	product	product_description	unit_of_measure	current_wholesale_price	current_ret
0	1	Whole Bean/Teas	Coffee beans	Organic Beans	Brazilian - Organic	It's like Carnival in a cup. Clean and smooth.	12 oz	14.40	
1	2	Whole Bean/Teas	Coffee beans	House blend Beans	Our Old Time Diner Blend	Out packed blend of beans that is reminiscent ...	12 oz	14.40	
2	3	Whole Bean/Teas	Coffee beans	Espresso Beans	Espresso Roast	Our house blend for a good espresso shot.	1 lb	11.80	
3	4	Whole Bean/Teas	Coffee beans	Espresso Beans	Primo Espresso Roast	Our primium single source of hand roasted beans.	1 lb	16.36	
4	5	Whole Bean/Teas	Coffee beans	Gourmet Beans	Columbian Medium Roast	A smooth cup of coffee any time of day.	1 lb	12.00	

In [45]:
```
sns.countplot(df3['product_type'])
```

Out[45]: <AxesSubplot:xlabel='product_type', ylabel='count'>

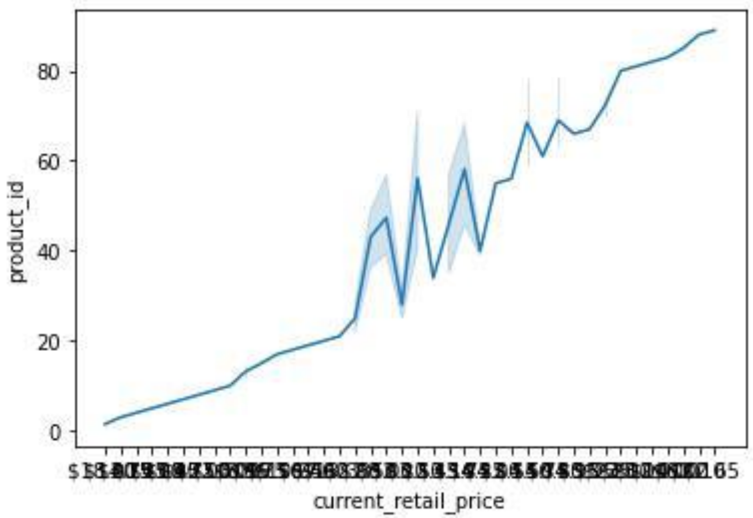


```
In [46]: df3['product_type'].value_counts()

Out[46]: Barista Espresso         7
         Gourmet brewed coffee    6
         Scone                    6
         Brewed Chai tea          6
         Hot chocolate            5
         Brewed herbal tea        4
         Brewed Black tea         4
         Pastry                   3
         Drip coffee              3
         Premium brewed coffee    3
         Chai tea                 3
         Seasonal drink           3
         Regular syrup            3
         Biscotti                 3
         Organic brewed coffee    3
         Organic Beans            2
         Drinking Chocolate       2
         Housewares               2
         Espresso Beans           2
         Premium Beans            2
         Gourmet Beans            2
         Brewed Green tea         2
         Herbal tea               2
         Black tea                2
         Clothing                 2
         Sugar free syrup         1
         Green tea                1
         Green beans              1
         House blend Beans        1
         Organic Chocolate        1
         Specialty coffee         1
         Name: product_type, dtype: int64
```

In [47]:
```
sns.lineplot(df3['current_retail_price'],df3['product_id'])
```

Out[47]: <AxesSubplot:xlabel='current_retail_price', ylabel='product_id'>

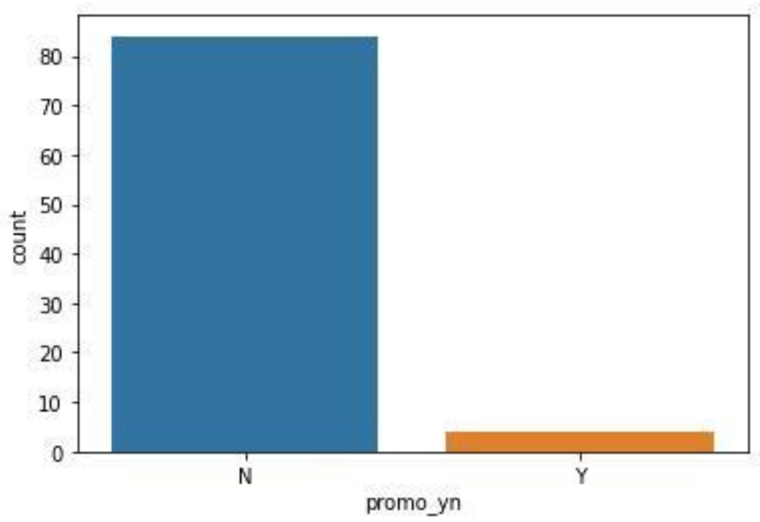


In [48]:

```
sns.countplot(df3['promo_yn'])
```

Out[48]:

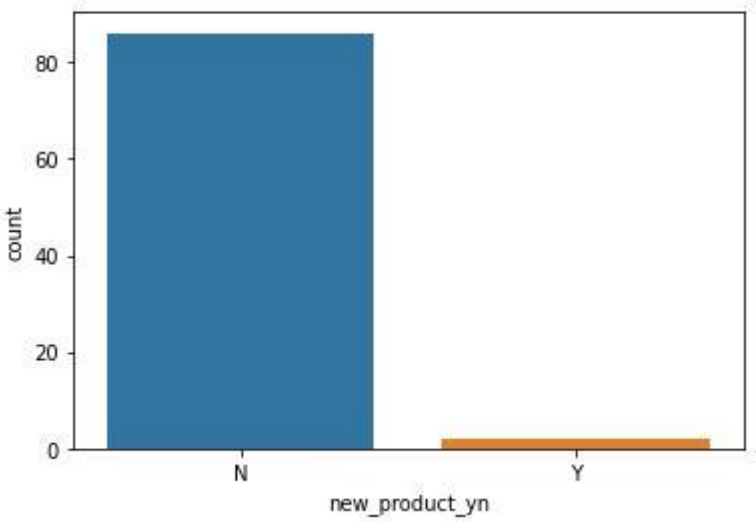
```
<AxesSubplot:xlabel='promo_yn', ylabel='count'>
```



Here from the count plot we can conclude that there is not any promo available for most of the products available in the shop.

```
In [49]: sns.countplot(df3['new_product_yn'])
```

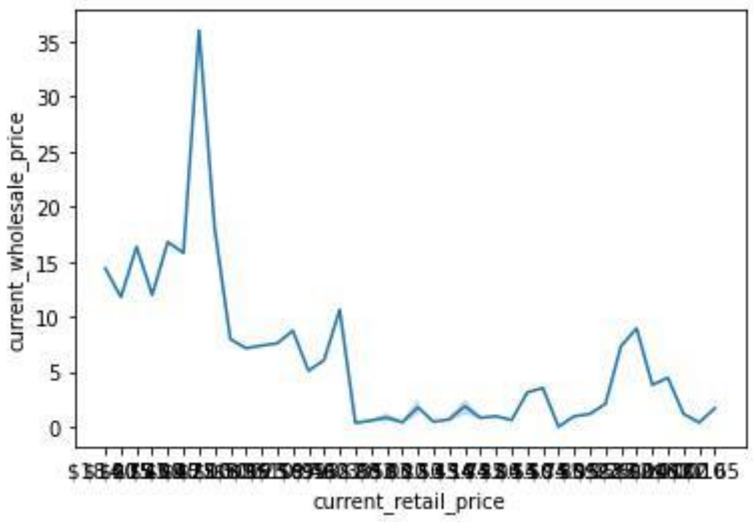
Out[49]: <AxesSubplot:xlabel='new_product_yn', ylabel='count'>



Here from the count plot we can conclude that most of the products available in the shop are not new , they are old products.

In [50]:
```
sns.lineplot(df3['current_retail_price'],df3['current_wholesale_price'])
```

Out[50]: <AxesSubplot:xlabel='current_retail_price', ylabel='current_wholesale_price'>



➔ PANDAS PROFILING

Pandas profiling is a library which is very useful for exploratory data analysis and using which we can automate some part of our exploratory data analysis on our dataset.

In [51]:
```
from pandas_profiling import ProfileReport
```

In [52]:
```
 prof = ProfileReport(df)
prof.to_file(output_file='sales receipts.html')
```

\

In [53]:
```
 prof = ProfileReport(df1)
prof.to_file(output_file='customers.html')
```

In [54]:
```
prof = ProfileReport(df2)
prof.to_file(output_file='pastry inventory.html')
```

In [55]:
```
prof = ProfileReport(df3)
prof.to_file(output_file='product.html')
```

In [ ]:
```
```

GITHUB LINK : https://github.com/ParthShukla211/FEYNN-LABS_PROJECT---1