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]:
```
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]:
```
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	ur
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]:

```
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	-0.03393
sales_outlet_id	-0.134200	1.000000	0.696921	0.429706	0.012392	0.004210	0.024360	-0.002860	0.004255	-0.00167
staff_id	-0.050462	0.696921	1.000000	0.294914	0.015983	-0.008372	0.010359	0.002996	0.003410	-0.00039
customer_id	0.004820	0.429706	0.294914	1.000000	-0.018909	-0.008114	0.001156	0.011265	-0.005202	-0.01621
order	-0.052610	0.012392	0.015983	-0.018909	1.000000	0.000616	-0.173570	-0.125321	0.452822	0.75872
line_item_id	-0.047631	0.004210	-0.008372	-0.008114	0.000616	1.000000	0.604757	-0.315383	-0.050380	0.07405
product_id	-0.046251	0.024360	0.010359	0.001156	-0.173570	0.604757	1.000000	-0.175536	-0.164309	-0.13853
quantity	0.015083	-0.002860	0.002996	0.011265	-0.125321	-0.315383	-0.175536	1.000000	0.353336	-0.11920
line_item_amount	-0.010319	0.004255	0.003410	-0.005202	0.452822	-0.050380	-0.164309	0.353336	1.000000	0.67216
unit_price	-0.033934	-0.001673	-0.000396	-0.016218	0.758723	0.074058	-0.138539	-0.119205	0.672168	1.00000

EXPLORATORY DATA ANALYSIS

6. Univariate analysis on our dataset.

Performing **Univariate EDA** on our dataset.

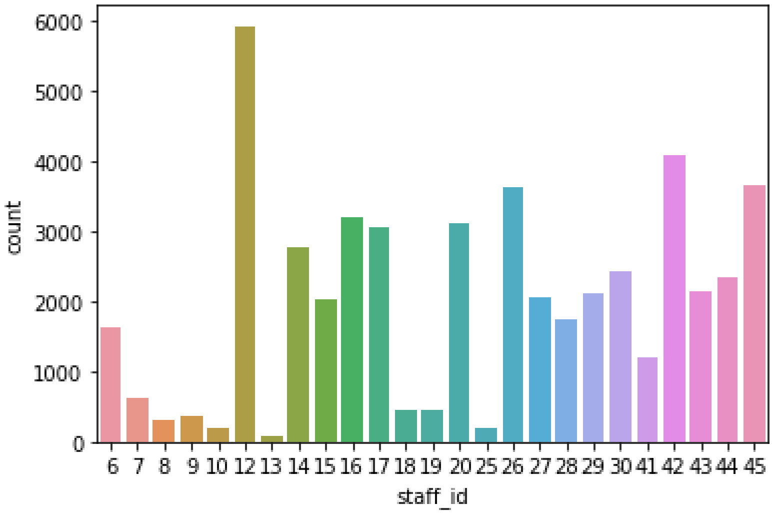
In [7]:

```
import seaborn as sns
```

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

```
c:\users\hp\appdata\local\programs\python\python37\lib\site-packages\seaborn\_decorators.py:43: FutureWarning: Pass the
following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passi
ng other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

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

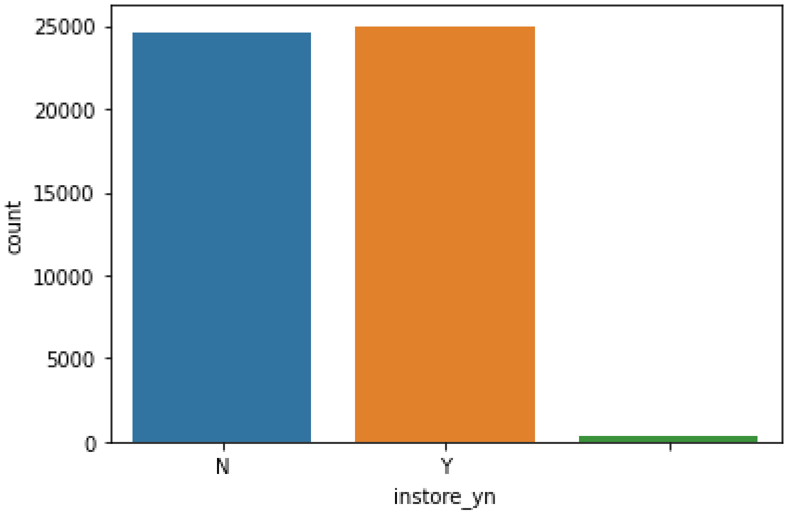


After seeing the countplot of staff_if , we can easily say that staff_id 12 is very often among all , *so we can conclide that the staff having id 12 might be very loyal to work or is having much pressure to work in perticular time frame*.

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

```
c:\users\hp\appdata\local\programs\python\python37\lib\site-packages\seaborn\_decorators.py:43: FutureWarning: Pass the
following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passi
ng other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

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

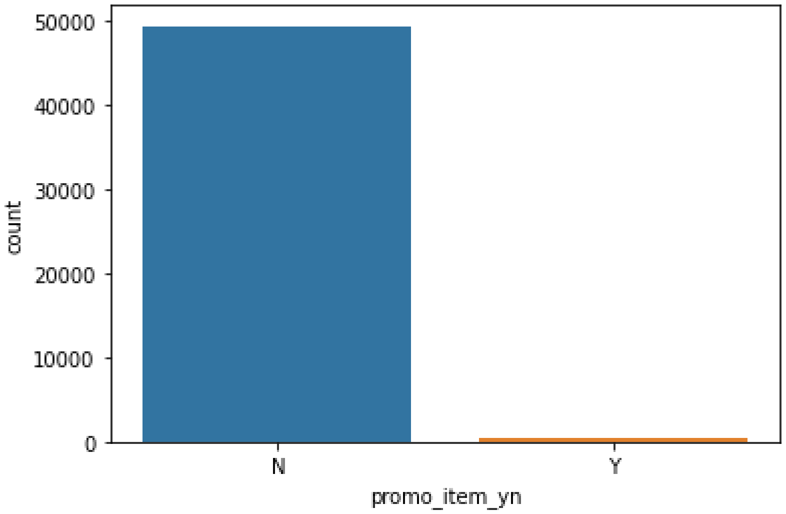


Here Instore_yn has majorly two values Y and N. and it is having approximately same value count of Y and N , so it is **balanced**.

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

```
c:\users\hp\appdata\local\programs\python\python37\lib\site-packages\seaborn\_decorators.py:43: FutureWarning: Pass the
following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passi
ng other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

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]:

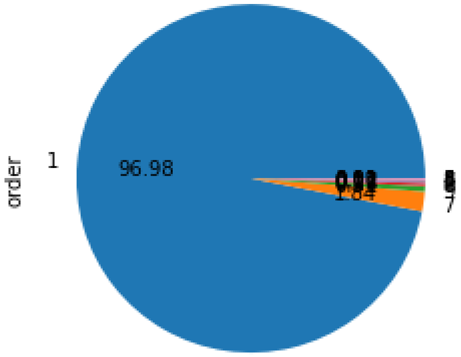
	transaction_id	transaction_date	transaction_time	sales_outlet_id	staff_id	customer_id	instore_yn	order	line_item_id	product_id	quantity	line
0	7	2019-04-01	12:04:43	3	12	558	N	1	1	52	1	
1	11	2019-04-01	15:54:39	3	17	781	N	1	1	27	2	
2	19	2019-04-01	14:34:59	3	17	788	Y	1	1	46	2	
3	32	2019-04-01	16:06:04	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]:
```
df.head()
```

Out[15]:

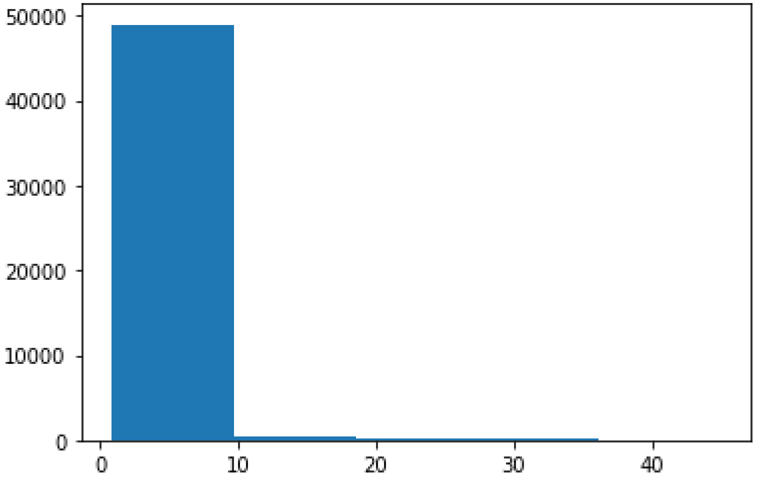
	transaction_id	transaction_date	transaction_time	sales_outlet_id	staff_id	customer_id	instore_yn	line_item_id	product_id	quantity	line_item_
0	7	2019-04-01	12:04:43	3	12	558	N	1	52	1	
1	11	2019-04-01	15:54:39	3	17	781	N	1	27	2	
2	19	2019-04-01	14:34:59	3	17	788	Y	1	46	2	
3	32	2019-04-01	16:06:04	3	12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

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

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

In [17]:
```
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>)
```

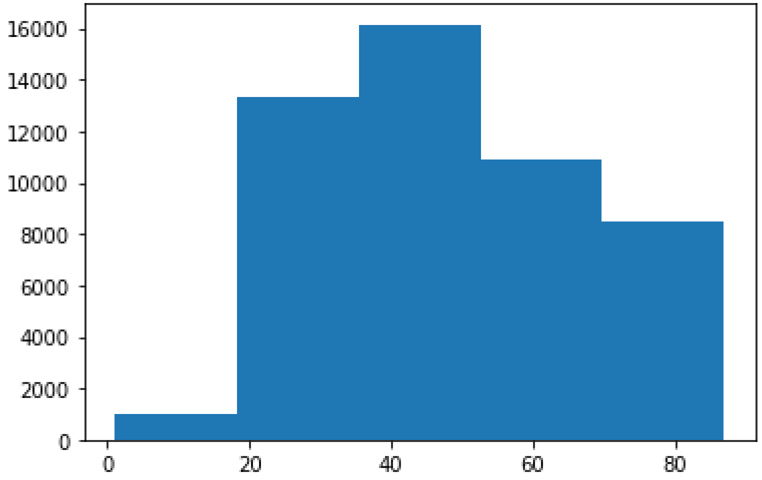


```
In [18]: df['product_id'].unique()
```

```
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>)
```

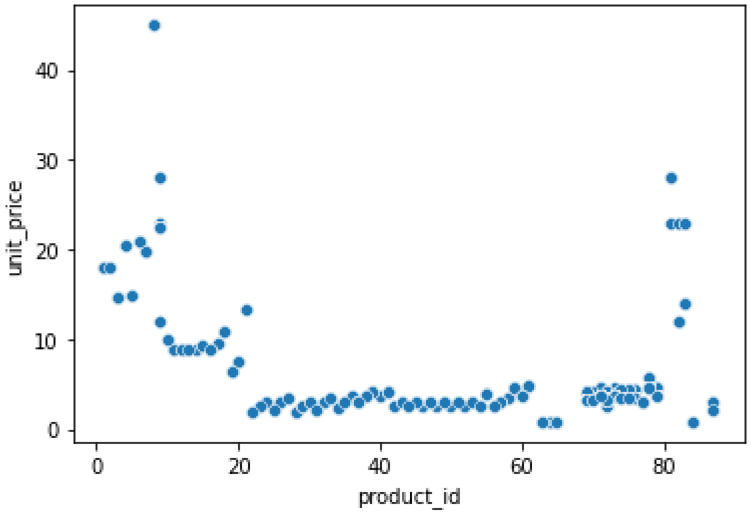


7. Bi-variate analysis on our dataset.

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

```
c:\users\hp\appdata\local\programs\python\python37\lib\site-packages\seaborn\_decorators.py:43: FutureWarning: Pass the
following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and pa
ssing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

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

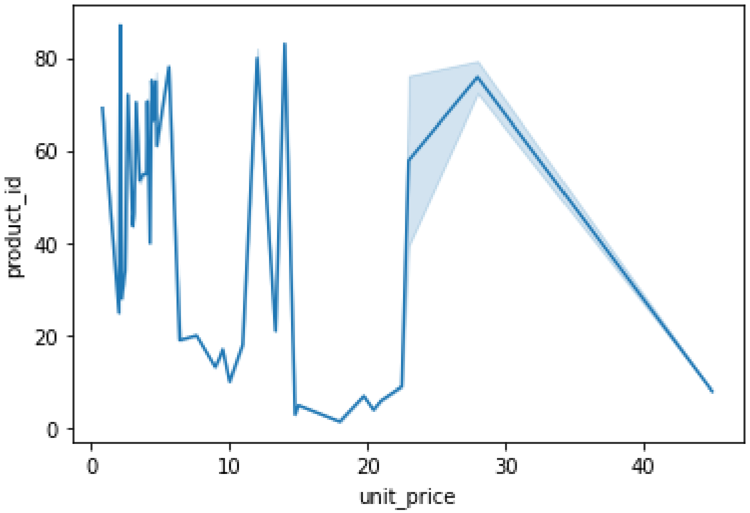


So here in the scatterplot of **product_id vs unit_price** we can see that products having id between 0 to 20 is of high to medium of price and products having id between 20 to 80 is of low price , *it is so because it might possible that 0 to 20 product id is for some glossories and 20 to 80 product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

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

```
c:\users\hp\appdata\local\programs\python\python37\lib\site-packages\seaborn\_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

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



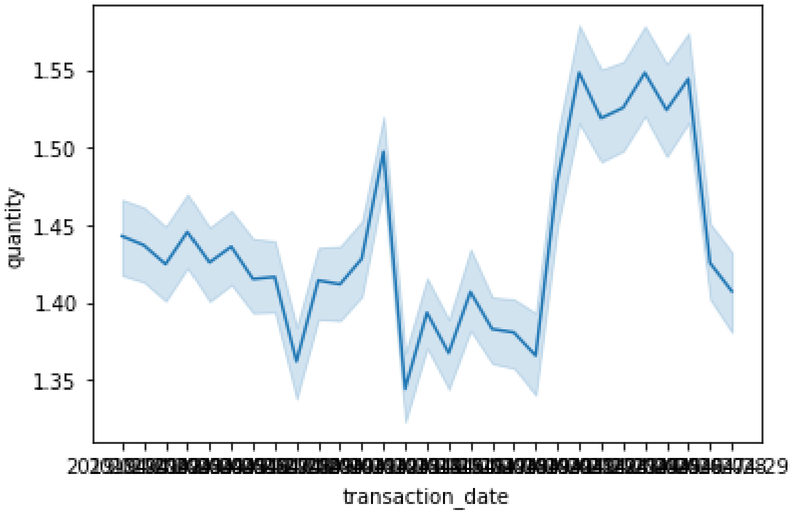
In [22]:

```
sns.lineplot(df['transaction_date'],df['quantity'])
```

```
c:\users\hp\appdata\local\programs\python\python37\lib\site-packages\seaborn\_decorators.py:43: FutureWarning: Pass the
following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and pa
ssing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[22]:

```
<AxesSubplot:xlabel='transaction_date', ylabel='quantity'>
```



In [ ]: