Week 2 Data Exploration

In [1]:

```
# This code appears in every demonstration Notebook.
# By default, when you run each cell, only the last output of the codes will show.
# This code makes all outputs of a cell show.
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

We will explore the user and order data from JD.com

1. We import the necessary packages.

In [2]:

```
import pandas as pd
```

2. We read in the datasets.

In [3]:

```
users = pd.read_csv('JD_user_data.csv')
orders = pd.read_csv('JD_order_data.csv')
```

3. Take a look at the data

In [4]:

```
users.head()
orders.head()
```

Out[4]:

	user_ID	user_level	first_order_month	plus	gender	age	marital_status	educatio
0	000089d6a6	1	2017-08	0	F	26-35	S	
1	0000babd1f	1	2018-03	0	U	U	U	-
2	0000bc018b	3	2016-06	0	F	>=56	M	
3	0000d0e5ab	3	2014-06	0	M	26-35	M	
4	0000dce472	3	2012-08	1	U	U	U	-

Out[4]:

	order_ID	user_ID	sku_ID	order_date	order_time	quantity	type	promise
0	d0cf5cc6db	0abe9ef2ce	581d5b54c1	2018-03-01	2018-03-01 17:14:25.0	1	2	-
1	7444318d01	33a9e56257	067b673f2b	2018-03-01	2018-03-01 11:10:40.0	1	1	2
2	f973b01694	4ea3cf408f	623d0a582a	2018-03-01	2018-03-01 09:13:26.0	1	1	2
3	8c1cec8d4b	b87cb736cb	fc5289b139	2018-03-01	2018-03-01 21:29:50.0	1	1	2
4	d43a33c38a	4829223b6f	623d0a582a	2018-03-01	2018-03-01 19:13:37.0	1	1	1

In [5]:
```
users.head(10) # The number argument specifies the number of rows to show
```

Out[5]:

	user_ID	user_level	first_order_month	plus	gender	age	marital_status	educatio
0	000089d6a6	1	2017-08	0	F	26-35	S	
1	0000babd1f	1	2018-03	0	U	U	U	-
2	0000bc018b	3	2016-06	0	F	>=56	M	
3	0000d0e5ab	3	2014-06	0	M	26-35	M	
4	0000dce472	3	2012-08	1	U	U	U	-
5	0000f81d1b	1	2018-02	0	F	26-35	M	
6	00012bb423	4	2008-11	1	F	26-35	M	
7	00015ff032	3	2015-06	1	M	26-35	M	
8	0001aa7059	4	2014-06	0	F	36-45	M	
9	0001bbdc89	2	2017-12	0	F	16-25	S	

In [6]:

```
users.columns
# Displays the variables of the dataframe
```

Out[6]:

```
Index(['user_ID', 'user_level', 'first_order_month', 'plus', 'gender', 'age',
       'marital_status', 'education', 'city_level', 'purchase_power'],
      dtype='object')
```

In [7]:

```
# Lets explore the gender variable in users
users['gender'].value_counts()
# value_counts() gives the frequency distribution
```

Out[7]:

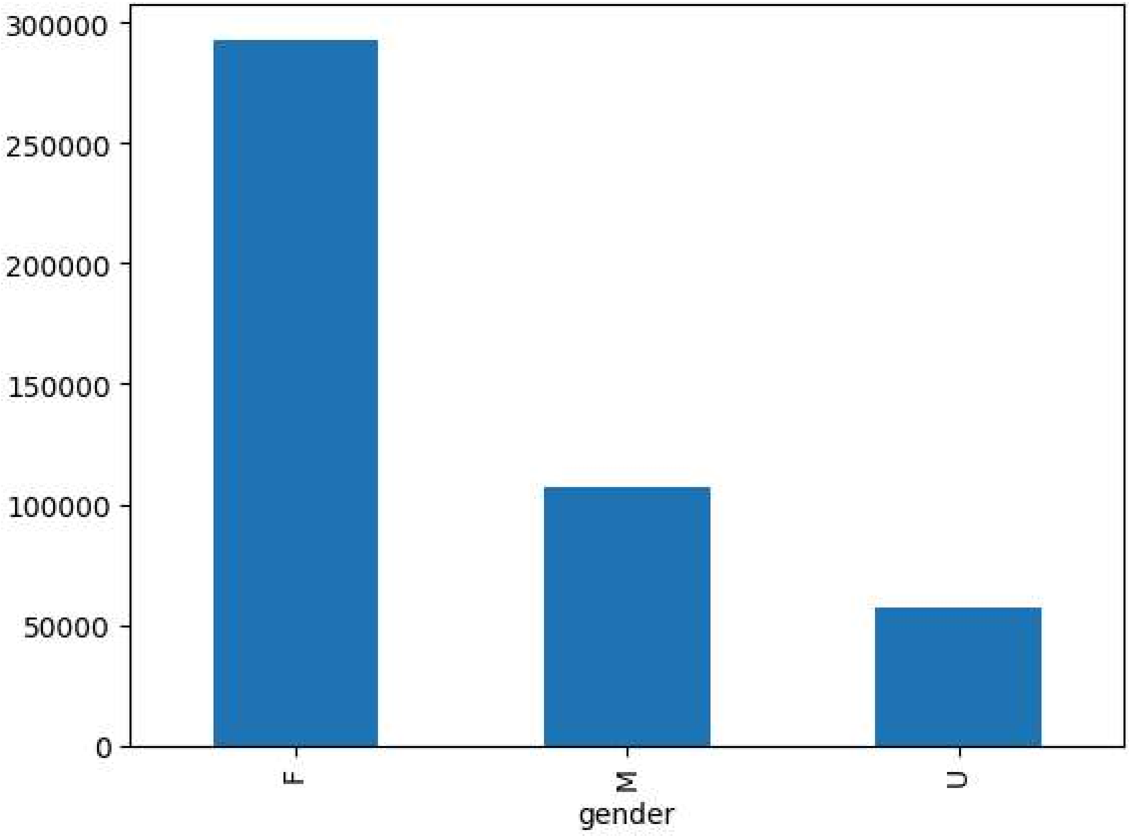
```
gender
F    292897
M    107084
U     57317
Name: count, dtype: int64
```

In [8]:

```
# Make a bar chart for the frquency distribution
users['gender'].value_counts().plot(kind = 'bar')
```

Out[8]:

```
<Axes: xlabel='gender'>
```



In [9]:

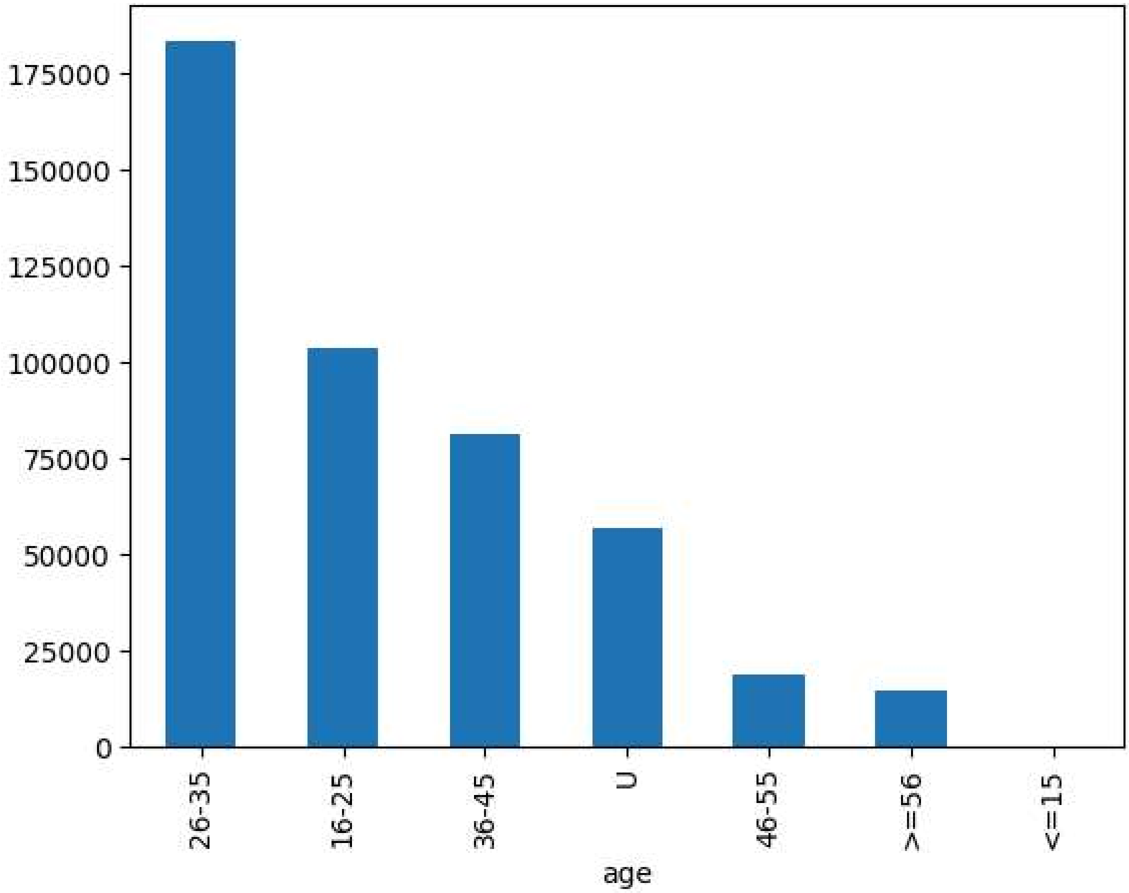
```
# Exercise: exploring age
age_dis = users['age'].value_counts()
```

In [10]:

```
users['age'].value_counts().plot(kind = 'bar')
```

Out[10]:

```
<Axes: xlabel='age'>
```



In [11]:
```
import matplotlib.pyplot as plt

#Importing the graph package matplotlib
```

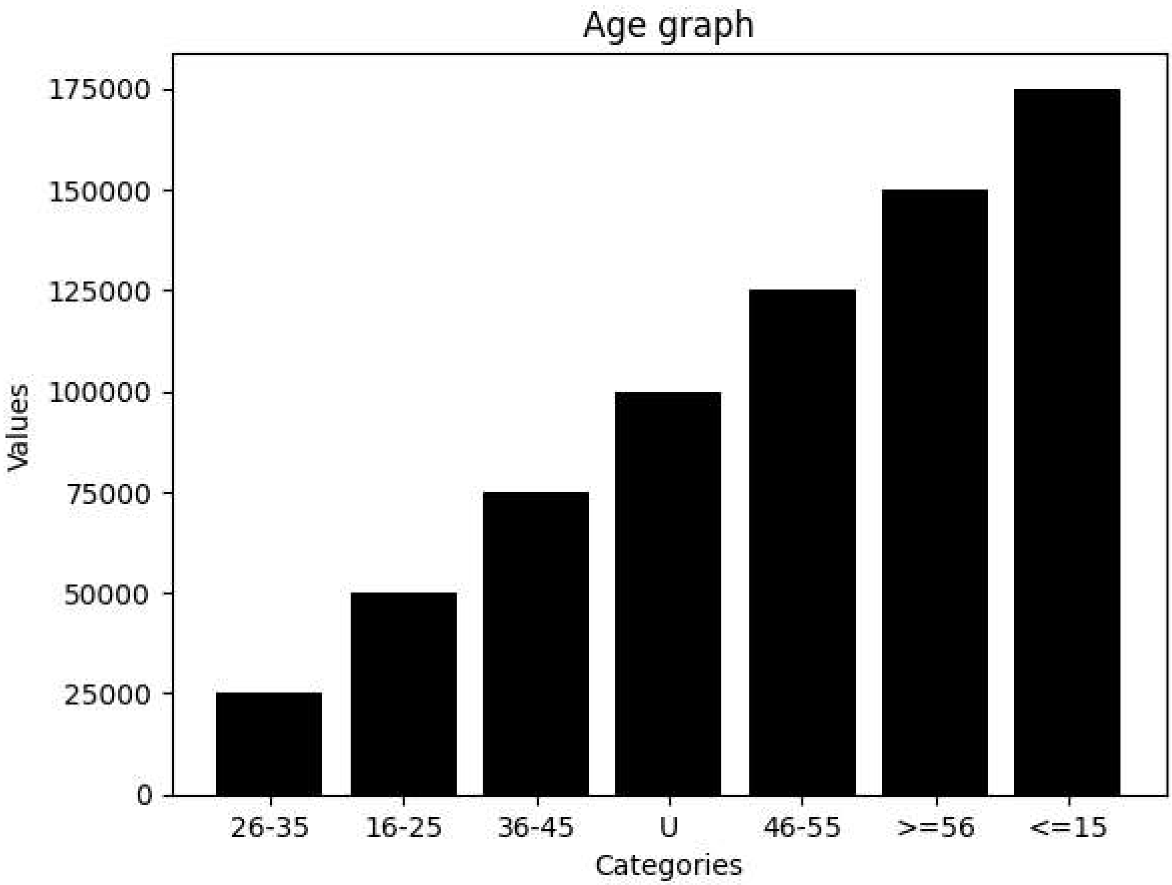
In [12]:
```
# Sample data
categories = ['26-35', '16-25', '36-45', 'U', '46-55', '>=56', '<=15']
values = [25000, 50000, 75000, 100000, 125000, 150000, 175000]

# Create a bar graph
plt.bar(categories, values, color='black')

# Add labels and title
plt.xlabel('Categories')
plt.ylabel('Values')
plt.title('Age graph')

# Show the plot
plt.show()
```

Out[12]:
```
<BarContainer object of 7 artists>
```

Out[12]:
```
Text(0.5, 0, 'Categories')
```

Out[12]:
```
Text(0, 0.5, 'Values')
```

Out[12]:
```
Text(0.5, 1.0, 'Age graph')
```



In [13]:
```
age_dis.index
```

Out[13]:
```
Index(['26-35', '16-25', '36-45', 'U', '46-55', '>=56', '<=15'], dtype='object', n
ame='age')
```

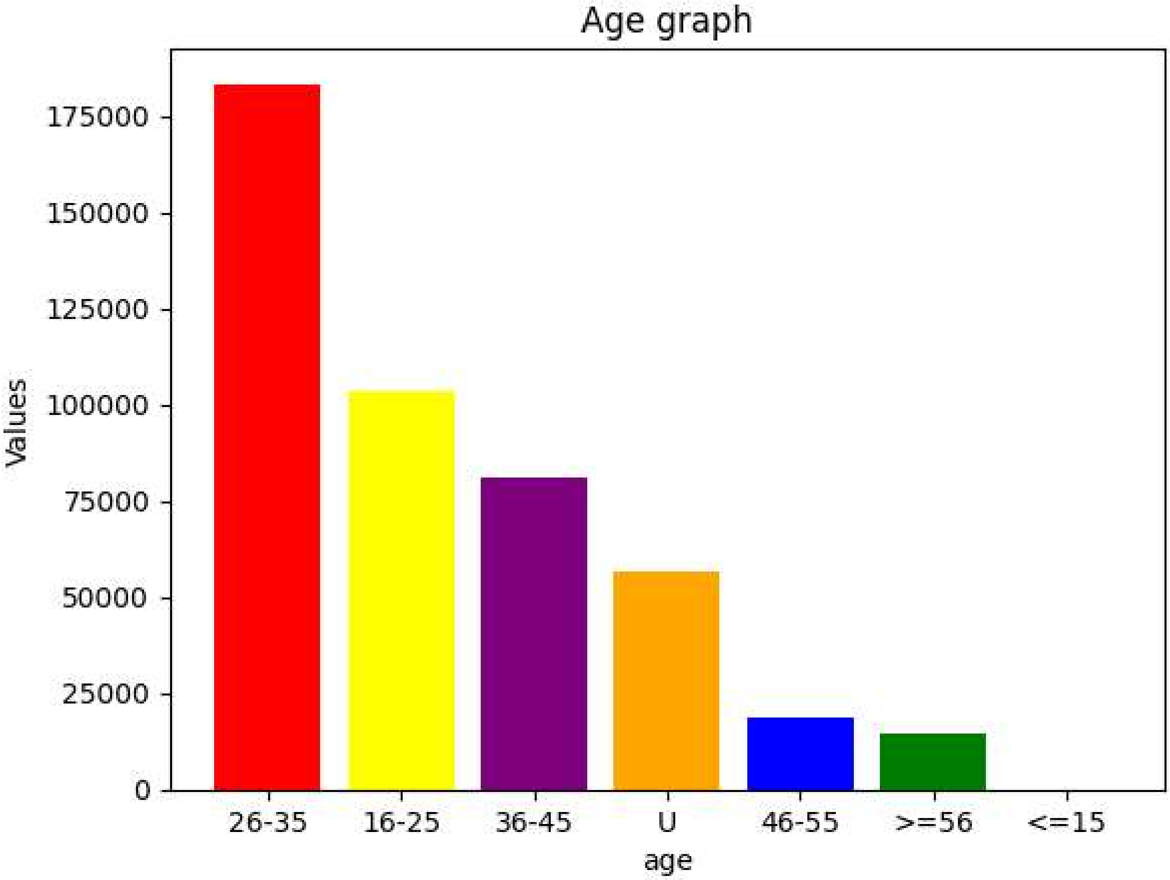
In [14]:
```
age_dis.values
```

Out[14]:
```
array([183239, 103306,  81076,  56457,  18679,  14517,     24],
      dtype=int64)
```

In [15]:
```
# Sample data
custom_color = ['Red','Yellow','Purple', 'Orange','Blue','Green']
age = age_dis.index
values = age_dis.values

# Create a bar graph
plt.bar(categories, values, color= custom_color)

# Add labels and title
plt.xlabel('age')
plt.ylabel('Values')
plt.title('Age graph')

# Show the plot
plt.show()
```

Out[15]:
```
<BarContainer object of 7 artists>
```

```
Out[15]: Text(0.5, 0, 'age')

Out[15]: Text(0, 0.5, 'Values')

Out[15]: Text(0.5, 1.0, 'Age graph')
```



5. Explore a single interval variable

```
In [16]: orders.columns
```

```
Out[16]: Index(['order_ID', 'user_ID', 'sku_ID', 'order_date', 'order_time', 'quantity',
                'type', 'promise', 'original_unit_price', 'final_unit_price',
                'direct_discount_per_unit', 'quantity_discount_per_unit',
                'bundle_discount_per_unit', 'coupon_discount_per_unit', 'gift_item',
                'dc_ori', 'dc_des'],
               dtype='object')
```

```
In [17]: orders['original_unit_price'].describe()
```

```
Out[17]: count    549989.000000
         mean        102.813542
         std          95.035563
         min           0.000000
         25%          59.000000
         50%          79.000000
         75%         139.000000
         max       12158.000000
         Name: original_unit_price, dtype: float64
```

In [18]:
```
import numpy as np
```

In [19]:
```
np.var(orders['original_unit_price'])
```

Out[19]:
```
9031.741770278562
```

In [20]:
```
np.percentile(orders['original_unit_price'], 90) # Finding quantile or percentile
```

Out[20]:
```
240.0
```

In [21]:
```
# Find the records with the maximum price
# The max price is an outlier
orders['original_unit_price'] == 12158
```

Out[21]:
```
0         False
1         False
2         False
3         False
4         False
          ...  
549984    False
549985    False
549986    False
549987    False
549988    False
Name: original_unit_price, Length: 549989, dtype: bool
```

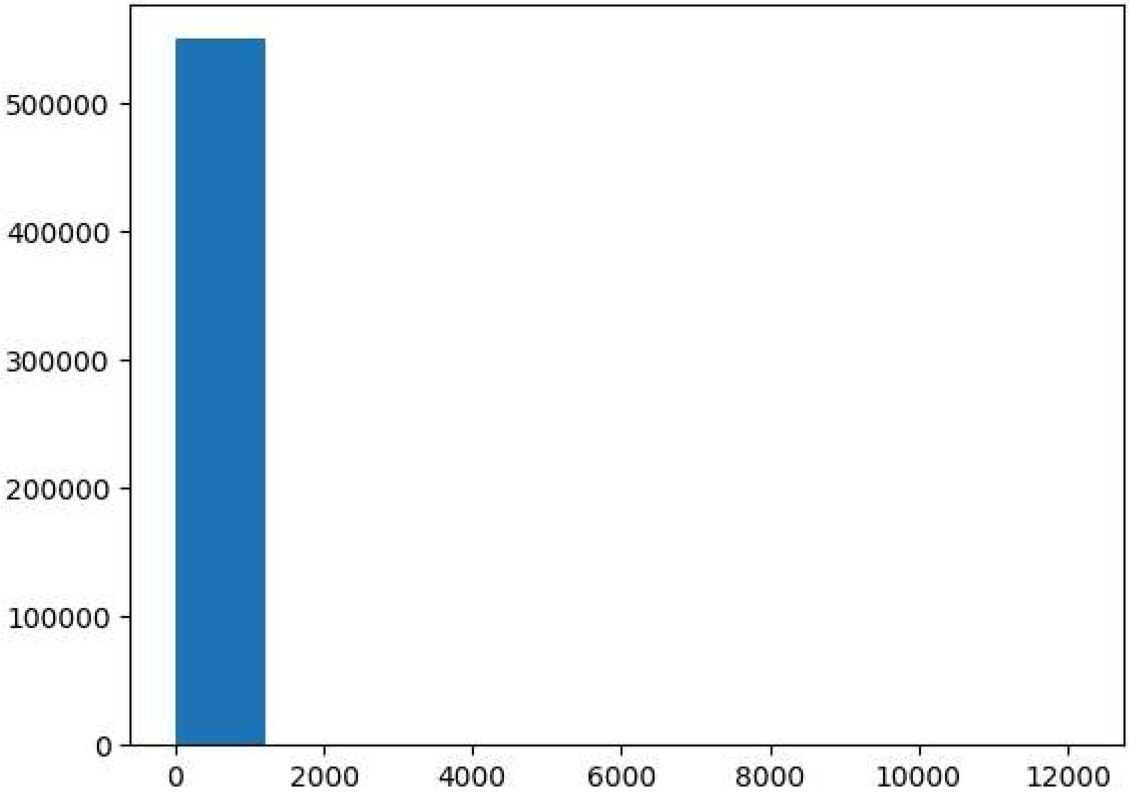
In [22]:
```
orders[orders['original_unit_price'] == 12158]
```

Out[22]:

	order_ID	user_ID	sku_ID	order_date	order_time	quantity	type	promi
52720	a63239c796	b695af3c92	1904d943c0	2018-03-03	2018-03-03 14:56:01.0	1	2	

In [23]:
```
# A histogram to explore the distribution of the interval variable
# original price
plt.hist(orders['original_unit_price'])
```

Out[23]:
```
(array([5.49971e+05, 5.00000e+00, 2.00000e+00, 1.00000e+00, 2.00000e+00,
        2.00000e+00, 2.00000e+00, 0.00000e+00, 1.00000e+00, 3.00000e+00]),
 array([    0. ,  1215.8,  2431.6,  3647.4,  4863.2,  6079. ,  7294.8,
         8510.6,  9726.4, 10942.2, 12158. ]),
 <BarContainer object of 10 artists>)
```



In [38]:
```
orders = orders[orders['original_unit_price'] != 12158]
```

In [40]:
```
orders['original_unit_price'].quantile(0.99)
```

Out[40]:
```
336.0
```

In [43]:
```
orders['original_unit_price'].mean()
```

Out[43]:
```
102.7916229960169
```

In [42]:
```
orders['original_unit_price'].std()
```

Out[42]:
```
93.63512178749312
```

In [31]:
```
out_up = orders['original_unit_price'].mean() + 3 * orders['original_unit_price'].s
```

In [32]:
```
print(out_up)
```

```
383.69698835849624
```

In [33]:
```
#Delete the outliers using slicing
normal_price = orders[orders['original_unit_price']<388]
```

In [34]:
```
normal_price
```

Out[34]:

	order_ID	user_ID	sku_ID	order_date	order_time	quantity	type	pro
0	d0cf5cc6db	0abe9ef2ce	581d5b54c1	2018-03-01	2018-03-01 17:14:25.0	1	2	
1	7444318d01	33a9e56257	067b673f2b	2018-03-01	2018-03-01 11:10:40.0	1	1	
2	f973b01694	4ea3cf408f	623d0a582a	2018-03-01	2018-03-01 09:13:26.0	1	1	
3	8c1cec8d4b	b87cb736cb	fc5289b139	2018-03-01	2018-03-01 21:29:50.0	1	1	
4	d43a33c38a	4829223b6f	623d0a582a	2018-03-01	2018-03-01 19:13:37.0	1	1	
...	...	...	...	...	...	...	...	
549984	3ad06b9fbe	a27b3ed4d4	a9109972d1	2018-03-31	2018-03-31 01:22:47.0	1	2	
549985	c9d77a7ed0	18f92434cd	7f53769d3f	2018-03-31	2018-03-31 08:55:57.0	1	1	
549986	b9ad79338f	b5caf8a580	8dc4a01dec	2018-03-31	2018-03-31 13:31:01.0	1	1	
549987	be3a9414b1	20ba6655f3	2dd6b818ec	2018-03-31	2018-03-31 12:51:18.0	1	2	
549988	02d31f05c9	f260895cbe	10d369ef96	2018-03-31	2018-03-31 18:21:16.0	1	2	

546883 rows × 17 columns

In [35]:

```
orders.shape
#before slicing
```

Out[35]:

```
(549988, 17)
```

In [44]:

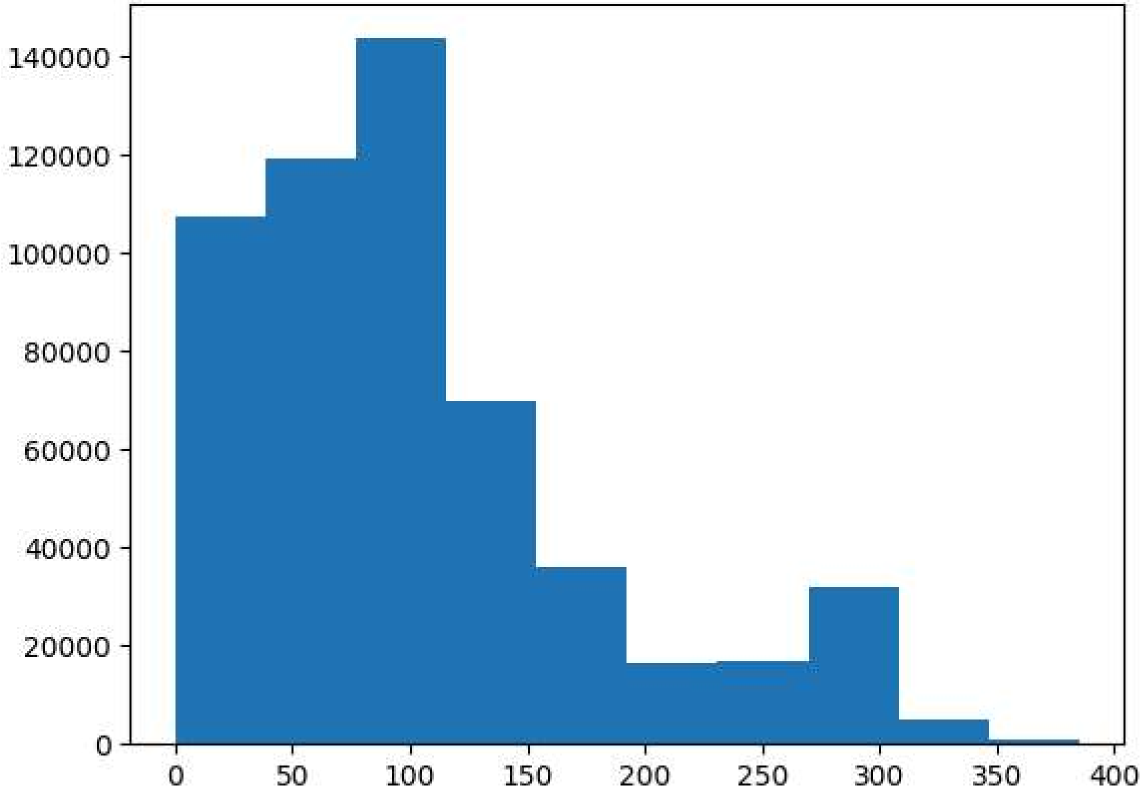
```
normal_price.shape
#after slicing
```

Out[44]:

```
(546883, 17)
```

```
In [45]: plt.hist(normal_price['original_unit_price'])
```

```
Out[45]: (array([107222., 119175., 143434.,  69727.,  35921.,  16537.,  16975.,
                  31718.,   5087.,   1087.]),
          array([  0. ,  38.5,  77. , 115.5, 154. , 192.5, 231. , 269.5, 308. ,
                 346.5, 385. ]),
          <BarContainer object of 10 artists>)
```



Exploration of two interval variables

```
In [ ]:
```

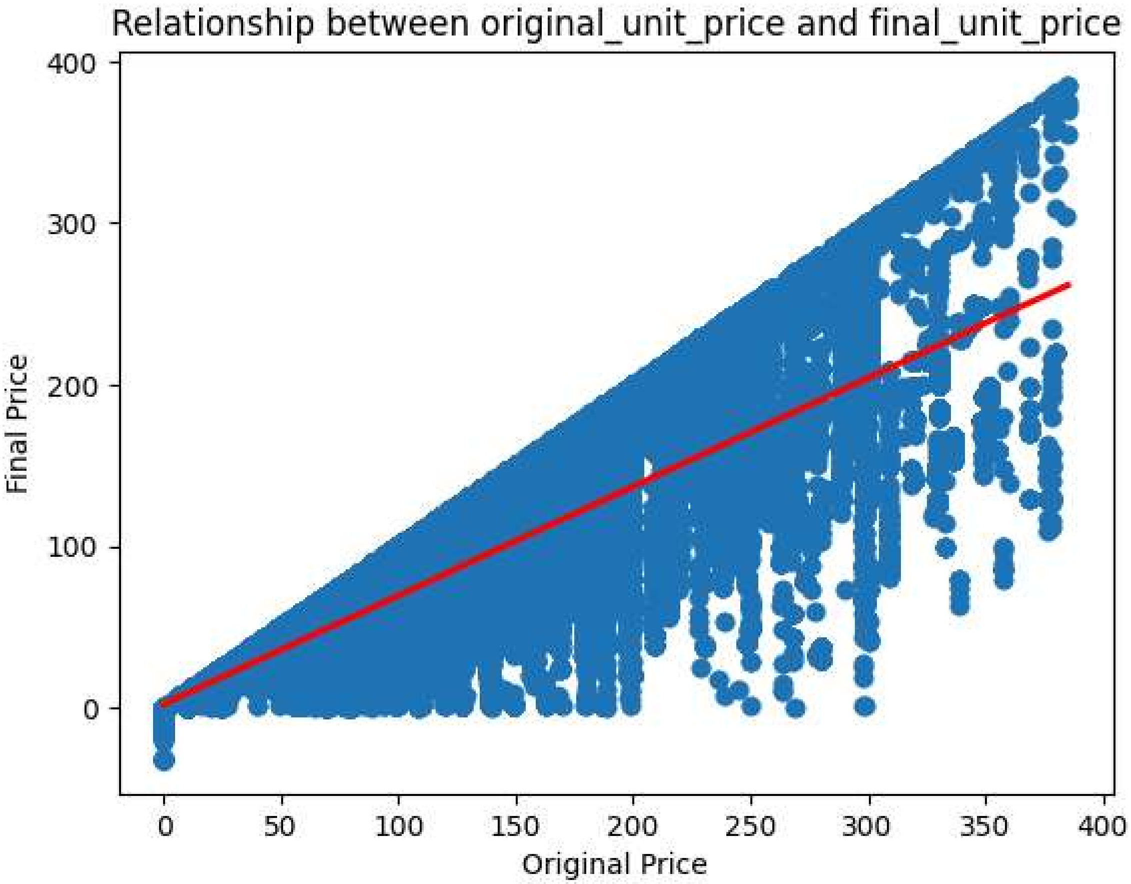
```
In [46]: normal_price.columns
```

```
Out[46]: Index(['order_ID', 'user_ID', 'sku_ID', 'order_date', 'order_time', 'quantity',
                'type', 'promise', 'original_unit_price', 'final_unit_price',
                'direct_discount_per_unit', 'quantity_discount_per_unit',
                'bundle_discount_per_unit', 'coupon_discount_per_unit', 'gift_item',
                'dc_ori', 'dc_des'],
               dtype='object')
```

```
In [51]: import seaborn as sns
```

```
In [52]: #Make a scatterplot between 'original_unit_price', 'final_unit_price',
         plt.scatter(normal_price.original_unit_price,normal_price.final_unit_price)
         sns.regplot(x = 'original_unit_price', y = 'final_unit_price', data = normal_price,
         plt.title('Relationship between original_unit_price and final_unit_price')
plt.xlabel('Original Price')
plt.ylabel('Final Price')
```

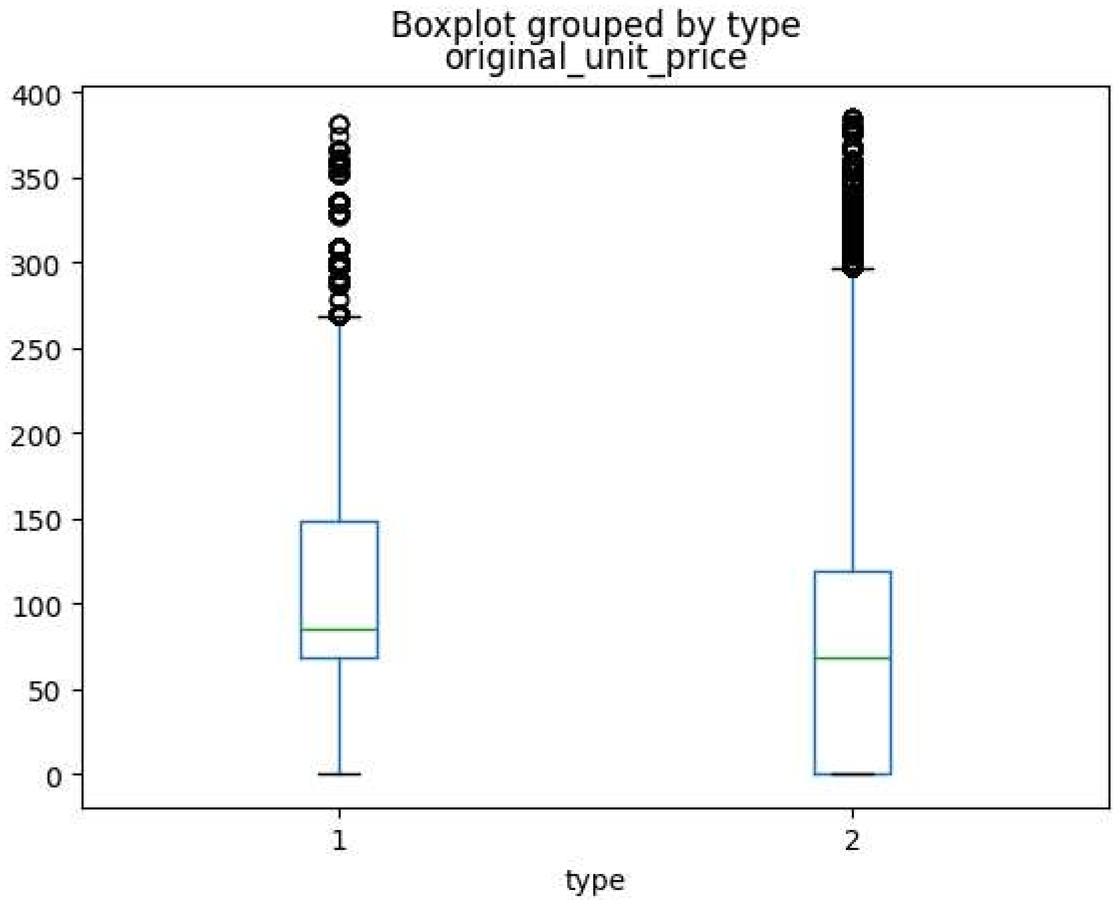
```
Out[52]: <matplotlib.collections.PathCollection at 0x209960d9b20>
Out[52]: <Axes: xlabel='original_unit_price', ylabel='final_unit_price'>
Out[52]: Text(0.5, 1.0, 'Relationship between original_unit_price and final_unit_price')
Out[52]: Text(0.5, 0, 'Original Price')
Out[52]: Text(0, 0.5, 'Final Price')
```



Exploring relationship between interval variable and categorical variable

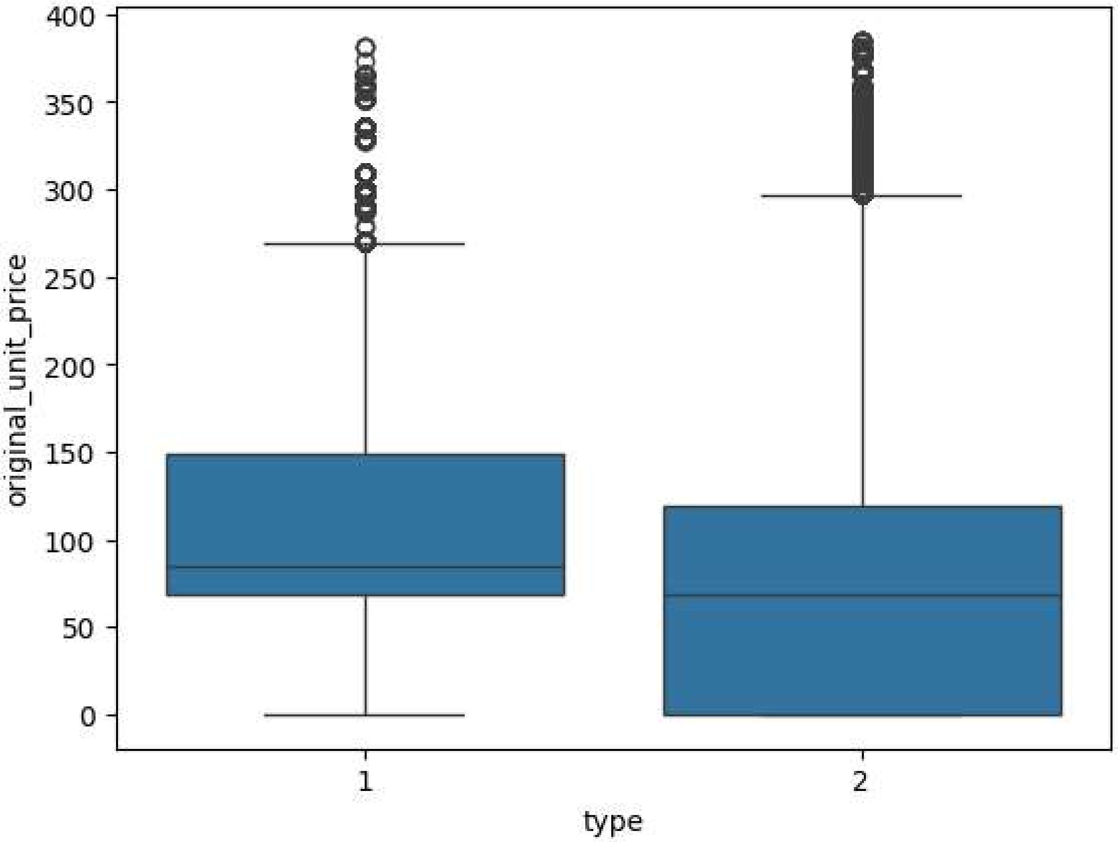
```
In [53]: normal_price.boxplot(column = 'original_unit_price', by = 'type', grid = False)
         #price difference between products sold by jd and other third party
```

```
Out[53]: <Axes: title={'center': 'original_unit_price'}, xlabel='type'>
```



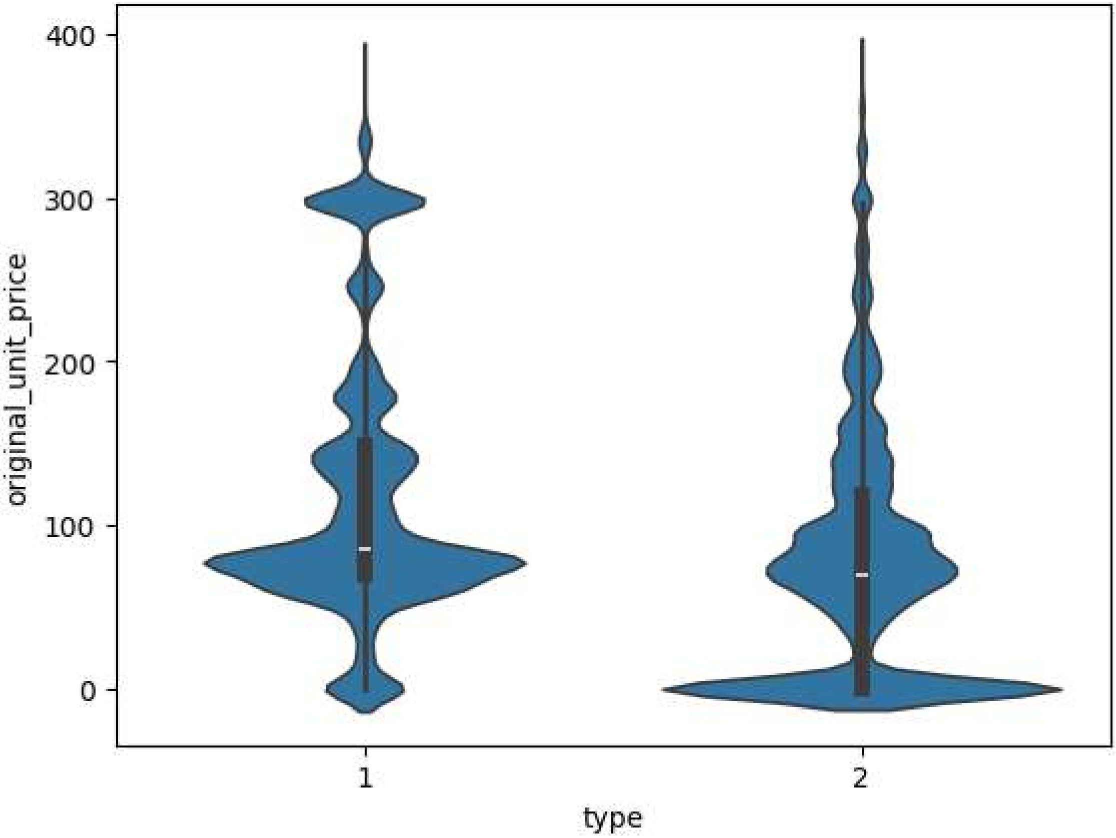
```
In [54]: sns.boxplot(x = 'type', y = 'original_unit_price', data = normal_price)
```

```
Out[54]: <Axes: xlabel='type', ylabel='original_unit_price'>
```



```
In [55]: sns.violinplot(x = 'type', y = 'original_unit_price', data = normal_price)
```

```
Out[55]: <Axes: xlabel='type', ylabel='original_unit_price'>
```


original_unit_price
type