Question 1

a

In [1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [7]:
```
df = pd.read_csv("/Users/kychen/Downloads/2019 Winter Data Science Intern
df.head()
```

Out[7]:

	order_id	shop_id	user_id	order_amount	total_items	payment_method	created_at
0	1	53	746	224	2	cash	2017-03-13 12:36:56
1	2	92	925	90	1	cash	2017-03-03 17:38:52
2	3	44	861	144	1	cash	2017-03-14 4:23:56
3	4	18	935	156	1	credit_card	2017-03-26 12:43:37
4	5	18	883	156	1	credit_card	2017-03-01 4:35:11

In [16]:
```
#Check missing value
df.isnull().sum()
```

Out[16]:
```
order_id          0
shop_id           0
user_id           0
order_amount      0
total_items       0
payment_method    0
created_at        0
avg_item_value    0
dtype: int64
```

In [17]:
```
#Check duplcate rows
dup = df[df.duplicated()]
dup
```

Out[17]:

order_id	shop_id	user_id	order_amount	total_items	payment_method	created_at	avg_item_val

There is no missing value or duplicated rows in the dataset.

In [8]:
```
df.shape
```

Out[8]:
```
(5000, 7)
```

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   order_id        5000 non-null   int64 
 1   shop_id         5000 non-null   int64 
 2   user_id         5000 non-null   int64 
 3   order_amount    5000 non-null   int64 
 4   total_items     5000 non-null   int64 
 5   payment_method  5000 non-null   object
 6   created_at      5000 non-null   object
dtypes: int64(5), object(2)
memory usage: 273.6+ KB
```

In [10]:
```
df.describe()
```

Out[10]:

	order_id	shop_id	user_id	order_amount	total_items
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.00000
mean	2500.500000	50.078800	849.092400	3145.128000	8.78720
std	1443.520003	29.006118	87.798982	41282.539349	116.32032
min	1.000000	1.000000	607.000000	90.000000	1.00000
25%	1250.750000	24.000000	775.000000	163.000000	1.00000
50%	2500.500000	50.000000	849.000000	284.000000	2.00000
75%	3750.250000	75.000000	925.000000	390.000000	3.00000
max	5000.000000	100.000000	999.000000	704000.000000	2000.00000

In [21]:

```
#Define a new feature named avg_item_value
df['avg_item_value'] = df['order_amount'] * 1. / df['total_items']
df.head()
```

Out[21]:

	order_id	shop_id	user_id	order_amount	total_items	payment_method	created_at	avg_item_va
0	1	53	746	224	2	cash	2017-03-13 12:36:56	1
1	2	92	925	90	1	cash	2017-03-03 17:38:52	
2	3	44	861	144	1	cash	2017-03-14 4:23:56	1
3	4	18	935	156	1	credit_card	2017-03-26 12:43:37	1
4	5	18	883	156	1	credit_card	2017-03-01 4:35:11	1

In [86]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   order_id        5000 non-null   int64  
 1   shop_id         5000 non-null   int64  
 2   user_id         5000 non-null   int64  
 3   order_amount    5000 non-null   int64  
 4   total_items     5000 non-null   int64  
 5   payment_method  5000 non-null   object 
 6   created_at      5000 non-null   object 
 7   avg_item_value  5000 non-null   float64
dtypes: float64(1), int64(5), object(2)
memory usage: 312.6+ KB
```

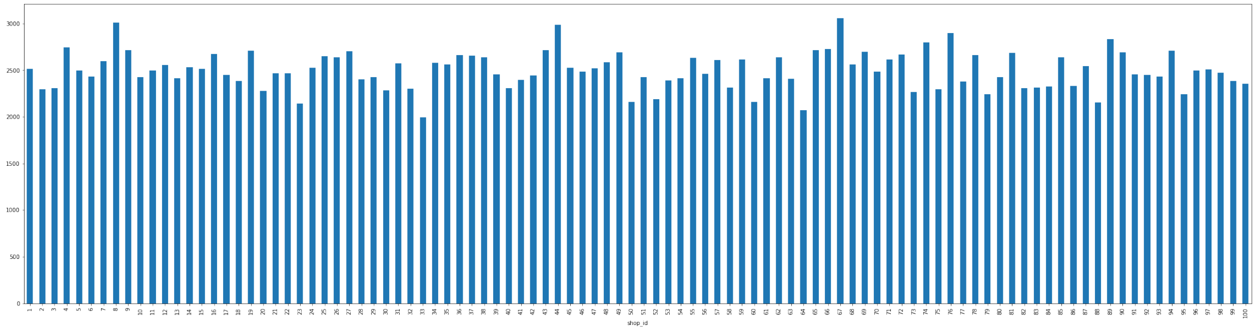
In [87]:
```
df.describe()
```

Out[87]:

	order_id	shop_id	user_id	order_amount	total_items	avg_item_value
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.00000	5000.000000
mean	2500.500000	50.078800	849.092400	3145.128000	8.78720	387.742800
std	1443.520003	29.006118	87.798982	41282.539349	116.32032	2441.963725
min	1.000000	1.000000	607.000000	90.000000	1.00000	90.000000
25%	1250.750000	24.000000	775.000000	163.000000	1.00000	133.000000
50%	2500.500000	50.000000	849.000000	284.000000	2.00000	153.000000
75%	3750.250000	75.000000	925.000000	390.000000	3.00000	169.000000
max	5000.000000	100.000000	999.000000	704000.000000	2000.00000	25725.000000

In [34]:
```
df_groupshop = df.groupby('shop_id')['order_id'].mean()
df_groupshop.plot.bar(figsize=(40,10))
```

Out[34]:
```
<AxesSubplot:xlabel='shop_id'>
```



The visualization of shop_id with order shows an approximate normal distribution.

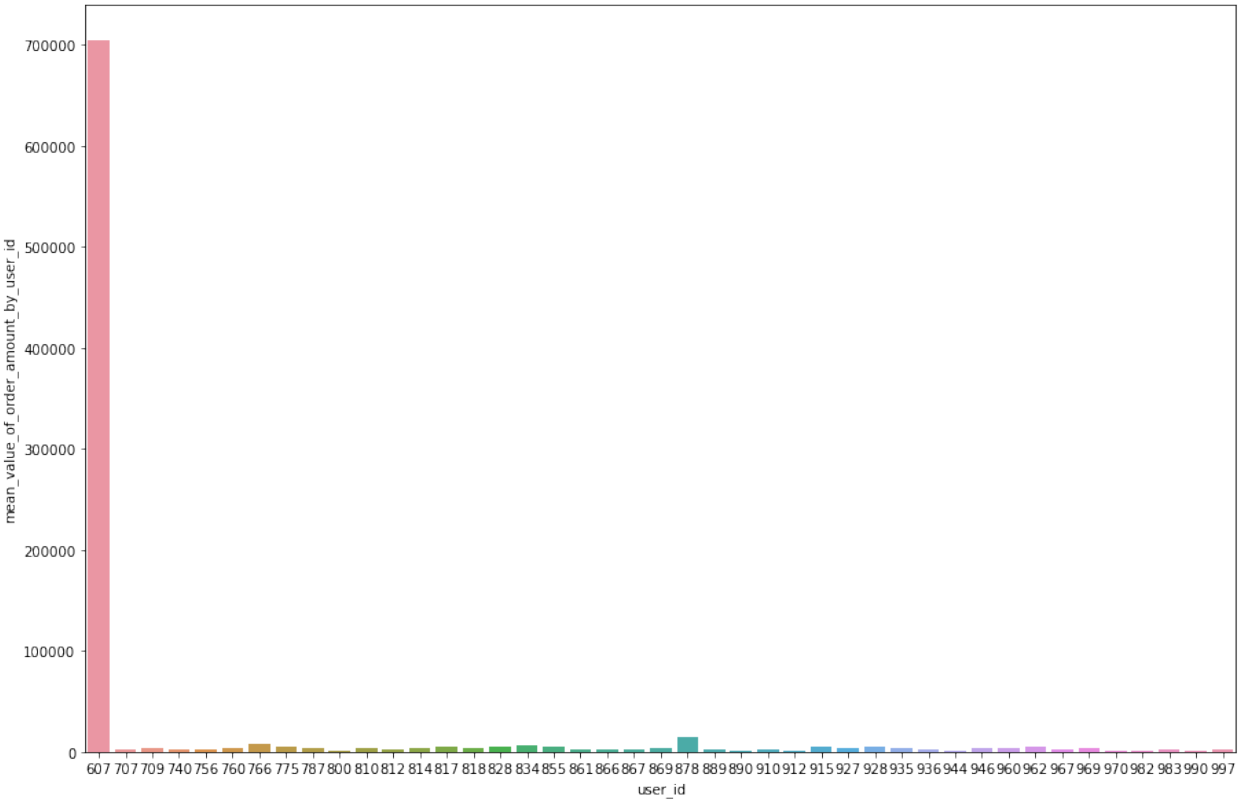
In [85]:
```
usergroup=pd.DataFrame({'mean_value_of_order_amount_by_user_id': df.group
usergroup.head()
```

Out[85]:

	user_id	mean_value_of_order_amount_by_user_id
0	607	704000.000000
1	700	299.375000
2	701	397.076923
3	702	406.615385
4	703	380.687500

In [55]:
```
subset_order_by_userID = usergroup[usergroup['mean_value_of_order_amount_
fig = plt.figure(figsize=(15,10))
sns.barplot(x=subset_order_by_userID['user_id'],y=subset_order_by_userID
```

Out[55]:
```
<AxesSubplot:xlabel='user_id', ylabel='mean_value_of_order_amount_by_us
er_id'>
```



From the dataframe and visualization above,user 607 has suspicious behavior. User 607 pay 704,000 dollars with 2000 items each trade ,which is a extreme outlier.

In [88]:

```
#Check the record of user 607
df[df['user_id']==607]
```

Out[88]:

	order_id	shop_id	user_id	order_amount	total_items	payment_method	created_at	avg_iter
15	16	42	607	704000	2000	credit_card	2017-03-07 4:00:00	
60	61	42	607	704000	2000	credit_card	2017-03-04 4:00:00	
520	521	42	607	704000	2000	credit_card	2017-03-02 4:00:00	
1104	1105	42	607	704000	2000	credit_card	2017-03-24 4:00:00	
1362	1363	42	607	704000	2000	credit_card	2017-03-15 4:00:00	
1436	1437	42	607	704000	2000	credit_card	2017-03-11 4:00:00	
1562	1563	42	607	704000	2000	credit_card	2017-03-19 4:00:00	
1602	1603	42	607	704000	2000	credit_card	2017-03-17 4:00:00	
2153	2154	42	607	704000	2000	credit_card	2017-03-12 4:00:00	
2297	2298	42	607	704000	2000	credit_card	2017-03-07 4:00:00	
2835	2836	42	607	704000	2000	credit_card	2017-03-28 4:00:00	
2969	2970	42	607	704000	2000	credit_card	2017-03-28 4:00:00	
3332	3333	42	607	704000	2000	credit_card	2017-03-24 4:00:00	
4056	4057	42	607	704000	2000	credit_card	2017-03-28 4:00:00	
4646	4647	42	607	704000	2000	credit_card	2017-03-02 4:00:00	
4868	4869	42	607	704000	2000	credit_card	2017-03-22 4:00:00	
4882	4883	42	607	704000	2000	credit_card	2017-03-25 4:00:00	

Here are the consutomer record of User 607.This user purchases from the same shop and same time with different dates.All of the purchase takes place at 4 AM. Additionally, User 607 purchases 2000 items every time. Each order worths 704,000 dollars.

From all the informtion above, we may conclude that User 607 had fraud behaviors and the purchase might be conducted by computer programming script, instead of human being.

```
Besides User 607, user 878 and 766 also had relatively large number of
order amount among all users. Here we will do some EDA for user 878 and
766.
```

In [89]:

```
df[df['user_id']==878]
```

Out[89]:

	order_id	shop_id	user_id	order_amount	total_items	payment_method	created_at	avg_iter
691	692	78	878	154350	6	debit	2017-03-27 22:51:43	
818	819	60	878	354	2	debit	2017-03-27 12:42:01	
927	928	2	878	94	1	credit_card	2017-03-10 18:09:05	
1575	1576	47	878	290	2	cash	2017-03-07 22:06:51	
1833	1834	74	878	153	1	credit_card	2017-03-06 17:33:21	
2011	2012	87	878	298	2	cash	2017-03-04 14:14:35	
3474	3475	20	878	254	2	cash	2017-03-17 3:43:03	
3647	3648	98	878	266	2	cash	2017-03-06 1:49:57	
4106	4107	26	878	176	1	debit	2017-03-20 4:32:18	
4215	4216	80	878	435	3	debit	2017-03-05 3:07:32	
4670	4671	98	878	266	2	debit	2017-03-30 0:22:20	

In [58]:

```
df[df['user_id']==766]
```

Out[58]:

	order_id	shop_id	user_id	order_amount	total_items	payment_method	created_at	avg_iter
1132	1133	81	766	354	2	cash	2017-03-07 15:43:40	
1278	1279	70	766	346	2	credit_card	2017-03-15 20:05:22	
1464	1465	4	766	128	1	debit	2017-03-22 23:36:53	
1691	1692	84	766	459	3	debit	2017-03-09 1:39:10	
2115	2116	63	766	544	4	debit	2017-03-14 17:28:31	
2169	2170	4	766	256	2	debit	2017-03-20 23:10:22	
2936	2937	69	766	262	2	cash	2017-03-17 22:22:59	
3422	3423	20	766	381	3	cash	2017-03-30 17:36:08	
3724	3725	78	766	77175	3	credit_card	2017-03-16 14:13:26	
3977	3978	55	766	171	1	debit	2017-03-10 9:54:00	

In the row of order_id 692 and 1133, user 766 and 878 shows abnormal one-time purchase in the same shop(shop 78). It is almost impossible for a sneak store to own such expensive product.(with avg value of 25,725 dollars)

In [63]:
```
usergroup2=pd.DataFrame({'mean_value_of_order_amount_by_shop_id': df.grou
usergroup2
```

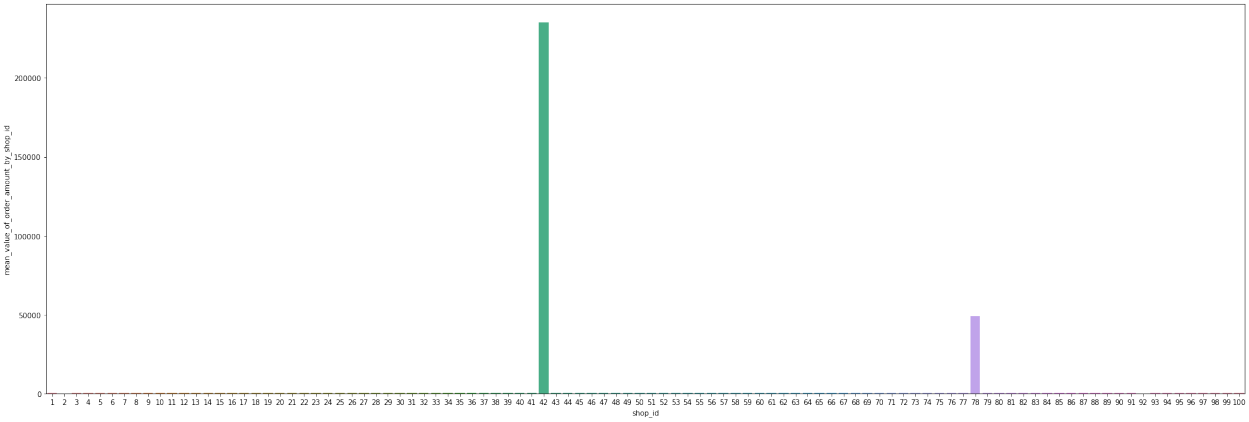
Out[63]:

	shop_id	mean_value_of_order_amount_by_shop_id
0	1	308.818182
1	2	174.327273
2	3	305.250000
3	4	258.509804
4	5	290.311111
...	...	...
95	96	330.000000
96	97	324.000000
97	98	245.362069
98	99	339.444444
99	100	213.675000

100 rows × 2 columns

In [67]:
```
fig = plt.figure(figsize=(30,10))
sns.barplot(x=usergroup2['shop_id'],y=usergroup2['mean_value_of_order_amo
```

Out[67]:
```
<AxesSubplot:xlabel='shop_id', ylabel='mean_value_of_order_amount_by_sh
op_id'>
```



From the visualization above, shop 42 and shop 78 shows the abnormal order amount values. We will check the purchase record of shop 42 and shop 78

In [91]:

```
#Check the record of shop 42
df[df['shop_id']==42]

df_shopid_42=df[df['shop_id']==42]
df_shopid_42.head()
```

Out[91]:

	order_id	shop_id	user_id	order_amount	total_items	payment_method	created_at	avg_item
15	16	42	607	704000	2000	credit_card	2017-03-07 4:00:00	
40	41	42	793	352	1	credit_card	2017-03-24 14:15:41	
60	61	42	607	704000	2000	credit_card	2017-03-04 4:00:00	
308	309	42	770	352	1	credit_card	2017-03-11 18:14:39	
409	410	42	904	704	2	credit_card	2017-03-04 14:32:58	

In [92]:

```
#Check the record of shop 78
df[df['shop_id']==78]
df_shopid_78=df[df['shop_id']==78]
df_shopid_78.head()
```

Out[92]:

	order_id	shop_id	user_id	order_amount	total_items	payment_method	created_at	avg_item
160	161	78	990	25725	1	credit_card	2017-03-12 5:56:57	2
490	491	78	936	51450	2	debit	2017-03-26 17:08:19	2
493	494	78	983	51450	2	cash	2017-03-16 21:39:35	2
511	512	78	967	51450	2	cash	2017-03-09 7:23:14	2
617	618	78	760	51450	2	cash	2017-03-18 11:18:42	2

For shop 78, the fraud behavior shows by the extreme high value of average item price. The price is abnormal for products in a sneaker store.

For shop 42, the fraud behavior shows by the extreme high in total numbers and order amount in some of the orders. by checking the time in the row, all of the abnormal consuming behaviors are conducted by the computer programming script, because of the same set-up time (4 am in the morning) and credit card payment.

In [ ]:

```
#Clean the data by removing outliers(user)
```

In [93]:

```
#remove user 607, user 766, user 878, shop 78,shop 42


new_df = df[df['user_id']!=607]
new_df = df[df['user_id']!=766]
new_df = df[df['user_id']!=878]


new_df = new_df[new_df['shop_id']!=78]
new_df=new_df[new_df['shop_id']!=42]
new_df.head()
```

Out[93]:

	order_id	shop_id	user_id	order_amount	total_items	payment_method	created_at	avg_item_va
0	1	53	746	224	2	cash	2017-03-13 12:36:56	1
1	2	92	925	90	1	cash	2017-03-03 17:38:52	9
2	3	44	861	144	1	cash	2017-03-14 4:23:56	14
3	4	18	935	156	1	credit_card	2017-03-26 12:43:37	15
4	5	18	883	156	1	credit_card	2017-03-01 4:35:11	15

In [74]:

```
df.describe()
```

Out[74]:

	order_id	shop_id	user_id	order_amount	total_items	avg_item_value
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.00000	5000.000000
mean	2500.500000	50.078800	849.092400	3145.128000	8.78720	387.742800
std	1443.520003	29.006118	87.798982	41282.539349	116.32032	2441.963725
min	1.000000	1.000000	607.000000	90.000000	1.00000	90.000000
25%	1250.750000	24.000000	775.000000	163.000000	1.00000	133.000000
50%	2500.500000	50.000000	849.000000	284.000000	2.00000	153.000000
75%	3750.250000	75.000000	925.000000	390.000000	3.00000	169.000000
max	5000.000000	100.000000	999.000000	704000.000000	2000.00000	25725.000000

In [73]:

```
new_df.describe()
```

Out[73]:

	order_id	shop_id	user_id	order_amount	total_items	avg_item_value
count	4893.000000	4893.000000	4893.000000	4893.000000	4893.000000	4893.000000
mean	2499.116493	49.881872	849.801349	300.240752	1.996117	150.414878
std	1444.328268	29.144331	86.967382	156.031481	0.983191	23.850875
min	1.000000	1.000000	700.000000	90.000000	1.000000	90.000000
25%	1246.000000	24.000000	775.000000	163.000000	1.000000	132.000000
50%	2499.000000	50.000000	849.000000	284.000000	2.000000	153.000000
75%	3750.000000	74.000000	925.000000	387.000000	3.000000	166.000000
max	5000.000000	100.000000	999.000000	1086.000000	8.000000	201.000000

On Shopify, we have exactly 100 sneaker shops, and each of these shops sells only one model of shoe. We want to do some analysis of the average order value (AOV). When we look at orders data over a 30 day window, we naively calculate an AOV of $3145.13. Given that we know these shops are selling sneakers, a relatively affordable item, something seems wrong with our analysis.

Think about what could be going wrong with our calculation. Think about a better way to evaluate this data. What metric would you report for this dataset? What is its value?

In [12]:

```
wrong_aov = round(sum(df["order_amount"]) / len(df), 2)
print(wrong_aov)
```

```
3145.13
```

The reason why the average order value (AOV) is $3145.13 is that we divide the "sum of order amount" with the "number of orders". In fact, the customer may have purchased more than one item in a single order, thus causing the value of AOV to be wrong.

The mean value of order amount is 3145.13,which is extremely too high for sneaker stores.

There are outliers in order_amount variable. Additionally, we may also need to consider about how this anamoly consuming behavior take place and investigate whether some users or stores are associated with fraud or abuse.

The median is 284,which is reasonable.

By the code and visualization that showed above, we detect fraud and abuse behaviors for some users and shops.

User 607 has suspicious behavior. User 607 pay 704,000 dollars with 2000 items each trade,which is a extreme outlier. The purchase might be conducted by computer programming script instead of human being because every purchase shows in the same time(4 AM in the morning).

User 766 and 878 shows abnormal one-time purchase in the same shop(shop 78). It is almost impossible for a sneak store to own such expensive product.(with avg value of 25,725 dollars)

For shop 78, the fraud behavior shows by the extreme high value of average item price. The price is abnormal for products in a sneaker store.

For shop 42, the fraud behavior shows by the extreme high in total numbers and order amount in some of the orders. by checking the time in the row, all of the abnormal consuming behaviors are conducted by the computer programming script, because of the same set-up time (4 AM in the morning) and credit card payment.

After removing outliers, the new mean order amount is 300.24 dollars, which makes more sense rather than 3145.13.

Before removing outliers, median is better than mean in this case because the dataset contains abnormally extreme value. Mean value cannot decribe the sampling distribution of consumption level. After removing outliers, both of them might work well.

Type *Markdown* and LaTeX: α^2

b & c

For this dataset, I would report the average item value (avg_item_value) for each store.

Using this index, we can estimate the average price of goods in each store, so as to better understand the distribution of consumption level in these stores.

In [76]:

```
new_df.describe()
```

Out[76]:

	order_id	shop_id	user_id	order_amount	total_items	avg_item_value
count	4893.000000	4893.000000	4893.000000	4893.000000	4893.000000	4893.000000
mean	2499.116493	49.881872	849.801349	300.240752	1.996117	150.414878
std	1444.328268	29.144331	86.967382	156.031481	0.983191	23.850875
min	1.000000	1.000000	700.000000	90.000000	1.000000	90.000000
25%	1246.000000	24.000000	775.000000	163.000000	1.000000	132.000000
50%	2499.000000	50.000000	849.000000	284.000000	2.000000	153.000000
75%	3750.000000	74.000000	925.000000	387.000000	3.000000	166.000000
max	5000.000000	100.000000	999.000000	1086.000000	8.000000	201.000000

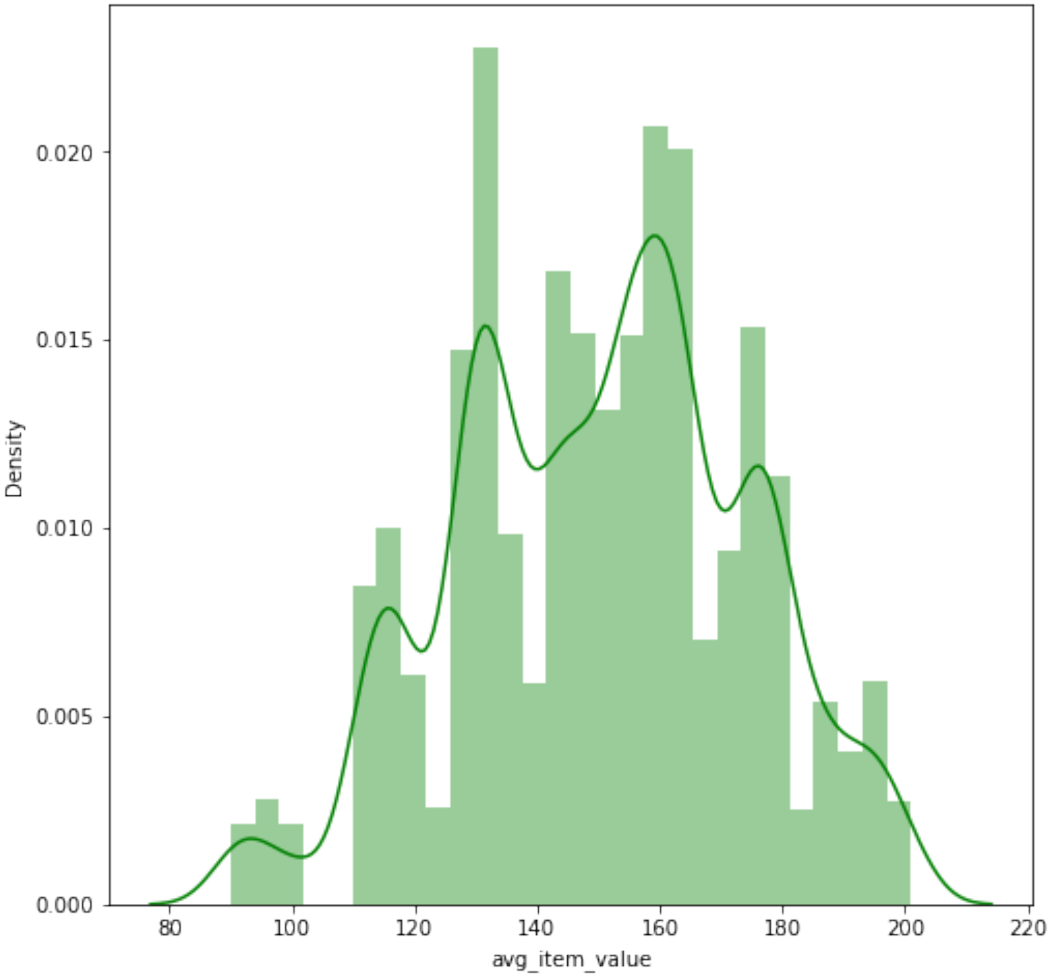
In [78]:

```
fig = plt.figure(figsize=(8, 8))
sns.distplot(new_df['avg_item_value'], color='green', kde=True)
```

```
/Users/kychen/opt/anaconda3/lib/python3.9/site-packages/seaborn/distrib
utions.py:2619: FutureWarning: `distplot` is a deprecated function and
will be removed in a future version. Please adapt your code to use eith
er `displot` (a figure-level function with similar flexibility) or `his
tplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[78]:

```
<AxesSubplot:xlabel='avg_item_value', ylabel='Density'>
```



In [80]:

```
new_df2 = new_df.drop(new_df[new_df['avg_item_value'] > 357.92].index)
```

In [81]:

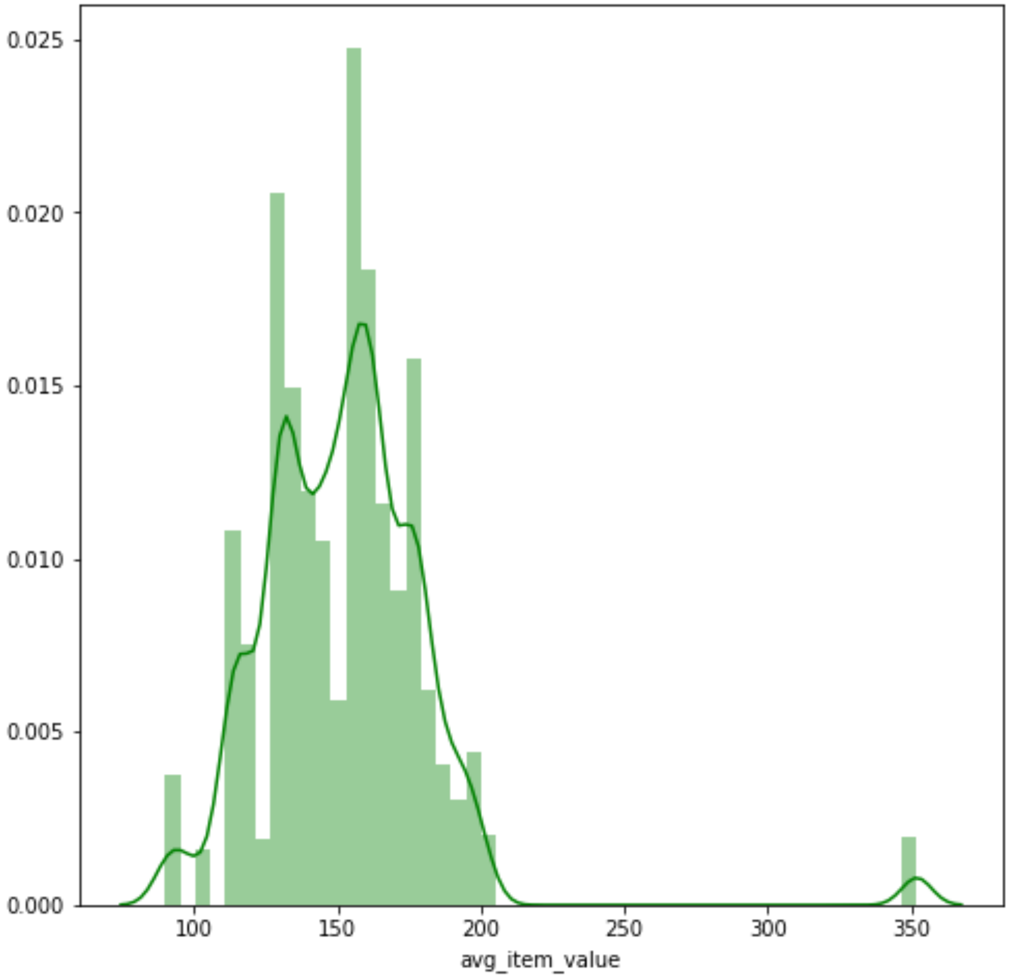
```
new_df2.describe()
```

Out[81]:

	order_id	shop_id	user_id	order_amount	total_items	avg_item_value
count	4893.000000	4893.000000	4893.000000	4893.000000	4893.000000	4893.000000
mean	2499.116493	49.881872	849.801349	300.240752	1.996117	150.414878
std	1444.328268	29.144331	86.967382	156.031481	0.983191	23.850875
min	1.000000	1.000000	700.000000	90.000000	1.000000	90.000000
25%	1246.000000	24.000000	775.000000	163.000000	1.000000	132.000000
50%	2499.000000	50.000000	849.000000	284.000000	2.000000	153.000000
75%	3750.000000	74.000000	925.000000	387.000000	3.000000	166.000000
max	5000.000000	100.000000	999.000000	1086.000000	8.000000	201.000000

```
In [94]: fig = plt.figure(figsize=(8, 8))
         sns.distplot(new_df2['avg_item_value'], color='green', kde=True)

Out[94]: <matplotlib.axes._subplots.AxesSubplot at 0x7fecbbbe8b10>
```



```
In [82]: new_df2 = new_df2.drop(new_df2[new_df2['avg_item_value'] > 240].index)
```

In [83]:

```
new_df2.describe()
```

Out[83]:

	order_id	shop_id	user_id	order_amount	total_items	avg_item_value
count	4893.000000	4893.000000	4893.000000	4893.000000	4893.000000	4893.000000
mean	2499.116493	49.881872	849.801349	300.240752	1.996117	150.414878
std	1444.328268	29.144331	86.967382	156.031481	0.983191	23.850875
min	1.000000	1.000000	700.000000	90.000000	1.000000	90.000000
25%	1246.000000	24.000000	775.000000	163.000000	1.000000	132.000000
50%	2499.000000	50.000000	849.000000	284.000000	2.000000	153.000000
75%	3750.000000	74.000000	925.000000	387.000000	3.000000	166.000000
max	5000.000000	100.000000	999.000000	1086.000000	8.000000	201.000000

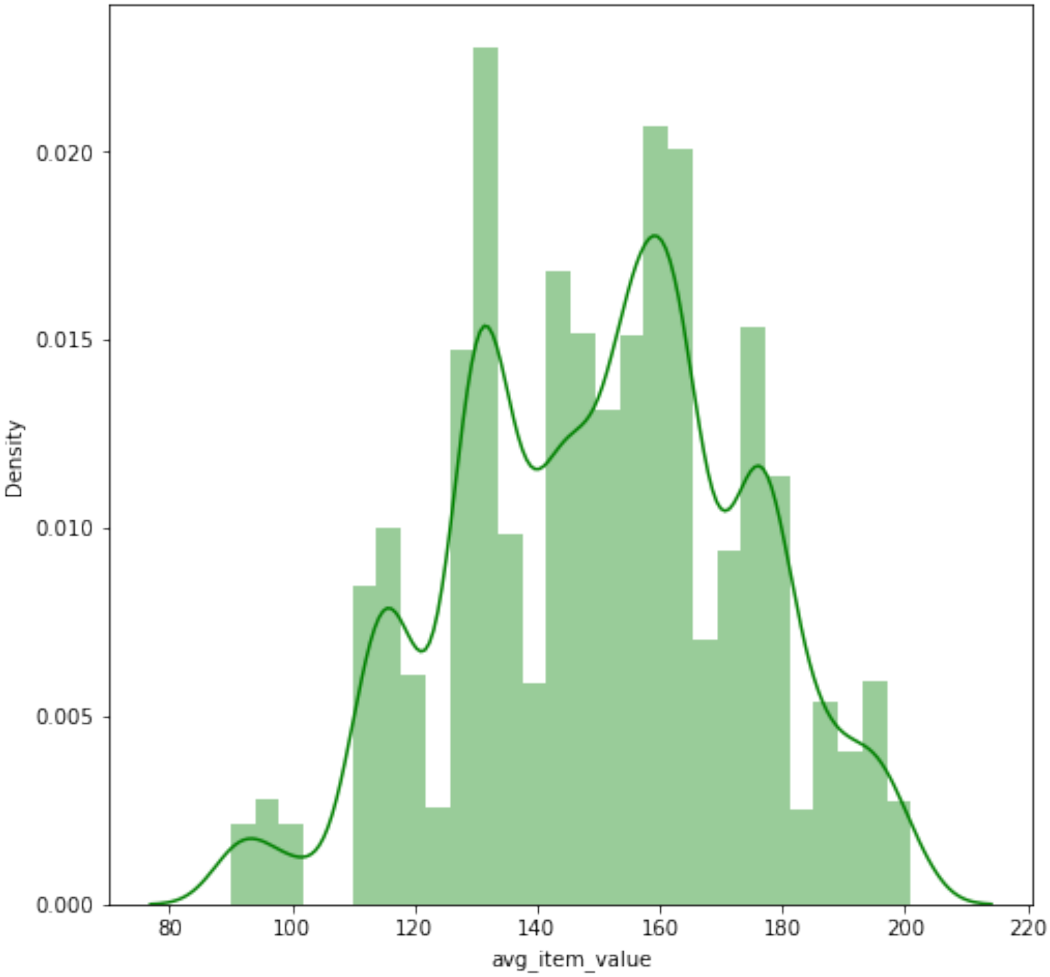
In [84]:

```
fig = plt.figure(figsize=(8, 8))
sns.distplot(new_df2['avg_item_value'], color='green', kde=True)
```

```
/Users/kychen/opt/anaconda3/lib/python3.9/site-packages/seaborn/distrib
utions.py:2619: FutureWarning: `distplot` is a deprecated function and
will be removed in a future version. Please adapt your code to use eith
er `displot` (a figure-level function with similar flexibility) or `his
tplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[84]:

```
<AxesSubplot:xlabel='avg_item_value', ylabel='Density'>
```



Question 2

How many orders were shipped by Speedy Express in total? What is the last name of the employee with the most orders? What product was ordered the most by customers in Germany?

a

```
input:
SELECT COUNT(Shippers.ShipperName)
FROM Orders
LEFT JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID
WHERE Shippers.ShipperName = "Speedy Express"
```

```
answer:
54
```

b

```
input:
SELECT Employees.LastName, COUNT(OrderID) AS Employee_Order_Number
FROM Orders
LEFT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
GROUP BY LastName
ORDER BY COUNT(OrderID) DESC;
```

```
answer:
Peacock
```

c

```
input:
SELECT COUNT(Orders.OrderID) as Orders_Number, Products.ProductName
FROM Orders
JOIN Customers ON Orders.CustomerID == Customers.CustomerID
JOIN OrderDetails ON Orders.OrderID == OrderDetails.OrderID
JOIN Products ON OrderDetails.ProductID = Products.ProductID
WHERE Customer.country == 'Germany'
GROUP BY Products.ProductName
ORDER BY COUNT(Orders.OrderID) DESC;
```

```
answer:
Gorgonzola Telino
```