Shopify_2022_datascience

January 18, 2022

Question 1: Given some sample data, write a program to answer the following: click here to access the required data set

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.

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

[1]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

Since we a getting $3145.13 as AOV, we want to do a bit of mannual inspection of the data set (first 100 rows) first:

[2]:
```
df = pd.read_csv('2019 Winter Data Science Intern Challenge Data Set - Sheet1.
 ↪csv')
print(df.shape)
```

```
(5000, 7)
```

[3]:
```
pd.set_option('display.max_rows', 100)
df.head(100)
```

[3]:
```
    order_id  shop_id  user_id  order_amount  total_items payment_method  \
0          1       53      746           224            2           cash
1          2       92      925            90            1           cash
2          3       44      861           144            1           cash
3          4       18      935           156            1    credit_card
4          5       18      883           156            1    credit_card
5          6       58      882           138            1    credit_card
6          7       87      915           149            1           cash
7          8       22      761           292            2           cash
8          9       64      914           266            2          debit
```

9	10	52	788	146	1	credit_card
10	11	66	848	322	2	credit_card
11	12	40	983	322	2	debit
12	13	54	799	266	2	credit_card
13	14	100	709	111	1	cash
14	15	87	849	447	3	credit_card
15	16	42	607	704000	2000	credit_card
16	17	17	731	176	1	cash
17	18	28	752	164	1	credit_card
18	19	83	761	258	2	cash
19	20	63	898	408	3	credit_card
20	21	66	987	322	2	cash
21	22	97	789	486	3	credit_card
22	23	88	985	704	4	credit_card
23	24	75	964	256	2	credit_card
24	25	73	917	495	3	cash
25	26	82	848	177	1	cash
26	27	47	882	145	1	cash
27	28	53	942	112	1	credit_card
28	29	40	944	322	2	cash
29	30	59	790	178	1	credit_card
30	31	76	857	310	2	cash
31	32	57	839	294	2	debit
32	33	76	712	465	3	credit_card
33	34	7	800	224	2	credit_card
34	35	34	704	244	2	debit
35	36	61	781	316	2	cash
36	37	84	979	459	3	credit_card
37	38	66	961	322	2	credit_card
38	39	10	921	148	1	credit_card
39	40	61	756	316	2	credit_card
40	41	42	793	352	1	credit_card
41	42	1	847	316	2	debit
42	43	18	934	624	4	debit
43	44	21	792	284	2	credit_card
44	45	99	759	195	1	credit_card
45	46	29	969	652	4	credit_card
46	47	33	954	346	2	cash
47	48	52	791	438	3	cash
48	49	3	714	296	2	debit
49	50	64	768	399	3	debit
50	51	58	984	276	2	debit
51	52	81	959	531	3	cash
52	53	30	968	459	3	cash
53	54	33	842	692	4	debit
54	55	79	823	181	1	cash
55	56	51	851	561	3	credit_card

56	57	53	739	560	5	credit_card
57	58	51	759	187	1	cash
58	59	47	837	145	1	credit_card
59	60	80	908	145	1	debit
60	61	42	607	704000	2000	credit_card
61	62	60	720	531	3	debit
62	63	86	981	260	2	debit
63	64	91	962	160	1	debit
64	65	72	887	480	3	credit_card
65	66	7	988	112	1	debit
66	67	66	743	322	2	cash
67	68	21	812	284	2	debit
68	69	86	994	390	3	credit_card
69	70	58	876	138	1	debit
70	71	11	725	184	1	credit_card
71	72	34	813	122	1	credit_card
72	73	86	960	130	1	debit
73	74	14	968	116	1	cash
74	75	5	862	142	1	credit_card
75	76	84	744	459	3	cash
76	77	20	975	127	1	cash
77	78	41	985	354	3	cash
78	79	8	806	132	1	credit_card
79	80	20	838	254	2	credit_card
80	81	52	979	584	4	credit_card
81	82	28	868	328	2	debit
82	83	48	857	234	2	cash
83	84	100	817	111	1	cash
84	85	41	707	118	1	debit
85	86	53	908	224	2	credit_card
86	87	56	711	234	2	credit_card
87	88	64	713	399	3	credit_card
88	89	55	998	513	3	credit_card
89	90	28	705	164	1	cash
90	91	50	744	193	1	debit
91	92	3	835	296	2	credit_card
92	93	22	856	146	1	credit_card
93	94	58	810	276	2	debit
94	95	87	961	149	1	cash
95	96	35	975	328	2	credit_card
96	97	70	758	346	2	cash
97	98	92	850	180	2	cash
98	99	79	741	181	1	debit
99	100	18	752	780	5	cash

	created_at
0	2017-03-13 12:36:56

1	2017-03-03 17:38:52
2	2017-03-14 4:23:56
3	2017-03-26 12:43:37
4	2017-03-01 4:35:11
5	2017-03-14 15:25:01
6	2017-03-01 21:37:57
7	2017-03-08 2:05:38
8	2017-03-17 20:56:50
9	2017-03-30 21:08:26
10	2017-03-26 23:36:40
11	2017-03-12 17:58:30
12	2017-03-16 14:15:34
13	2017-03-22 2:39:49
14	2017-03-10 11:23:18
15	2017-03-07 4:00:00
16	2017-03-21 4:23:38
17	2017-03-21 12:09:07
18	2017-03-17 13:18:47
19	2017-03-29 15:11:52
20	2017-03-30 20:11:59
21	2017-03-04 15:44:00
22	2017-03-22 1:19:41
23	2017-03-12 3:07:42
24	2017-03-03 13:01:03
25	2017-03-25 21:35:12
26	2017-03-22 7:38:43
27	2017-03-17 9:41:53
28	2017-03-05 2:12:53
29	2017-03-04 22:49:28
30	2017-03-23 21:34:39
31	2017-03-19 5:31:45
32	2017-03-10 23:54:10
33	2017-03-03 5:31:15
34	2017-03-13 0:00:44
35	2017-03-08 15:57:42
36	2017-03-05 22:44:34
37	2017-03-14 6:04:23
38	2017-03-06 5:51:08
39	2017-03-07 17:03:29
40	2017-03-24 14:15:41
41	2017-03-20 14:58:02
42	2017-03-21 6:59:10
43	2017-03-08 4:16:53
44	2017-03-02 8:13:24
45	2017-03-04 8:58:23
46	2017-03-25 14:18:33
47	2017-03-05 15:50:51

48	2017-03-20 16:48:03
49	2017-03-02 17:26:07
50	2017-03-23 7:05:31
51	2017-03-25 22:58:11
52	2017-03-28 8:15:20
53	2017-03-08 7:36:34
54	2017-03-25 4:47:17
55	2017-03-13 5:05:25
56	2017-03-18 8:45:21
57	2017-03-30 22:34:47
58	2017-03-13 6:35:07
59	2017-03-27 9:03:35
60	2017-03-04 4:00:00
61	2017-03-21 18:08:34
62	2017-03-24 6:11:03
63	2017-03-01 19:05:37
64	2017-03-02 0:15:05
65	2017-03-29 10:22:34
66	2017-03-01 10:40:40
67	2017-03-03 15:52:53
68	2017-03-21 22:05:37
69	2017-03-04 10:38:49
70	2017-03-14 17:12:18
71	2017-03-20 17:43:28
72	2017-03-04 14:15:34
73	2017-03-22 0:14:48
74	2017-03-26 9:14:45
75	2017-03-06 0:39:27
76	2017-03-19 13:05:29
77	2017-03-22 13:41:25
78	2017-03-27 3:39:39
79	2017-03-03 14:00:25
80	2017-03-11 14:30:09
81	2017-03-04 6:53:29
82	2017-03-11 18:14:37
83	2017-03-12 22:10:47
84	2017-03-30 12:05:30
85	2017-03-27 20:46:35
86	2017-03-17 2:08:47
87	2017-03-04 12:15:16
88	2017-03-09 13:22:15
89	2017-03-11 18:13:28
90	2017-03-02 6:36:55
91	2017-03-09 4:01:51
92	2017-03-24 11:47:02
93	2017-03-07 4:27:06
94	2017-03-21 3:20:50

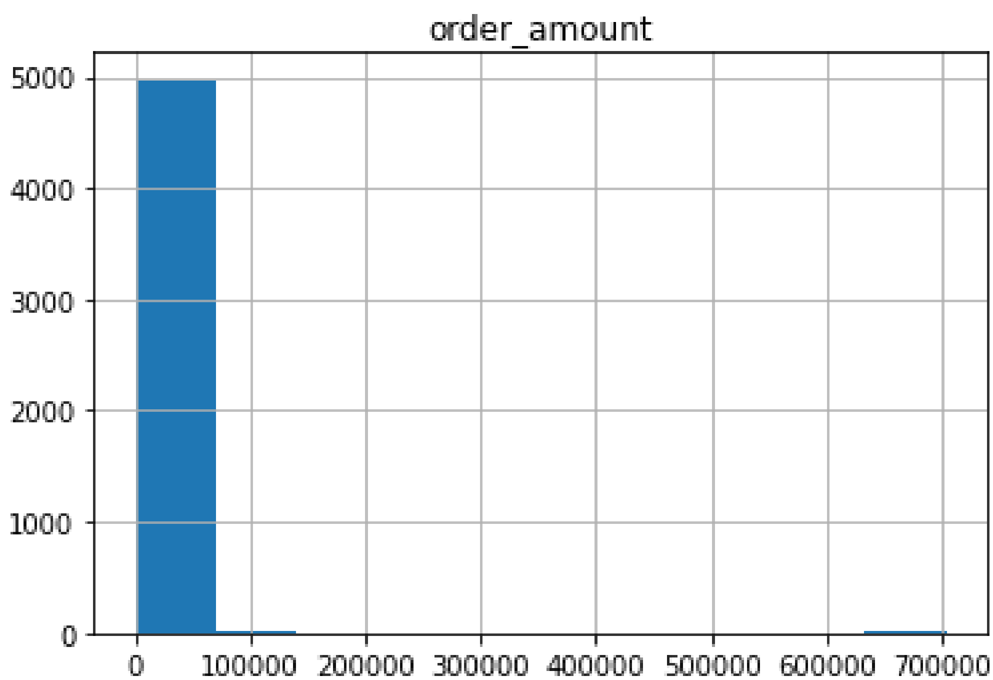
```
95   2017-03-01 6:27:42
96  2017-03-07 14:57:54
97  2017-03-22 19:21:26
98   2017-03-16 0:16:22
99  2017-03-06 23:41:16
```

As we can see, there's a total number of 5000 orders. And from our manual inspection of fisrt 100 rows of the data set, we see most orders are under $500 but there are some orders with abnormal order_amount. For example, both order 16 and 607 have order_amount $704000 and total_items of 2000 pairs of sneakers.

So my guess is the orders which comes with huge order_amount (since they are ordering a huge amounts of sneakers in 1 order) boosted the our metric (AOV) of the data set. I'll plot a histogram to confirm this.

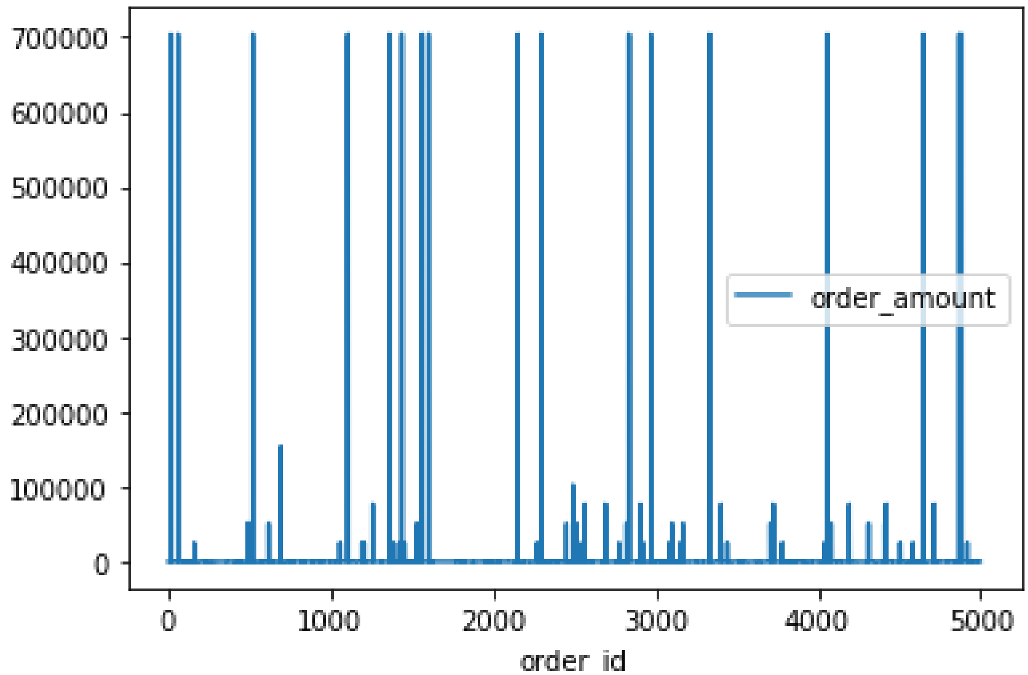
[4]:
```
df.hist(column='order_amount')
```

[4]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f89be2e9410>]],
      dtype=object)
```



[5]:
```
df.plot.line(x='order_id',y='order_amount')
```

[5]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f89be391f90>
```



[6]:
```
df_sorted = df.sort_values(by='order_amount', ascending=False)
df_sorted.head(100)
```

[6]:

	order_id	shop_id	user_id	order_amount	total_items	payment_method \
2153	2154	42	607	704000	2000	credit_card
3332	3333	42	607	704000	2000	credit_card
520	521	42	607	704000	2000	credit_card
1602	1603	42	607	704000	2000	credit_card
60	61	42	607	704000	2000	credit_card
2835	2836	42	607	704000	2000	credit_card
4646	4647	42	607	704000	2000	credit_card
2297	2298	42	607	704000	2000	credit_card
1436	1437	42	607	704000	2000	credit_card
4882	4883	42	607	704000	2000	credit_card
4056	4057	42	607	704000	2000	credit_card
15	16	42	607	704000	2000	credit_card
1104	1105	42	607	704000	2000	credit_card
1562	1563	42	607	704000	2000	credit_card
2969	2970	42	607	704000	2000	credit_card
4868	4869	42	607	704000	2000	credit_card
1362	1363	42	607	704000	2000	credit_card
691	692	78	878	154350	6	debit
2492	2493	78	834	102900	4	debit
3724	3725	78	766	77175	3	credit_card

4420	4421	78	969	77175	3	debit
4192	4193	78	787	77175	3	credit_card
3403	3404	78	928	77175	3	debit
2690	2691	78	962	77175	3	debit
2564	2565	78	915	77175	3	debit
4715	4716	78	818	77175	3	debit
1259	1260	78	775	77175	3	credit_card
2906	2907	78	817	77175	3	debit
3705	3706	78	828	51450	2	credit_card
3101	3102	78	855	51450	2	credit_card
4412	4413	78	756	51450	2	debit
3167	3168	78	927	51450	2	cash
490	491	78	936	51450	2	debit
4079	4080	78	946	51450	2	cash
1529	1530	78	810	51450	2	cash
4311	4312	78	960	51450	2	debit
2818	2819	78	869	51450	2	debit
2821	2822	78	814	51450	2	cash
617	618	78	760	51450	2	cash
2512	2513	78	935	51450	2	debit
511	512	78	967	51450	2	cash
2452	2453	78	709	51450	2	cash
493	494	78	983	51450	2	cash
2495	2496	78	707	51450	2	cash
4040	4041	78	852	25725	1	cash
4918	4919	78	823	25725	1	cash
1056	1057	78	800	25725	1	debit
2922	2923	78	740	25725	1	debit
2270	2271	78	855	25725	1	credit_card
1193	1194	78	944	25725	1	debit
1452	1453	78	812	25725	1	credit_card
3780	3781	78	889	25725	1	cash
4505	4506	78	866	25725	1	debit
2773	2774	78	890	25725	1	cash
3151	3152	78	745	25725	1	credit_card
1384	1385	78	867	25725	1	cash
3085	3086	78	910	25725	1	cash
2548	2549	78	861	25725	1	cash
160	161	78	990	25725	1	credit_card
4584	4585	78	997	25725	1	cash
1419	1420	78	912	25725	1	cash
3440	3441	78	982	25725	1	debit
1204	1205	78	970	25725	1	credit_card
1364	1365	42	797	1760	5	cash
1367	1368	42	926	1408	4	cash
1471	1472	42	907	1408	4	debit
3538	3539	43	830	1086	6	debit

4141	4142	54	733	1064	8	debit
3513	3514	42	726	1056	3	debit
2987	2988	42	819	1056	3	cash
938	939	42	808	1056	3	credit_card
3077	3078	89	754	980	5	debit
2494	2495	50	757	965	5	debit
1563	1564	91	934	960	6	debit
4847	4848	13	993	960	6	cash
2307	2308	61	723	948	6	credit_card
3532	3533	51	828	935	5	cash
1256	1257	6	942	935	5	credit_card
2560	2561	6	845	935	5	credit_card
2039	2040	11	756	920	5	credit_card
3073	3074	90	877	890	5	debit
1150	1151	82	853	885	5	debit
879	880	60	870	885	5	debit
4523	4524	26	995	880	5	credit_card
2032	2033	88	798	880	5	cash
4952	4953	26	786	880	5	cash
1946	1947	33	866	865	5	cash
4958	4959	70	711	865	5	credit_card
2353	2354	27	811	845	5	cash
1962	1963	46	879	830	5	debit
522	523	46	761	830	5	credit_card
2967	2968	46	774	830	5	debit
3865	3866	68	815	816	6	debit
1123	1124	29	911	815	5	credit_card
771	772	19	818	815	5	debit
3927	3928	97	979	810	5	credit_card
2757	2758	66	772	805	5	credit_card
3438	3439	66	842	805	5	credit_card
742	743	12	727	804	4	cash
1764	1765	12	789	804	4	debit

	created_at
2153	2017-03-12 4:00:00
3332	2017-03-24 4:00:00
520	2017-03-02 4:00:00
1602	2017-03-17 4:00:00
60	2017-03-04 4:00:00
2835	2017-03-28 4:00:00
4646	2017-03-02 4:00:00
2297	2017-03-07 4:00:00
1436	2017-03-11 4:00:00
4882	2017-03-25 4:00:00
4056	2017-03-28 4:00:00
15	2017-03-07 4:00:00

1104	2017-03-24 4:00:00
1562	2017-03-19 4:00:00
2969	2017-03-28 4:00:00
4868	2017-03-22 4:00:00
1362	2017-03-15 4:00:00
691	2017-03-27 22:51:43
2492	2017-03-04 4:37:34
3724	2017-03-16 14:13:26
4420	2017-03-09 15:21:35
4192	2017-03-18 9:25:32
3403	2017-03-16 9:45:05
2690	2017-03-22 7:33:25
2564	2017-03-25 1:19:35
4715	2017-03-05 5:10:44
1259	2017-03-27 9:27:20
2906	2017-03-16 3:45:46
3705	2017-03-14 20:43:15
3101	2017-03-21 5:10:34
4412	2017-03-02 4:13:39
3167	2017-03-12 12:23:08
490	2017-03-26 17:08:19
4079	2017-03-20 21:14:00
1529	2017-03-29 7:12:01
4311	2017-03-01 3:02:10
2818	2017-03-17 6:25:51
2821	2017-03-02 17:13:25
617	2017-03-18 11:18:42
2512	2017-03-18 18:57:13
511	2017-03-09 7:23:14
2452	2017-03-27 11:04:04
493	2017-03-16 21:39:35
2495	2017-03-26 4:38:52
4040	2017-03-02 14:31:12
4918	2017-03-15 13:26:46
1056	2017-03-15 10:16:45
2922	2017-03-12 20:10:58
2270	2017-03-14 23:58:22
1193	2017-03-16 16:38:26
1452	2017-03-17 18:09:54
3780	2017-03-11 21:14:50
4505	2017-03-22 22:06:01
2773	2017-03-26 10:36:43
3151	2017-03-18 13:13:07
1384	2017-03-17 16:38:06
3085	2017-03-26 1:59:27
2548	2017-03-17 19:36:00
160	2017-03-12 5:56:57

```
4584  2017-03-25 21:48:44
1419  2017-03-30 12:23:43
3440  2017-03-19 19:02:54
1204  2017-03-17 22:32:21
1364   2017-03-10 6:28:21
1367   2017-03-13 2:38:34
1471  2017-03-12 23:00:22
3538  2017-03-17 19:56:29
4141  2017-03-07 17:05:18
3513  2017-03-24 17:51:05
2987   2017-03-03 9:09:25
938   2017-03-13 23:43:45
3077   2017-03-13 5:27:58
2494   2017-03-04 7:32:45
1563   2017-03-23 8:25:49
4847  2017-03-27 11:00:45
2307  2017-03-26 11:29:37
3532  2017-03-17 16:05:35
1256  2017-03-12 19:49:08
2560  2017-03-16 22:24:30
2039  2017-03-04 10:51:41
3073   2017-03-26 8:08:27
1150  2017-03-24 20:47:47
879   2017-03-27 20:15:11
4523   2017-03-09 8:28:31
2032   2017-03-18 4:24:14
4952   2017-03-17 1:50:18
1946   2017-03-14 5:05:37
4958  2017-03-08 17:22:51
2353   2017-03-13 7:07:39
1962  2017-03-14 17:11:01
522   2017-03-26 19:07:51
2967   2017-03-23 9:22:12
3865   2017-03-11 9:31:50
1123   2017-03-26 0:53:49
771    2017-03-07 8:48:16
3927   2017-03-11 7:37:13
2757   2017-03-14 8:43:29
3438  2017-03-22 17:58:37
742   2017-03-14 16:38:01
1764   2017-03-03 3:10:50
```

As we can see in the histogram, there are only 3 bins where most of the orders are low in order_amount. 2 very small bins are centered around $100000 and $700000. In the line plot, those bins corresponds to the spikes in y-axis. We then sorted the dataset in a descending order in terms of order_amount and displayed the first 100 orders. We can see only a handful of orders have order_amount bigger than $2000 but their order_amount comes in very huge number which influenced calculated AOV.

Therefore, I don't think average order value is a good metric on this dataset. Since from above inspections, we have orders that have both very high order_amount and total_items and orders that have only high order_amount but very low total_items. Also, since we know the 100 shops are only sell one model of shoe, the orders that have only high order_amount but low total_items are likely being errors in recording. I'll print those orders below.

[7]:
```
df_error = df.loc[(df['order_amount'] >= 2000) &
              (df['total_items']<=10)]
df_error.head(100)
```

[7]:

	order_id	shop_id	user_id	order_amount	total_items	payment_method	\
160	161	78	990	25725	1	credit_card	
490	491	78	936	51450	2	debit	
493	494	78	983	51450	2	cash	
511	512	78	967	51450	2	cash	
617	618	78	760	51450	2	cash	
691	692	78	878	154350	6	debit	
1056	1057	78	800	25725	1	debit	
1193	1194	78	944	25725	1	debit	
1204	1205	78	970	25725	1	credit_card	
1259	1260	78	775	77175	3	credit_card	
1384	1385	78	867	25725	1	cash	
1419	1420	78	912	25725	1	cash	
1452	1453	78	812	25725	1	credit_card	
1529	1530	78	810	51450	2	cash	
2270	2271	78	855	25725	1	credit_card	
2452	2453	78	709	51450	2	cash	
2492	2493	78	834	102900	4	debit	
2495	2496	78	707	51450	2	cash	
2512	2513	78	935	51450	2	debit	
2548	2549	78	861	25725	1	cash	
2564	2565	78	915	77175	3	debit	
2690	2691	78	962	77175	3	debit	
2773	2774	78	890	25725	1	cash	
2818	2819	78	869	51450	2	debit	
2821	2822	78	814	51450	2	cash	
2906	2907	78	817	77175	3	debit	
2922	2923	78	740	25725	1	debit	
3085	3086	78	910	25725	1	cash	
3101	3102	78	855	51450	2	credit_card	
3151	3152	78	745	25725	1	credit_card	
3167	3168	78	927	51450	2	cash	
3403	3404	78	928	77175	3	debit	
3440	3441	78	982	25725	1	debit	
3705	3706	78	828	51450	2	credit_card	
3724	3725	78	766	77175	3	credit_card	

3780	3781	78	889	25725	1	cash
4040	4041	78	852	25725	1	cash
4079	4080	78	946	51450	2	cash
4192	4193	78	787	77175	3	credit_card
4311	4312	78	960	51450	2	debit
4412	4413	78	756	51450	2	debit
4420	4421	78	969	77175	3	debit
4505	4506	78	866	25725	1	debit
4584	4585	78	997	25725	1	cash
4715	4716	78	818	77175	3	debit
4918	4919	78	823	25725	1	cash

	created_at
160	2017-03-12 5:56:57
490	2017-03-26 17:08:19
493	2017-03-16 21:39:35
511	2017-03-09 7:23:14
617	2017-03-18 11:18:42
691	2017-03-27 22:51:43
1056	2017-03-15 10:16:45
1193	2017-03-16 16:38:26
1204	2017-03-17 22:32:21
1259	2017-03-27 9:27:20
1384	2017-03-17 16:38:06
1419	2017-03-30 12:23:43
1452	2017-03-17 18:09:54
1529	2017-03-29 7:12:01
2270	2017-03-14 23:58:22
2452	2017-03-27 11:04:04
2492	2017-03-04 4:37:34
2495	2017-03-26 4:38:52
2512	2017-03-18 18:57:13
2548	2017-03-17 19:36:00
2564	2017-03-25 1:19:35
2690	2017-03-22 7:33:25
2773	2017-03-26 10:36:43
2818	2017-03-17 6:25:51
2821	2017-03-02 17:13:25
2906	2017-03-16 3:45:46
2922	2017-03-12 20:10:58
3085	2017-03-26 1:59:27
3101	2017-03-21 5:10:34
3151	2017-03-18 13:13:07
3167	2017-03-12 12:23:08
3403	2017-03-16 9:45:05
3440	2017-03-19 19:02:54
3705	2017-03-14 20:43:15

```
3724   2017-03-16 14:13:26
3780   2017-03-11 21:14:50
4040   2017-03-02 14:31:12
4079   2017-03-20 21:14:00
4192    2017-03-18 9:25:32
4311    2017-03-01 3:02:10
4412    2017-03-02 4:13:39
4420   2017-03-09 15:21:35
4505   2017-03-22 22:06:01
4584   2017-03-25 21:48:44
4715    2017-03-05 5:10:44
4918   2017-03-15 13:26:46
```

In terms of this dataset where every shop only sell one model of the shoe, if we can confirm that the above orders are errors in recording, then average order price of 1 pair of sneakers (sum of total_amount/total_items then divided by total number of orders) would be a better metric. I'll output it below:

[8]:
```
df_corrected = pd.merge(df,df_error, indicator=True, how='outer').
 ↪query('_merge=="left_only"').drop('_merge', axis=1)
average_shoe_price = (df_corrected['order_amount']/df_corrected['total_items']).
 ↪sum()/df_corrected.shape[0]
print(average_shoe_price)
```

```
152.47557529269278
```

The average order price of 1 pair of sneakers on corrected dataset is $152.48. If we can't confirm the error and don't correct the dataset, then the value would be:

[9]:
```
average_shoe_price_uncorrected = (df['order_amount']/df['total_items']).sum()/
 ↪df.shape[0]
print(average_shoe_price_uncorrected)
```

```
387.7428
```

A better metric on the uncorrected dataset would be the standard deviation of the average order price of 1 pair of sneakers as it at least shows us how concentrated the order prices of 1 pair of sneakers are (hopefully we'll see there's a huge deviation and know there's something wrong with the data recodring). I'll output it below:

[10]:
```
std_shoe_price_uncorrected = (df['order_amount']/df['total_items']).std()
print(std_shoe_price_uncorrected)
```

```
2441.963725368451
```

I'll output the std of the corrected set to show there's error in the recording since each shop are only selling the same model thus the order price of 1 pair should be close:

[11]:
```
std_shoe_price = (df_corrected['order_amount']/df_corrected['total_items']).
 ↪std()
print(std_shoe_price)
```

```
31.26021753289636
```

Question 2: For this question you'll need to use SQL. Follow this link to access the data set required for the challenge. Please use queries to answer the following questions. Paste your queries along with your final numerical answers below.

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. SELECT COUNT(OrderID) FROM Orders WHERE ShipperID = (SELECT ShipperID FROM Shippers WHERE ShipperName = 'Speedy Express');

Total number is 54.

b. SELECT LastName FROM Employees WHERE EmployeeID = (SELECT TOP 1 EmployeeID FROM Orders GROUP BY EmployeeID ORDER BY count(EmployeeID) DESC)

Her last name is Peacock.

c. SELECT ProductName FROM Products WHERE ProductID= (SELECT TOP 1 ProductID FROM (SELECT OrderDetails.ProductID, OrderDetails.Quantity FROM OrderDetails LEFT JOIN
(SELECT Orders.OrderID FROM Orders INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID WHERE Country = 'Germany') temp ON OrderDetails.OrderID=temp.OrderID) GROUP BY ProductID ORDER BY sum(Quantity) DESC)

The most ordered product by German customers is Gorgonzola Telino.