2019 Winter Data Science Intern Challenge

May 19, 2022

1 PROJECT 1: Fall 2022 Data Science Intern Challenge

2 NAME: Beauty Ebalehita Onolunose

3 QUESTIONS

3.1 1. Think about what could be going wrong with our calculation. Think about a better way to evaluate this data.

3.2 2. What metric would you report for this dataset?

3.3 3. What is its value?

4 A. Importing libraries

```
In [137]: import pandas as pd
          import numpy as np
          import matplotlib.pyplot as plt
          import csv
          %matplotlib inline
```

5 B. Gathering/Loading the Challenge Data into a DataFrame

```
In [138]: df_Intern = pd.read_csv('2019 Winter Data Science Intern Challenge Data Set - Sheet1.c
          df.head()
```

```
Out[138]:    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

                 created_at
          0  3/13/2017 12:36
          1   3/3/2017 17:38
          2   3/14/2017 4:23
```

```
3  3/26/2017 12:43
4    3/1/2017 4:35
```

6 C. Assessing The Data

6.1 i. Visually

6.2 ii. Programmatically

6.2.1 i. Visual assessment

```
In [9]: df_Intern
```

Out[9]:

	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
...	...	...	...	...	...	...
4970	4971	34	987	244	2	cash
4971	4972	49	854	129	1	debit
4972	4973	58	725	414	3	debit
4973	4974	92	970	360	4	cash

4974	4975	7	925	448	4	credit_card
4975	4976	76	932	155	1	cash
4976	4977	93	763	114	1	credit_card
4977	4978	69	903	131	1	credit_card
4978	4979	2	841	282	3	cash
4979	4980	75	820	128	1	debit
4980	4981	50	942	772	4	credit_card
4981	4982	86	715	260	2	debit
4982	4983	1	906	316	2	credit_card
4983	4984	46	969	332	2	cash
4984	4985	44	966	432	3	credit_card
4985	4986	8	916	396	3	debit
4986	4987	100	731	111	1	cash
4987	4988	11	826	184	1	credit_card
4988	4989	86	877	260	2	cash
4989	4990	11	843	552	3	credit_card
4990	4991	24	860	140	1	cash
4991	4992	61	707	158	1	cash
4992	4993	49	739	258	2	debit
4993	4994	4	834	384	3	debit
4994	4995	12	954	201	1	cash
4995	4996	73	993	330	2	debit
4996	4997	48	789	234	2	cash
4997	4998	56	867	351	3	cash
4998	4999	60	825	354	2	credit_card
4999	5000	44	734	288	2	debit

	created_at
0	3/13/2017 12:36
1	3/3/2017 17:38
2	3/14/2017 4:23
3	3/26/2017 12:43
4	3/1/2017 4:35
5	3/14/2017 15:25
6	3/1/2017 21:37
7	3/8/2017 2:05
8	3/17/2017 20:56
9	3/30/2017 21:08
10	3/26/2017 23:36
11	3/12/2017 17:58
12	3/16/2017 14:15
13	3/22/2017 2:39
14	3/10/2017 11:23
15	3/7/2017 4:00
16	3/21/2017 4:23
17	3/21/2017 12:09
18	3/17/2017 13:18
19	3/29/2017 15:11

```
20    3/30/2017 20:11
21     3/4/2017 15:44
22     3/22/2017 1:19
23     3/12/2017 3:07
24     3/3/2017 13:01
25    3/25/2017 21:35
26     3/22/2017 7:38
27     3/17/2017 9:41
28      3/5/2017 2:12
29     3/4/2017 22:49
...              ...
4970   3/15/2017 8:18
4971  3/28/2017 19:46
4972   3/8/2017 10:42
4973    3/2/2017 2:10
4974  3/24/2017 13:13
4975   3/25/2017 1:15
4976  3/20/2017 17:14
4977  3/13/2017 10:39
4978   3/9/2017 19:13
4979  3/25/2017 12:28
4980   3/23/2017 5:15
4981  3/10/2017 18:53
4982  3/26/2017 15:57
4983  3/23/2017 19:18
4984  3/15/2017 20:12
4985  3/15/2017 14:22
4986   3/30/2017 7:01
4987    3/2/2017 9:35
4988  3/12/2017 19:55
4989   3/2/2017 15:06
4990   3/26/2017 0:32
4991  3/13/2017 18:56
4992  3/24/2017 13:48
4993   3/18/2017 4:12
4994   3/22/2017 0:38
4995  3/30/2017 13:47
4996  3/16/2017 20:36
4997   3/19/2017 5:42
4998  3/16/2017 14:51
4999  3/18/2017 15:48

[5000 rows x 7 columns]
```

```
In [47]: '''From the visual assesment carried out on this data set i can categorically say that
```

```
Out[47]: 'From the visual assesment carried out on this data set i can categorically say that us
```

6.3 ii. Programmatic assessement

```
In [13]: # The .info() function gives a concise summary of the dataset
         df_Intern.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
order_id          5000 non-null int64
shop_id           5000 non-null int64
user_id           5000 non-null int64
order_amount      5000 non-null int64
total_items       5000 non-null int64
payment_method    5000 non-null object
created_at        5000 non-null object
dtypes: int64(5), object(2)
memory usage: 273.5+ KB
```

```
In [14]: '''From the programmatic assessment carried out, noticed that the created_at column is
```

```
Out[14]: 'From the programmatic assessment carried out, noticed that the created_at column is an
```

```
In [16]: df_Intern.describe()
```

Out[16]:

	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

7 D. Data Cleaning

7.1 1. Checking for duplicate data

7.2 2. Drop duplicate data

7.3 3. Check for missing values

7.4 4. If there are missing values, fill the missing values

7.5 6. Changing created_at from string(object) datatype to datetime stamp

Checking for duplicate data

```
In [18]: # Checking for duplicate data in the data set
         sum(df_Intern.duplicated())
```

```
Out[18]: 0

In [39]: '''I also noticed that this method of checking for duplicates is not accurate'''

Out[39]: 'I also noticed that this method of checking for duplicates is not accurate'

In [43]: # Since you don't get to see the entire pieces of data, this also does not do justice t
         df_Intern.duplicated().head()

Out[43]: 0    False
         1    False
         2    False
         3    False
         4    False
         dtype: bool

In [21]: '''There are no duplicate data from the investigation on the above cells'''

Out[21]: 'There are no duplicate data from the investigation on the above cells'
```

Drop duplicate data

```
In [23]: # Drop duplicate data

In [41]: '''Since there are no duplicate data visible in the data set, there won't be need to dr

Out[41]: "Since there are no duplicate data visible in the data set, there won't be need to drop
```

Check for missing values

```
In [42]: # Check for missing values
         df_Intern.isnull().head()

Out[42]:    order_id  shop_id  user_id  order_amount  total_items  payment_method  \
         0     False    False    False         False        False           False
         1     False    False    False         False        False           False
         2     False    False    False         False        False           False
         3     False    False    False         False        False           False
         4     False    False    False         False        False           False

            created_at
         0       False
         1       False
         2       False
         3       False
         4       False
```

Fill up the missing values

```
In [27]: '''There are no missing values, so there also won't be need to fill up the missing valu

Out[27]: "There are no missing values, so there also won't be need to fill up the missing values
```

Changing created_at from str(object) to datetime stamp

```
In [29]: df_Intern['created_at'] = pd.to_datetime(df_Intern['created_at'])

In [33]: # To confirm the change
         df_Intern.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
order_id          5000 non-null int64
shop_id           5000 non-null int64
user_id           5000 non-null int64
order_amount      5000 non-null int64
total_items       5000 non-null int64
payment_method    5000 non-null object
created_at        5000 non-null datetime64[ns]
dtypes: datetime64[ns](1), int64(5), object(1)
memory usage: 273.5+ KB


In [118]: df_Intern

Out[118]:             order_id  shop_id  user_id  ($)order_amount  total_items  \
          0        1        53      746      224                2         cash
          1        2        92      925       90                1         cash
          14      15        87      849      447                3  credit_card
          15      16        42      607   704000             2000  credit_card
          22      23        88      985      704                4  credit_card
          56      57        53      739      560                5  credit_card
          691    692        78      878   154350                6        debit
          4141  4142        54      733     1064                8        debit

                        created_at
          0      3/13/2017 12:36
          1       3/3/2017 17:38
          14     3/10/2017 11:23
          15       3/7/2017 4:00
          22      3/22/2017 1:19
          56      3/18/2017 8:45
          691    3/27/2017 22:51
          4141    3/7/2017 17:05
```

7.5.1 Adding a Dollar sign in order_amount column header

8 E. Exploratory Data Analysis

8.1 Answer To Questions

8.1.1 1. Think about what could be going wrong with our calculation. Think about a better way to evaluate this data.

```
In [139]: select_amount = df_Intern.loc[df_Intern['order_amount'] >= 70000]
          print (select_amount)
```

```
      order_id  shop_id  user_id  order_amount  total_items payment_method  \
15          16       42      607        704000         2000    credit_card
60          61       42      607        704000         2000    credit_card
520        521       42      607        704000         2000    credit_card
691        692       78      878        154350            6          debit
1104      1105       42      607        704000         2000    credit_card
1259      1260       78      775         77175            3    credit_card
1362      1363       42      607        704000         2000    credit_card
1436      1437       42      607        704000         2000    credit_card
1562      1563       42      607        704000         2000    credit_card
1602      1603       42      607        704000         2000    credit_card
2153      2154       42      607        704000         2000    credit_card
2297      2298       42      607        704000         2000    credit_card
2492      2493       78      834        102900            4          debit
2564      2565       78      915         77175            3          debit
2690      2691       78      962         77175            3          debit
2835      2836       42      607        704000         2000    credit_card
2906      2907       78      817         77175            3          debit
2969      2970       42      607        704000         2000    credit_card
3332      3333       42      607        704000         2000    credit_card
3403      3404       78      928         77175            3          debit
3724      3725       78      766         77175            3    credit_card
4056      4057       42      607        704000         2000    credit_card
4192      4193       78      787         77175            3    credit_card
4420      4421       78      969         77175            3          debit
4646      4647       42      607        704000         2000    credit_card
4715      4716       78      818         77175            3          debit
4868      4869       42      607        704000         2000    credit_card
4882      4883       42      607        704000         2000    credit_card

           created_at
15       3/7/2017 4:00
60       3/4/2017 4:00
520      3/2/2017 4:00
691   3/27/2017 22:51
1104   3/24/2017 4:00
1259   3/27/2017 9:27
```

```
1362   3/15/2017 4:00
1436   3/11/2017 4:00
1562   3/19/2017 4:00
1602   3/17/2017 4:00
2153   3/12/2017 4:00
2297    3/7/2017 4:00
2492    3/4/2017 4:37
2564   3/25/2017 1:19
2690   3/22/2017 7:33
2835   3/28/2017 4:00
2906   3/16/2017 3:45
2969   3/28/2017 4:00
3332   3/24/2017 4:00
3403   3/16/2017 9:45
3724  3/16/2017 14:13
4056   3/28/2017 4:00
4192   3/18/2017 9:25
4420   3/9/2017 15:21
4646    3/2/2017 4:00
4715    3/5/2017 5:10
4868   3/22/2017 4:00
4882   3/25/2017 4:00
```

```
In [49]: '''The issue with the calculation is that there is duplicate data from user_id: 607'''
```

```
Out[49]: 'The issue with the calculation is that there is duplicate data from user_id: 607'
```

8.1.2 Drop duplicates

```
In [141]: df_Intern.drop_duplicates(subset = 'order_amount', keep = 'first', inplace = True)
```

8.1.3 Test

```
In [142]: select_amount = df_Intern.loc[df_Intern['order_amount'] >= 70000]
          print (select_amount)
```

```
      order_id  shop_id  user_id  order_amount  total_items payment_method  \
15          16       42      607        704000         2000    credit_card
691        692       78      878        154350            6          debit
1259      1260       78      775         77175            3    credit_card
2492      2493       78      834        102900            4          debit

           created_at
15      3/7/2017 4:00
691   3/27/2017 22:51
1259   3/27/2017 9:27
2492    3/4/2017 4:37
```

2. What metric would you report for this dataset?

```
In [143]: # average order value
          df_Intern['order_amount'].mean()

Out[143]: 4758.2558139534885

In [145]: # # A better view of the metric
          df_Intern.describe()

Out[145]:           order_id     shop_id     user_id   order_amount  total_items
          count   258.000000  258.000000  258.000000     258.000000   258.000000
          mean    710.143411   52.135659  857.081395    4758.255814    10.639535
          std     996.869219   28.610510   89.122556   45562.560952   124.342809
          min       1.000000    1.000000  607.000000      90.000000     1.000000
          25%      86.000000   29.750000  779.500000     238.000000     2.000000
          50%     278.000000   52.000000  857.500000     411.000000     3.000000
          75%     859.250000   78.000000  936.000000     638.000000     4.000000
          max    4750.000000  100.000000  999.000000  704000.000000  2000.000000

In [58]: '''From the cell above you will notice that the count(rows) also reduced from 5000 to 2

Out[58]: 'From the cell above you will notice that the count(rows) also reduced from 5000 to 258

In [147]: df_Intern.head()

Out[147]:    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
          5         6       58      882           138            1    credit_card

                 created_at
          0  3/13/2017 12:36
          1   3/3/2017 17:38
          2   3/14/2017 4:23
          3  3/26/2017 12:43
          5  3/14/2017 15:25
```

8.1.4 3. What is its value?

Its value is $4758.25

9 F. Insights

9.1 The major issue with this data set, is the duplication of data, which gave an incorrect value.

9.2 I also noticed that an object type was the datatype of the created_at column.

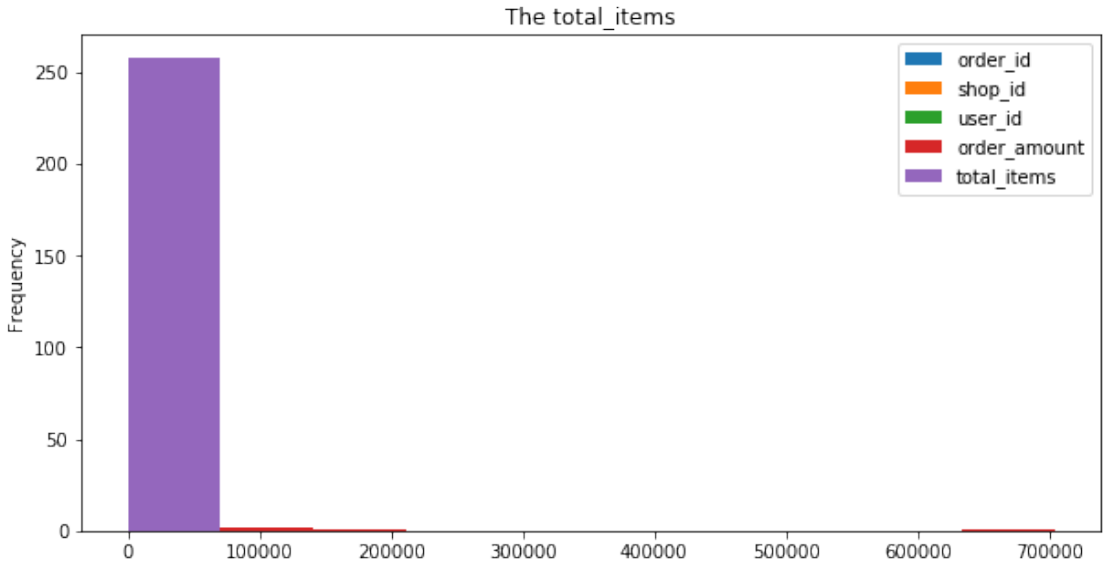
10 G. Visualizations

```
In [156]: df_Intern.plot(kind = 'line', title = 'The order_amount',  figsize = (10, 5))
```

```
Out[156]: <matplotlib.axes._subplots.AxesSubplot at 0x7f55cd47e518>
```

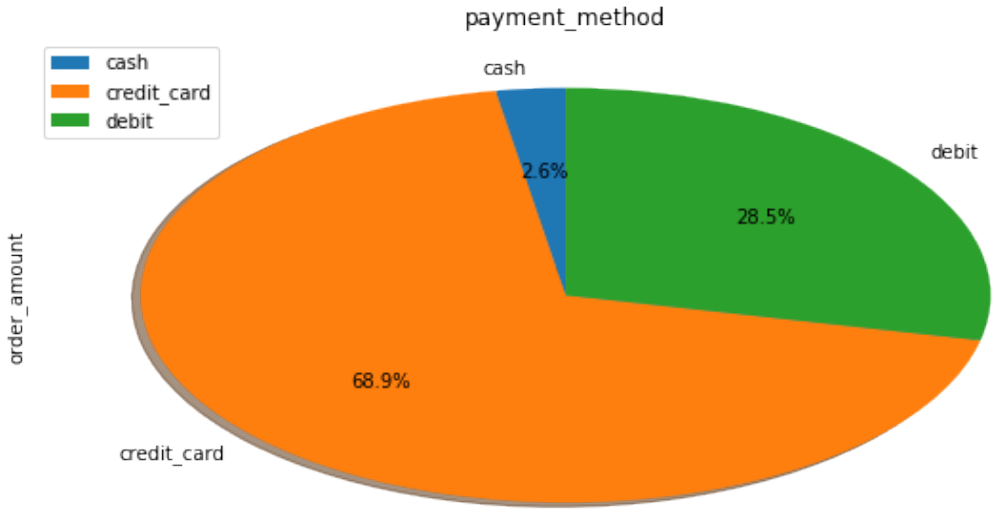

```
In [155]: df_Intern.plot(kind = 'hist', title = 'The total_items', figsize = (10, 5))
```

```
Out[155]: <matplotlib.axes._subplots.AxesSubplot at 0x7f55cd4bb358>
```



In [163]: df_Intern.groupby(['payment_method']).sum().plot(kind = 'pie', y = 'order_amount', tit

Out[163]: <matplotlib.axes._subplots.AxesSubplot at 0x7f55cc9137b8>



In [164]: '''From the analysis of payment_method, we have more of credit_card users with a perce

Out[164]: 'From the analysis of payment_method, we have more of credit_card users with a percent