PROBLEM 1: WHOLESALE CUSTOMERS ANALYSIS

Problem Statement:

A wholesale distributor operating in different regions of Portugal has information on annual spending of several items in their stores across different regions and channels. The data consists of 440 large retailers' annual spending on 6 different varieties of products in 3 different regions (Lisbon, Oporto, Other) and across different sales channel (Hotel, Retail).

SOLUTION: -

Data at a glance:

The data csv file is imported in python and operations performed to have a glimpse about the data variables and data columns.

df = pd.read_csv('Wholesale+Customers+Data.csv')

df.head()

df.tail()

```
df.head()
```

	Buyer/Spender	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
0	1	Retail	Other	12669	9656	7561	214	2674	1338
1	2	Retail	Other	7057	9810	9568	1762	3293	1776
2	3	Retail	Other	6353	8808	7684	2405	3516	7844
3	4	Hotel	Other	13265	1196	4221	6404	507	1788
4	5	Retail	Other	22615	5410	7198	3915	1777	5185

```
df.tail()
```

	Buyer/Spender	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
435	436	Hotel	Other	29703	12051	16027	13135	182	2204
436	437	Hotel	Other	39228	1431	764	4510	93	2346
437	438	Retail	Other	14531	15488	30243	437	14841	1867
438	439	Hotel	Other	10290	1981	2232	1038	168	2125
439	440	Hotel	Other	2787	1698	2510	65	477	52

Table 1.1

As can be observed from the head and tail functions, the columns are: Buyer/Spender, Channel, Region, Fresh, Milk, Grocery, Frozen, Detergents Paper and Delicatessen.

Buyer/ Spender serves no specific purpose, just contains serial nos. Region and Channel consists of categorical variables whereas the rest are numeric.

Further using df.info () information about each column variable can be known. Also no. of entities in each column, null or non-null could be known beforehand.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 440 entries, 0 to 439
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Buyer/Spender     440 non-null    int64
 1   Channel           440 non-null    object
 2   Region            440 non-null    object
 3   Fresh             440 non-null    int64
 4   Milk              440 non-null    int64
 5   Grocery           440 non-null    int64
 6   Frozen            440 non-null    int64
 7   Detergents_Paper  440 non-null    int64
 8   Delicatessen      440 non-null    int64
dtypes: int64(7), object(2)
```

From the above function a clear insight can be drawn on Data type of the variables and also their count. Apart from this it also tells about the null/non null values. It can be clearly seen that each of the 8 variables have 440 non null values. That also means the data has no null values. To put further evidence we may use the function df.isnull().sum() function.

```
df.isnull().sum()
```

```
Buyer/Spender       0
Channel             0
Region              0
Fresh               0
Milk                0
Grocery             0
Frozen              0
Detergents_Paper    0
Delicatessen        0
dtype: int64
```

Moving further, if we observe the Data type of variables, we can notice that Channel and Region are of object type whereas all others are int 64. This means that the object variables are categorical ones and the int 64 are numerical ones.

To check the no unique categorical variables, we can use the function df.Channel.unique() and df.Region.unique() functions.

```
df.Channel.unique()
```

```
array(['Retail', 'Hotel'], dtype=object)
```

```
df.Region.unique()
```

```
array(['Other', 'Lisbon', 'Oporto'], dtype=object)
```

The unique function tells us that there are two unique channels: Hotel and retail; and three unique regions: Lisbon, Oporto and Other.

We can also have a look at the shape of data i.e. no of rows and columns using df.shape(). The shape function clearly tells us that there are 440 rows and 9 columns.

```
df.shape
```

```
(440, 9)
```

1.1 Use methods of descriptive statistics to summarize data. Which Region and which Channel seems to spend more? Which Region and which Channel seems to spend less?

As the Buyer/Spender column spends no purpose apart from providing serial no. changing it to index.

Also, to get the spending trend a new column “Total Spend” needs to be created which would hold the sum of expenditures made on each of the items given across region and channel.

```
# Setting buyer/Spender as Index
df.set_index('Buyer/Spender', inplace = True)
# Creating another column total spend
df.loc[:,'Total Spend'] = df.Fresh + df.Milk + df.Grocery + df.Frozen
+ df.Detergents_Paper + df.Delicatessen
df.head()
```

Buyer/Spender	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen	Total Spend
1	Retail	Other	12669	9656	7561	214	2674	1338	34112
2	Retail	Other	7057	9810	9568	1762	3293	1776	33266
3	Retail	Other	6353	8808	7684	2405	3516	7844	36610
4	Hotel	Other	13265	1196	4221	6404	507	1788	27381
5	Retail	Other	22615	5410	7198	3915	1777	5185	46100

Table 1.2

To get an idea about the distribution of no of customers we can use df.value_counts(). This would help to get an overall idea about the average spending capacity across region and channel.

```
print(df.Channel.value_counts(),'\n')
print(df.Region.value_counts())
```

```
Hotel     298
Retail    142
Name: Channel, dtype: int64

Other     316
Lisbon     77
Oporto     47
Name: Region, dtype: int64
```

- 298 customers from Hotel and 142 from Retail
- 316 from other, 77 from Lisbon, 47 from Oporto

The next immediate step would to be to group the data into region and channel to evaluate the spending trend.

```
df_region_sum = df.groupby('Region').sum()
df_channel_sum = df.groupby('Channel').sum()
```

The groupby function groups the data into region and channel. The .sum() function implies that total sum will be calculated for every variable based on channel and sum.

```
df_region_sum.reset_index(inplace = True)
df_region_sum
```

	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen	Total Spend
0	Lisbon	854833	422454	570037	231026	204136	104327	2078350
1	Oporto	464721	239144	433274	190132	173311	54506	1327271
2	Other	3960577	1888759	2495251	930492	890410	512110	9275079

Table 1.3

The above table sums the total amount spend across each region and product. The index as been reset in order to get a separate column for “Region”.

```
df_channel_sum.reset_index(inplace = True)
df_channel_sum
```

	Channel	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen	Total Spend
0	Hotel	4015717	1028614	1180717	1116979	235587	421955	7342027
1	Retail	1264414	1521743	2317845	234671	1032270	248988	5338673

Table 1.4

The same logic applies for the above table too. It sums the spending amount across Channel and products.

Let's have a visualization at the spending amount across region and channel. For visualization, sns.barplot() and plt.subplot() can be used. Using subplot would give the visualization of two graphs together.

```
plt.figure(figsize = (13,5))
plt.subplot(1,2,1)
sns.barplot(df_region_sum.Region, df_region_sum['Total Spend'])
plt.subplot(1,2,2)
sns.barplot(df_channel_sum.Channel, df_channel_sum['Total Spend']);
```

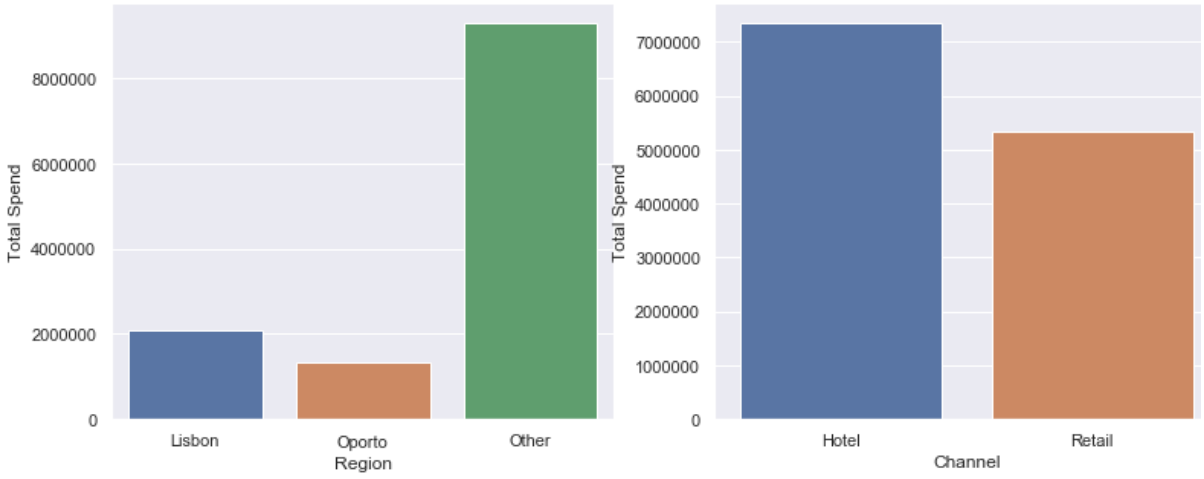

Fig 1.1

If we talk about total expenditure Region wise "**Other**" Region spends most: **10677599** and Oporto spends least: **1555088.**

If we talk about total expenditure Channel wise: **Hotels** tend to spend more when compared to **Retail**. **Hotels** total spend: **7342027**, **Retail** Spend: **5338673**.

The trend shows the total amount spend across region and channel. The channel and region showing maximum spend values reflect upon the total no of customers

coming from those regions and channel. More the customers more the total spend amount.

Although it might not be very right to draw some conclusion based on average expenditure made, but visualizing the same would give a rough idea about the average spending capacity across region and channel. (Not exact, as it is based on mean values, mean is often deceptive). Using sns.barplot() on the original data frame df.

```
plt.figure(figsize = (12,5))
plt.subplot(1,2,1)
sns.barplot(df.Region, df['Total Spend']);
plt.subplot(1,2,2)
sns.barplot(df.Channel, df['Total Spend'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x24ea15aaa08>
```

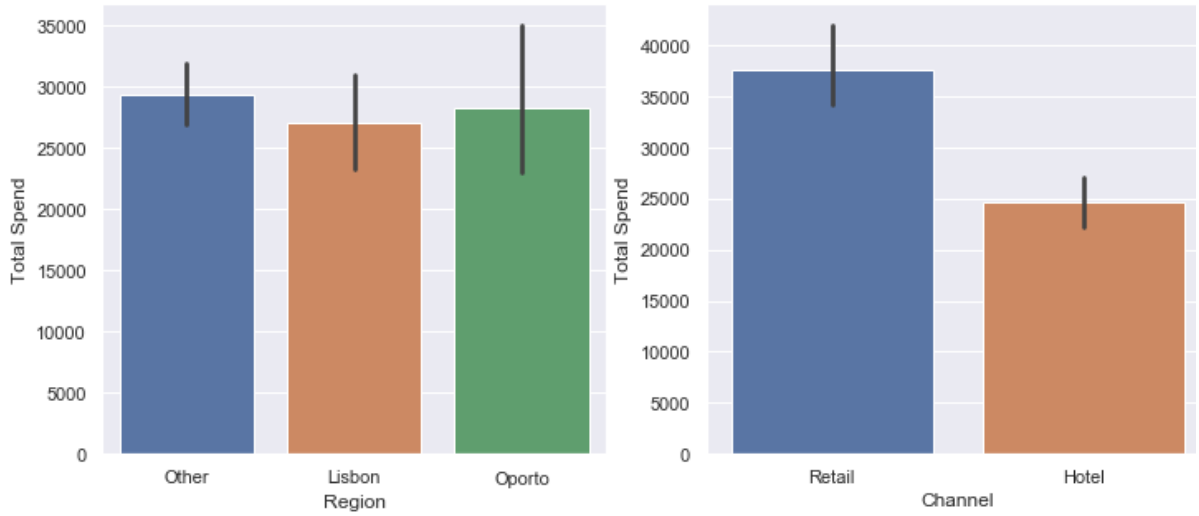

Fig 1.2

The above graph indicates the mean spending capacity across region and channel. It is very different from the total spend trend, but it is not correct to draw any insight based on mean values, as we do not yet know the outliers and inconsistency in the data.

CONCLUSION:

- Across Regions, “Other” tend to spend the most, while Oporto spend the least. Other: **10677599**, Oporto: **155048**.
- Across Channel, “Hotel” spends more then “Retail. Hotel: 7342027, Retail: **5338673**.
- Results obtained on sns.barplot() for the original data frame df, can’t be stated to be exactly true as they are based on mean values and means can often be deceptive. Also, we do not yet know the variation in the data.

1.2 There are 6 different varieties of items are considered. Do all varieties show similar behaviour across Region and Channel?

To check the behaviour, we need to use the describe () function. For that the data needs to be sorted based on region and channel. It can be done in two ways. We can create a new data frame and sort only the values containing a particular region or channel. The next way is to grouping the entire data based on region and channel and then using describe function on it. The former shows distribution based on a particular region or channel, whereas the latter gives distribution based on each product. It might seem a but chaotic here, but on applying the same it would get clearer.

A) SORTING THE DATA BASED ON REGION: -

```
df_lisbon = df[df['Region']=="Lisbon"]
df_oporto = df[df['Region']=="Oporto"]
df_other = df[df['Region'] == "Other"]
```

```
df_lisbon_des = df_lisbon.describe().T
df_oporto_des = df_oporto.describe().T
df_other_des = df_other.describe().T
```

Firstly, separate data frames consisting different regions are created, and individually describe function is used. Further, two new columns “IQR: Inter quartile range” and “COV: Coefficient of variation” are created, to get an idea about spread and consistency.

1) Lisbon

```
df_lisbon_des.loc[:,'IQR'] = df_lisbon_des['75%']-df_lisbon_des['25%']
```

```
df_lisbon_des.loc[:,'COV'] = df_lisbon_des['std']/df_lisbon_des['mean']
```

```
df_lisbon_des
```

	count	mean	std	min	25%	50%	75%	max	IQR	COV
Fresh	77.0	11101.727273	11557.438575	18.0	2806.0	7363.0	15218.0	56083.0	12412.0	1.041049
Milk	77.0	5486.415584	5704.856079	258.0	1372.0	3748.0	7503.0	28326.0	6131.0	1.039815
Grocery	77.0	7403.077922	8496.287728	489.0	2046.0	3838.0	9490.0	39694.0	7444.0	1.147670
Frozen	77.0	3000.337662	3092.143894	61.0	950.0	1801.0	4324.0	18711.0	3374.0	1.030599
Detergents_Paper	77.0	2651.116883	4208.462708	5.0	284.0	737.0	3593.0	19410.0	3309.0	1.587430
Delicatessen	77.0	1354.896104	1345.423340	7.0	548.0	806.0	1775.0	6854.0	1227.0	0.993008
Total Spend	77.0	30997.571429	20321.813773	4925.0	17184.0	25385.0	38699.0	107155.0	21515.0	0.655594

Table 1.5

The above table shows description of variables in Lisbon region. It can also be visualized using .hist() function.

```
print('\t\t\t\tLISBON DISTRIBUTION')
df_lisbon.hist(figsize = (10,10));
```

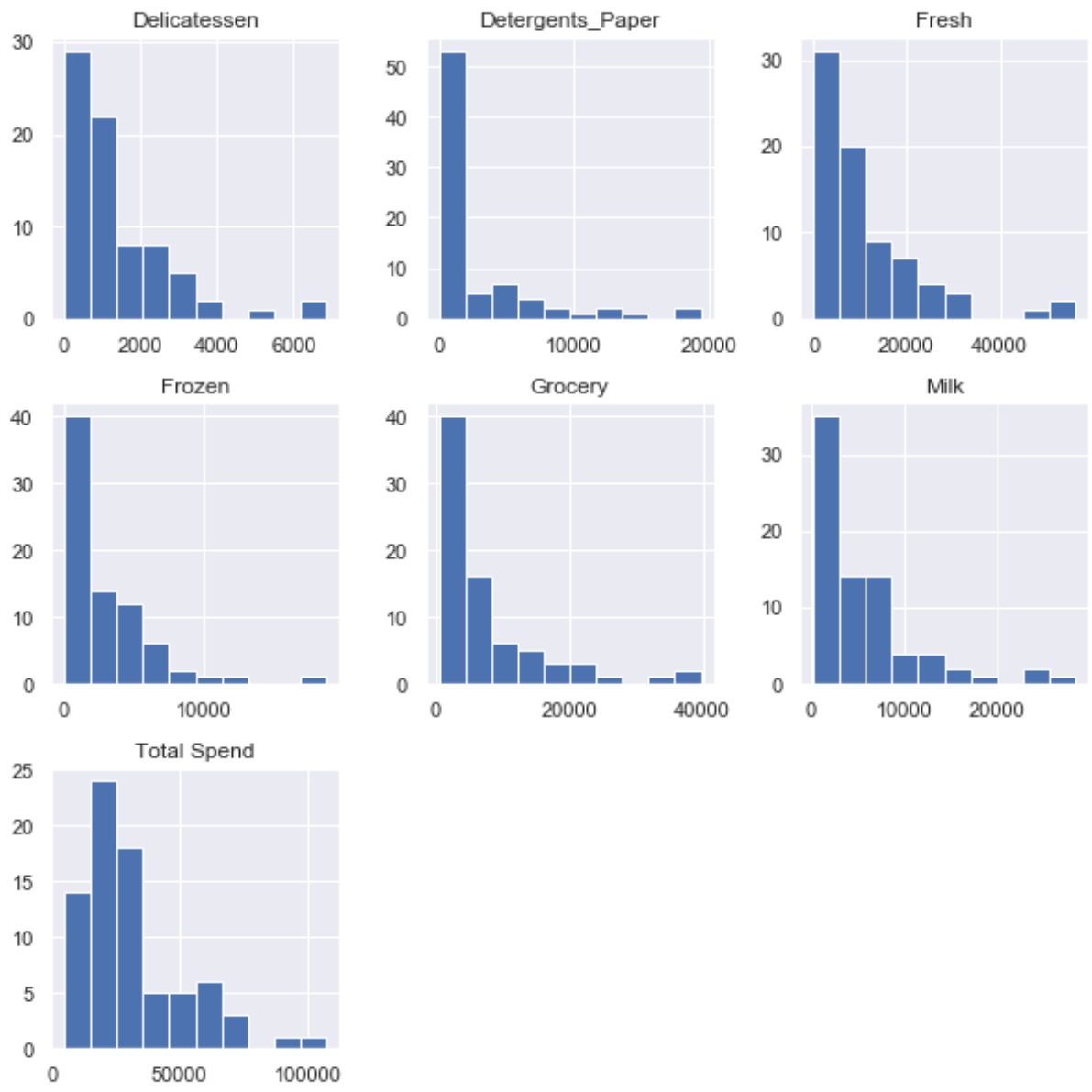

Fig 1.3

2) Oporto

```
df_oporto_des.loc[:,'IQR'] = df_oporto_des['75%']-df_oporto_des['25%']
```

```
df_oporto_des.loc[:,'COV'] = df_oporto_des['std']/df_oporto_des['mean']
```

OPORTO DISTRIBUTION

	count	mean	std	min	25%	50%	75%	max	IQR	COV
Fresh	47.0	9887.680851	8387.899211	3.0	2751.5	8090.0	14925.5	32717.0	12174.0	0.848318
Milk	47.0	5088.170213	5826.343145	333.0	1430.5	2374.0	5772.5	25071.0	4342.0	1.145076
Grocery	47.0	9218.595745	10842.745314	1330.0	2792.5	6114.0	11758.5	67298.0	8966.0	1.176182
Frozen	47.0	4045.361702	9151.784954	131.0	811.5	1455.0	3272.0	60869.0	2460.5	2.262291
Detergents_Paper	47.0	3687.468085	6514.717668	15.0	282.5	811.0	4324.5	38102.0	4042.0	1.766718
Delicatessen	47.0	1159.702128	1050.739841	51.0	540.5	898.0	1538.5	5609.0	998.0	0.906043
Total Spend	47.0	33086.978723	24234.507325	4129.0	20611.5	26953.0	36158.5	130877.0	15547.0	0.732448

Table 1.6

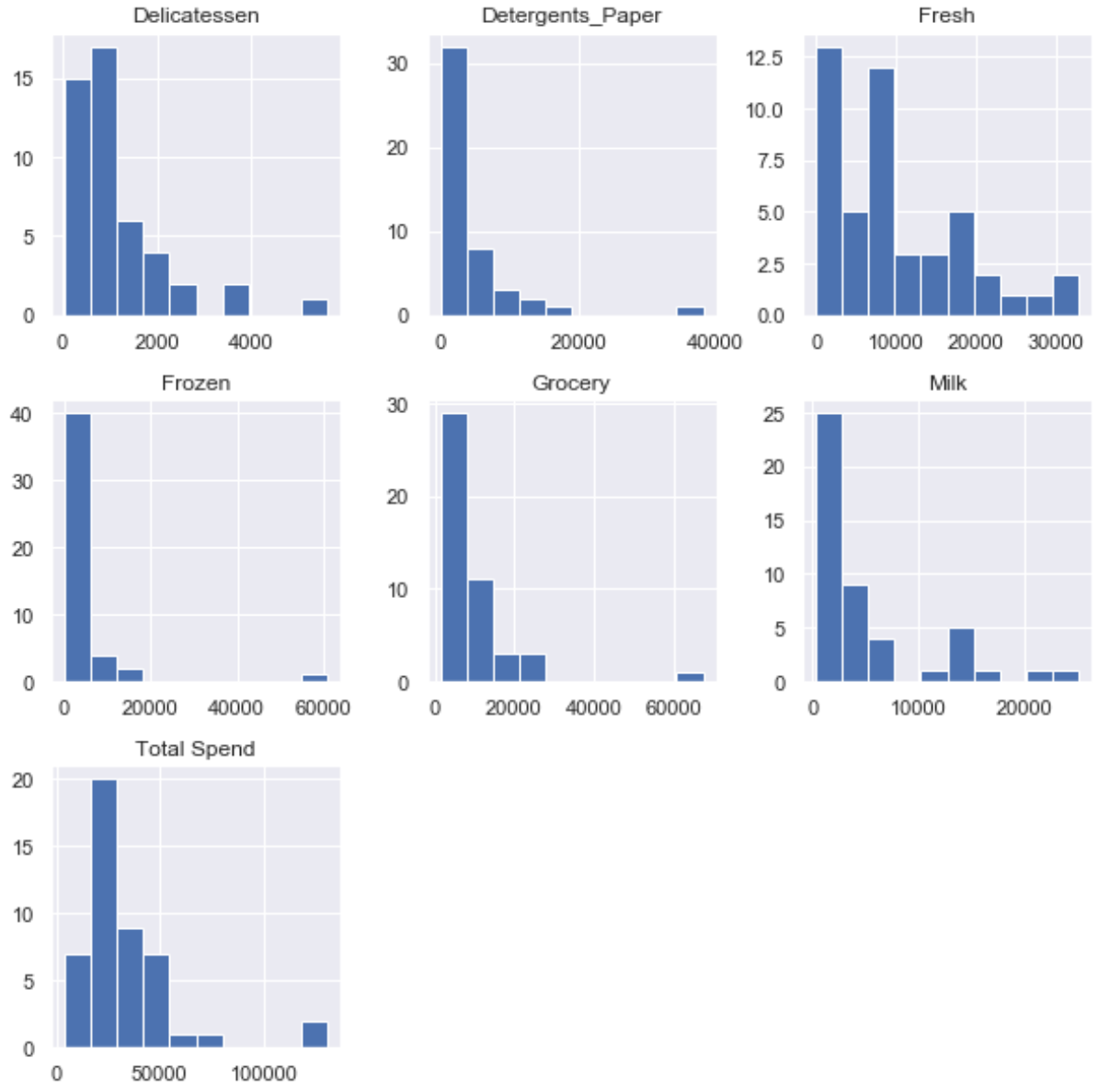

Fig 1.4

3) Other Distribution

```
df_other_des.loc[:,'IQR'] = df_other_des['75%']-df_other_des['25%']
df_other_des.loc[:,'COV'] = df_other_des['std']/df_other_des['mean']
print('OTHER DISTRIBUTION')
df_other_des
```

```
OTHER DISTRIBUTION
```

	count	mean	std	min	25%	50%	75%	max	IQR	COV
Fresh	316.0	12533.471519	13389.213115	3.0	3350.75	8752.5	17406.50	112151.0	14055.75	1.068277
Milk	316.0	5977.085443	7935.463443	55.0	1634.00	3684.5	7198.75	73498.0	5564.75	1.327648
Grocery	316.0	7896.363924	9537.287778	3.0	2141.50	4732.0	10559.75	92780.0	8418.25	1.207808
Frozen	316.0	2944.594937	4260.126243	25.0	664.75	1498.0	3354.75	36534.0	2690.00	1.446761
Detergents_Paper	316.0	2817.753165	4593.051613	3.0	251.25	856.0	3875.75	40827.0	3624.50	1.630040
Delicatessen	316.0	1620.601266	3232.581660	3.0	402.00	994.0	1832.75	47943.0	1430.75	1.994680
Total Spend	316.0	29351.515823	23618.538900	889.0	15133.50	24198.5	35878.25	176671.0	20744.75	0.804679

Table 1.7

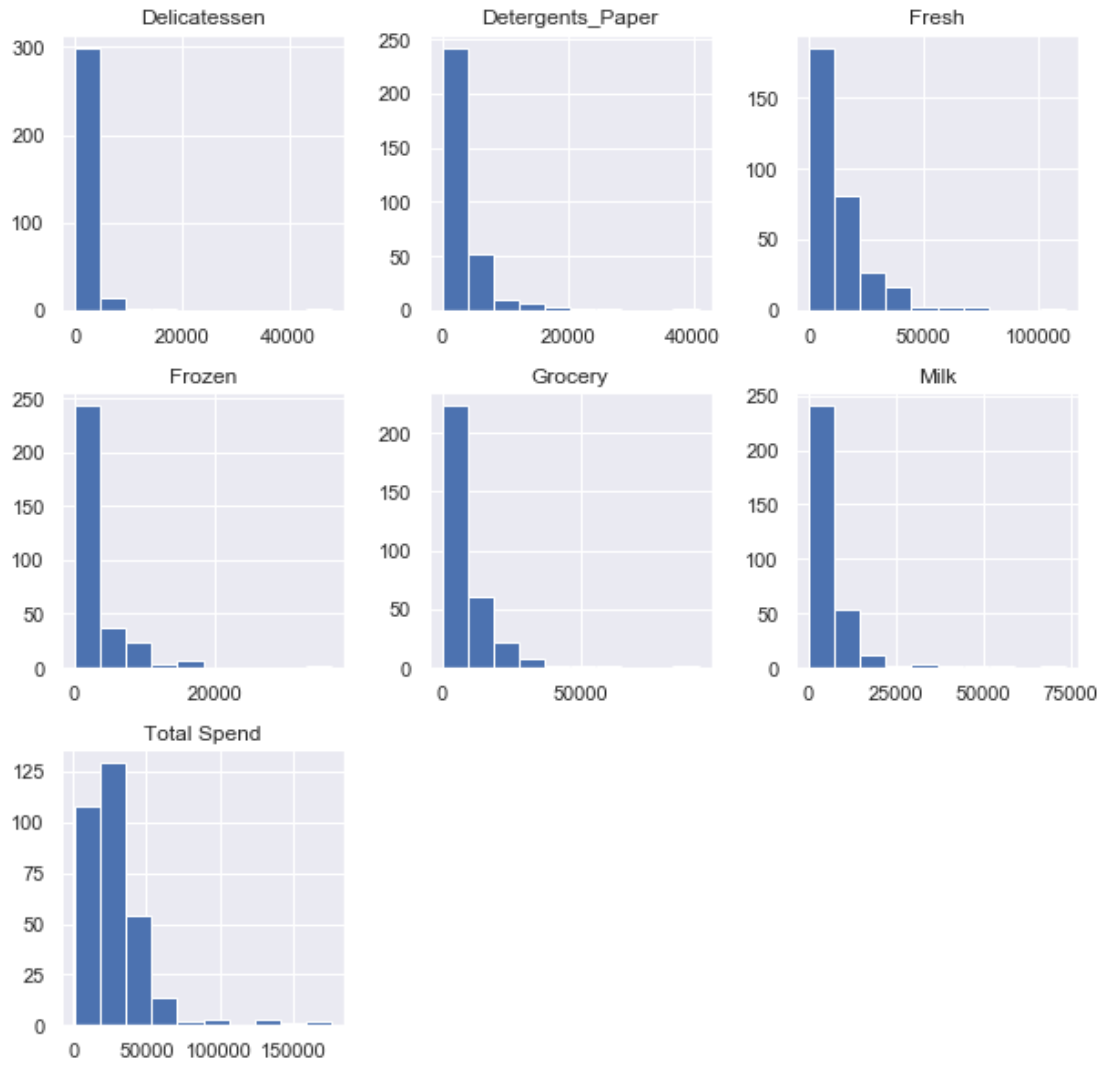

Fig 1.5

Insights:

Following insights can be drawn about each product variable across region. To zoom in further the data is grouped on region and describe function applied to create separate data frames.

```
df1 = df.groupby('Region').describe()
```

```
df_r_fresh = df1.Fresh
df_r_milk = df1.Milk
df_r_groc = df1.Grocery
df_r_froz = df1.Frozen
df_r_detr = df1.Detergents_Paper
df_r_del = df1.Delicatessen
```

1. Fresh:

```
df_r_milk.loc[:,'IQR'] = df_r_milk['75%'] - df_r_milk['25%']
```

```
df_r_milk.loc[:,'COV'] = df_r_milk['std']/df_r_milk['mean']
```

```
print('Fresh distribution across Regions: -')
df_r_fresh
```

```
Fresh distribution across Regions: -
```

Region	count	mean	std	min	25%	50%	75%	max	IQR	COV
Lisbon	77.0	11101.727273	11557.438575	18.0	2806.00	7363.0	15218.0	56083.0	12412.00	1.041049
Oporto	47.0	9887.680851	8387.899211	3.0	2751.50	8090.0	14925.5	32717.0	12174.00	0.848318
Other	316.0	12533.471519	13389.213115	3.0	3350.75	8752.5	17406.5	112151.0	14055.75	1.068277

Table 1.8

It can be clearly seen that the variable varies toom much across Lisbon and Other regions. Although not very consistent for Oporto but fat better comparatively. The IQR for Lisbon and Oporto are almost on similar grounds whereas, for Other its on the higher side. Keeping in view the count, other being on the highest side, the median expenditure is still comparable with Oporto and Lisbon. Min and Max for all the three are wide apart, with Oporto being the largest.

```
df.hist(by = 'Region', column = 'Fresh');
```

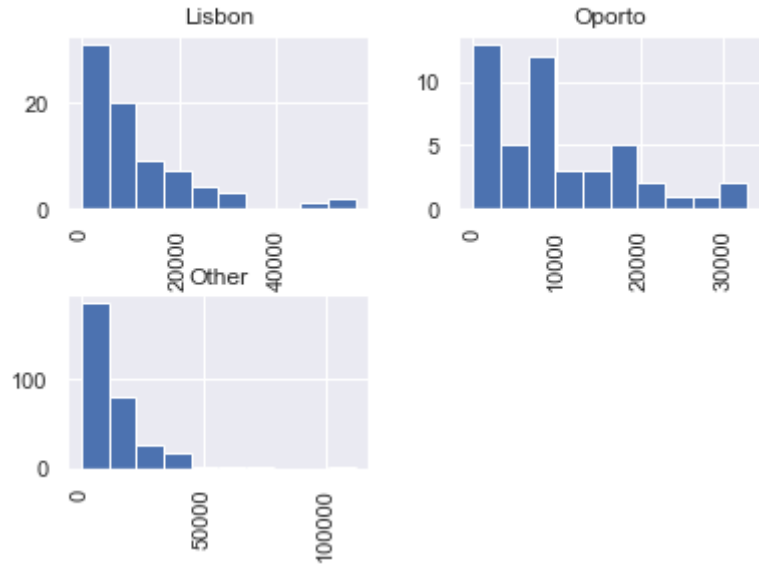

Fig 1.6

The visualization of the variable tells the similar story. Most inconsistent spread across Lisbon and other.

2. Milk

```
df_r_milk.loc[:,'IQR'] = df_r_milk['75%'] - df_r_milk['25%']
```

```
df_r_milk.loc[:,'COV'] = df_r_milk['std']/df_r_milk['mean']
```

```
df_r_milk
```

Region	count	mean	std	min	25%	50%	75%	max	IQR	COV
Lisbon	77.0	5486.415584	5704.856079	258.0	1372.0	3748.0	7503.00	28326.0	6131.00	1.039815
Oporto	47.0	5088.170213	5826.343145	333.0	1430.5	2374.0	5772.50	25071.0	4342.00	1.145076
Other	316.0	5977.085443	7935.463443	55.0	1634.0	3684.5	7198.75	73498.0	5564.75	1.327648

Table 1.9

Milk is widely spread across every region and is too inconsistent. For Oporto the standard deviation is too large comparing with the customer size (only 47). IQR is maximum for Lisbon, and least for Oporto. As stated earlier the COV for Oporto is

too much making it the most inconsistent. Median for Lisbon and Other are on similar grounds, while Oporto has a lesser value. Visualizing the milk spread based on regions would give some more idea about the spread and inconsistency.

```
df.hist(by = "Region", column = 'Milk',figsize = (7,5));
```

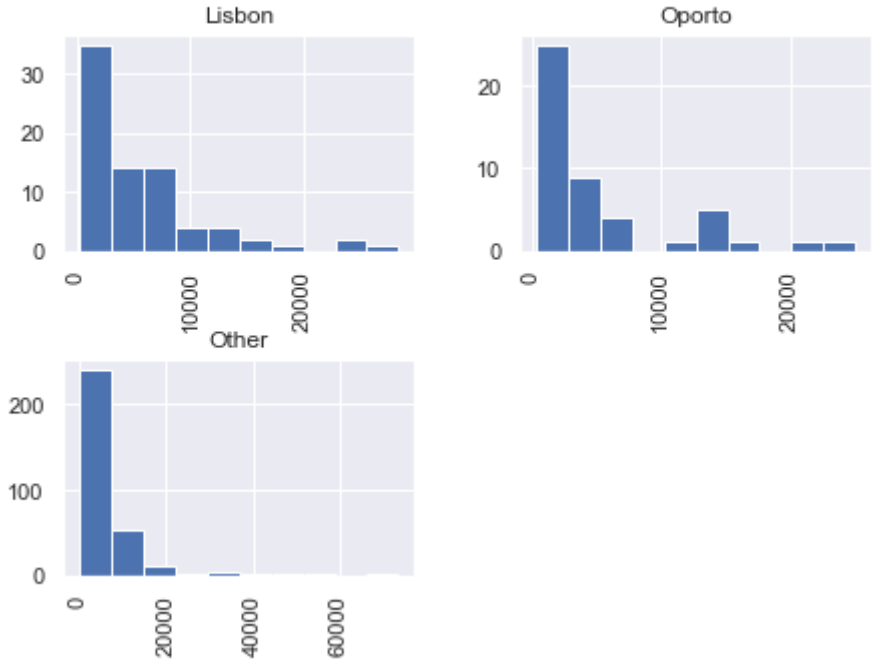

Fig 1.7

The high inconsistency in Oporto and Other is clearly visible.

3. Grocery

```
df_r_groc.loc[:,'IQR'] = df_r_groc['75%'] - df_r_groc['25%']
df_r_groc.loc[:,'COV'] = df_r_groc['std']/df_r_groc['mean']
df_r_groc
```

Region	count	mean	std	min	25%	50%	75%	max	IQR	COV
Lisbon	77.0	7403.077922	8496.287728	489.0	2046.0	3838.0	9490.00	39694.0	7444.00	1.147670
Oporto	47.0	9218.595745	10842.745314	1330.0	2792.5	6114.0	11758.50	67298.0	8966.00	1.176182
Other	316.0	7896.363924	9537.287778	3.0	2141.5	4732.0	10559.75	92780.0	8418.25	1.207808

Table 1.10

Grocery is also too much inconsistent. COV across all regions is too high. IQR for Oporto is highest, while Lisbon is on the lower side. The highest standard deviation

is for Oporto, while lowest for Lisbon. The variable is much more spread across Oporto and Other regions.

```
df.hist(by = 'Region', column = 'Grocery', figsize=(6,6));
```

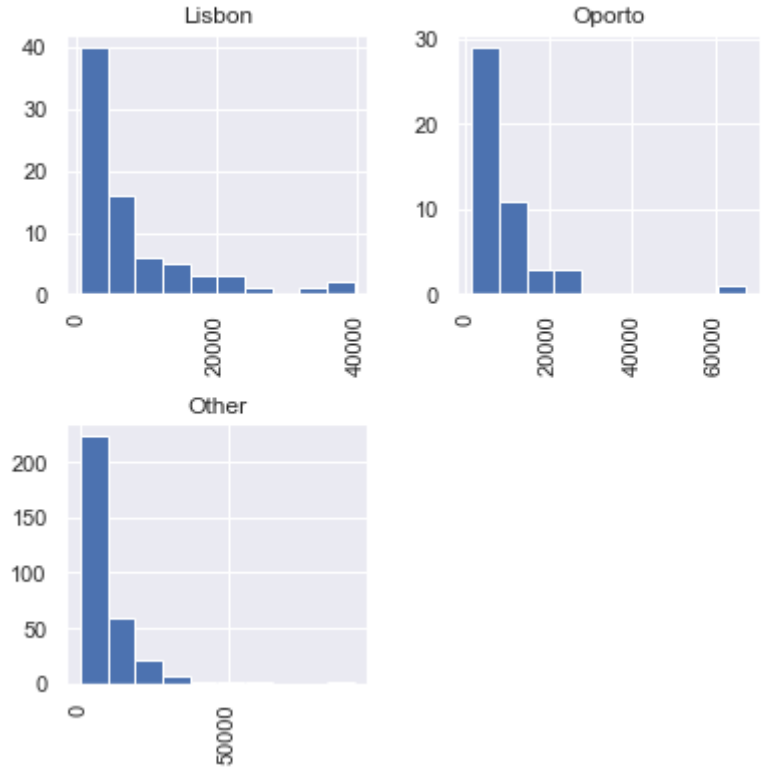

Fig 1.8

4. Frozen:

```
df_r_froz.loc[:,'IQR'] = df_r_froz.loc[:,'75%']-df_r_froz.loc[:,'25%']
df_r_froz.loc[:,'COV'] = df_r_froz.loc[:,'std']/df_r_froz.loc[:,'mean']
df_r_froz
```

Region	count	mean	std	min	25%	50%	75%	max	IQR	COV
Lisbon	77.0	3000.337662	3092.143894	61.0	950.00	1801.0	4324.00	18711.0	3374.0	1.030599
Oporto	47.0	4045.361702	9151.784954	131.0	811.50	1455.0	3272.00	60869.0	2460.5	2.262291
Other	316.0	2944.594937	4260.126243	25.0	664.75	1498.0	3354.75	36534.0	2690.0	1.446761

Table 1.11

The COV for Oporto is max making it the most inconsistent. Also, there is a large difference between the min and max values for Oporto. The same applies for all three but the difference is max for Oporto. All three are inconsistent. The standard

deviation is greater than mean, which is not a good sign for an even spread. IQR is highest for Lisbon, making it quite broad and consistent comparatively. The visualization speaks the same.

```
df.hist(by = 'Region', column = 'Frozen',figsize=(5,5.5));
```

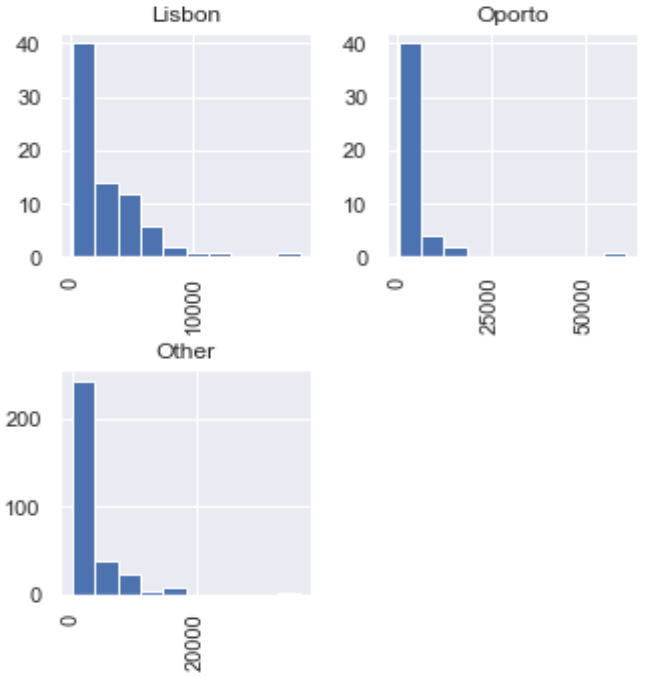

Fig 1.9

5. Detergents Paper:

```
df_r_detr.loc[:,'IQR'] = df_r_detr.loc[:,'75%'] - df_r_detr.loc[:,'25%']
df_r_detr.loc[:,'COV'] = df_r_detr.loc[:,'std']/df_r_detr.loc[:,'mean']
df_r_detr
```

Region	count	mean	std	min	25%	50%	75%	max	IQR	COV
Lisbon	77.0	2651.116883	4208.462708	5.0	284.00	737.0	3593.00	19410.0	3309.0	1.587430
Oporto	47.0	3687.468085	6514.717668	15.0	282.50	811.0	4324.50	38102.0	4042.0	1.766718
Other	316.0	2817.753165	4593.051613	3.0	251.25	856.0	3875.75	40827.0	3624.5	1.630040

Table 1.12

COV for Oporto is max, again making it the most inconsistent. Too high standard deviation compared to the mean. IQR for Oporto region is the largest while quite comparable values for Lisbon and Other regions. Looking at the visualization would give better understanding.

```
df.hist(by = 'Region', column = 'Detergents_Paper', figsize=(5,5.3));
```

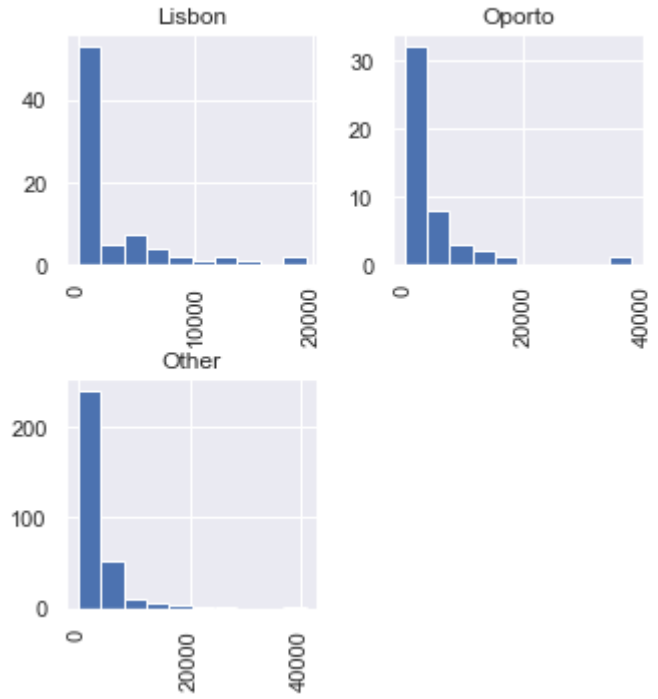

Fig 1.10

The visualization clearly states the inconsistency in the data. Comparatively Lisbon is more evenly spread in the latter parts moving towards right. While other and Oporto almost have no values towards right.

6. Delicatessen:

```
df_r_del.loc[:,'IQR'] = df_r_del.loc[:,'75%'] - df_r_del.loc[:,'25%']
df_r_del.loc[:,'COV'] = df_r_del.loc[:,'std']/df_r_del.loc[:,'mean']
df_r_del
```

Region	count	mean	std	min	25%	50%	75%	max	IQR	COV
Lisbon	77.0	1354.896104	1345.423340	7.0	548.0	806.0	1775.00	6854.0	1227.00	0.993008
Oporto	47.0	1159.702128	1050.739841	51.0	540.5	898.0	1538.50	5609.0	998.00	0.906043
Other	316.0	1620.601266	3232.581660	3.0	402.0	994.0	1832.75	47943.0	1430.75	1.994680

Table 1.13

The variable Delicatessen seems to have a somewhat better distribution in terms of consistency in Lisbon and Oporto, when compared to other. The COV for other is almost twice for Other. The IQR is min for Oporto and Max for Other. Significant difference between max and min values for all the three regions. Standard deviation

is highest for Other and lowest for Oporto. Visualizing the same could give a better idea about the spread.

```
df.hist(by = 'Region', column = 'Delicatessen', figsize=(5,5.5));
```

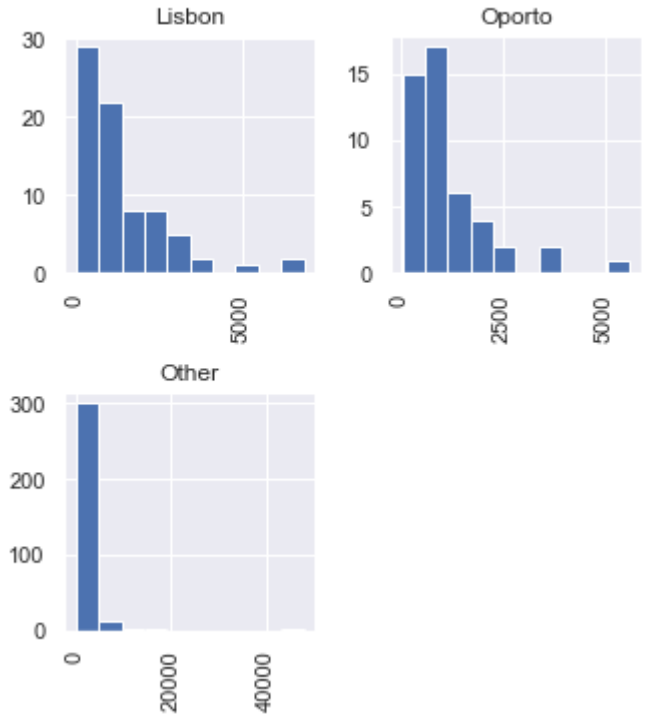

Fig 1.11

The visualization tells the same story. Lisbon and Oporto are comparatively evenly distributed on the left side. Region other has no values towards right.

B) SORTING BASED ON CHANNEL

```
df_hotel = df[df.Channel == "Hotel"]
df_retail = df[df.Channel == "Retail"]
```

```
df_hotel_des = df_hotel.describe().T
df_retail_des = df_retail.describe().T
```

The above code sorts the data into two data frames. describe () function is applied on each of the two data frames. Further columns of IQR and COV are explicitly added. Further following the previous approach, visualizations for each channel are depicted.

1. Hotel:

```
df_hotel_des.loc[:,'IQR'] = df_hotel_des['75%'] - df_hotel_des['25%']
df_hotel_des.loc[:,'COV'] = df_hotel_des['std']/df_hotel_des['mean']
```

```
print('HOTEL DISTRIBUTION')
df_hotel_des
```

```
HOTEL DISTRIBUTION
```

	count	mean	std	min	25%	50%	75%	max	IQR	COV
Fresh	298.0	13475.560403	13831.687502	3.0	4070.25	9581.5	18274.75	112151.0	14204.50	1.026428
Milk	298.0	3451.724832	4352.165571	55.0	1164.50	2157.0	4029.50	43950.0	2865.00	1.260867
Grocery	298.0	3962.137584	3545.513391	3.0	1703.75	2684.0	5076.75	21042.0	3373.00	0.894849
Frozen	298.0	3748.251678	5643.912500	25.0	830.00	2057.5	4558.75	60869.0	3728.75	1.505745
Detergents_Paper	298.0	790.560403	1104.093673	3.0	183.25	385.5	899.50	6907.0	716.25	1.396596
Delicatessen	298.0	1415.956376	3147.426922	3.0	379.00	821.0	1548.00	47943.0	1169.00	2.222828
Total Spend	298.0	26844.191275	22164.839073	904.0	13859.25	21254.5	32113.75	190169.0	18254.50	0.825685

Table 1.14

The distribution can be visualized as follows using the. hist () function.

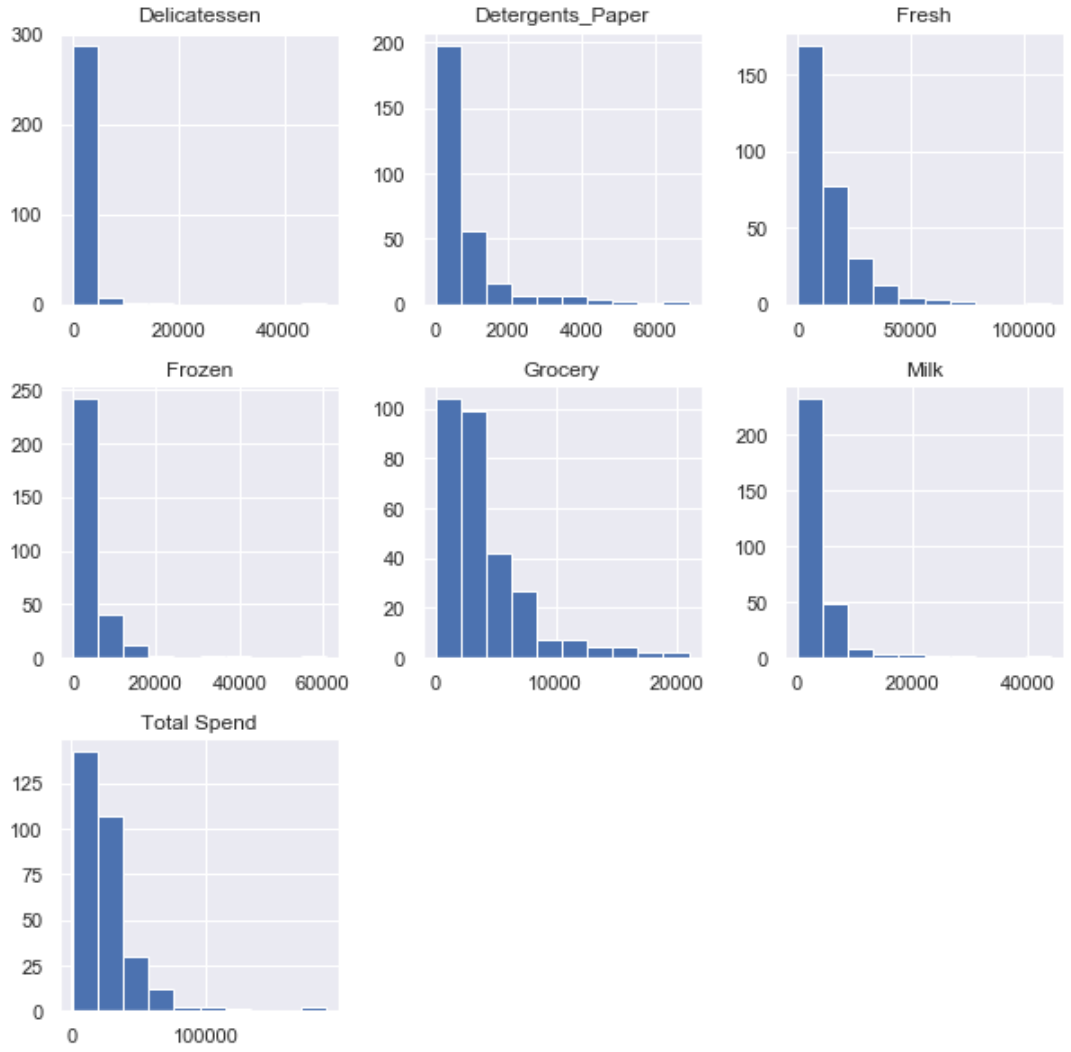

Fig 1.12

2. Retail:

```
df_retail_des.loc[:,'IQR'] = df_retail_des['75%'] - df_retail_des['25%']
df_retail_des.loc[:,'COV'] = df_retail_des['std'] / df_retail_des['mean']
print('RETAIL DISTRIBUTION')
df_retail_des
```

```
RETAIL DISTRIBUTION
```

	count	mean	std	min	25%	50%	75%	max	IQR	COV
Fresh	142.0	8904.323944	8987.714750	18.0	2347.75	5993.5	12229.75	44466.0	9882.00	1.009365
Milk	142.0	10716.500000	9679.631351	928.0	5938.00	7812.0	12162.75	73498.0	6224.75	0.903246
Grocery	142.0	16322.852113	12267.318094	2743.0	9245.25	12390.0	20183.50	92780.0	10938.25	0.751543
Frozen	142.0	1652.612676	1812.803662	33.0	534.25	1081.0	2146.75	11559.0	1612.50	1.096932
Detergents_Paper	142.0	7269.507042	6291.089697	332.0	3683.50	5614.5	8662.50	40827.0	4979.00	0.865408
Delicatessen	142.0	1753.436620	1953.797047	3.0	566.75	1350.0	2156.00	16523.0	1589.25	1.114267
Total Spend	142.0	37596.288732	23711.757409	10902.0	23778.50	30614.0	42577.75	162078.0	18799.25	0.630694

Table 1.15

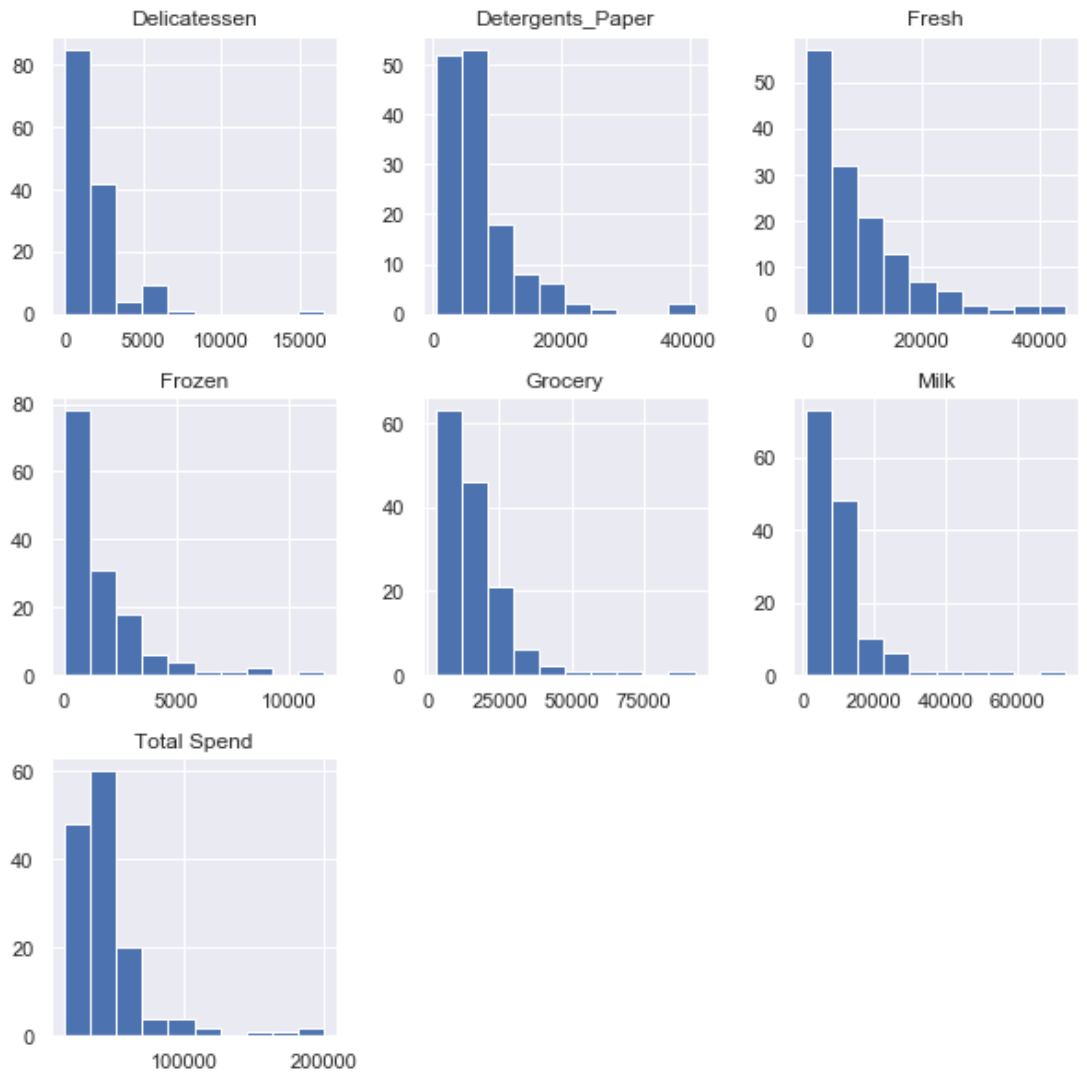

Fig 1.13

Insights:

Following insights can be drawn about each product variable across region. To zoom in further the data is grouped on channel and describe function applied to create separate data frames.

```
df2 = df.groupby('Channel').describe()
```

```
df2_c_fresh = df2.Fresh
df2_c_milk = df2.Milk
df2_c_groc = df2.Grocery
df2_c_froz = df2.Frozen
df2_c_det = df2.Detergents_Paper
df2_c_del = df2.Delicatessen
```

1. Fresh:

```
df2_c_fresh.loc[:,'IQR'] = df2_c_fresh.loc[:,'75%'] - df2_c_fresh.loc[:,'25%']
df2_c_fresh.loc[:,'COV'] = df2_c_fresh.loc[:,'std']/df2_c_fresh.loc[:,'mean']
df2_c_fresh
```

Channel	count	mean	std	min	25%	50%	75%	max	IQR	COV
Hotel	298.0	13475.560403	13831.687502	3.0	4070.25	9581.5	18274.75	112151.0	14204.5	1.026428
Retail	142.0	8904.323944	8987.714750	18.0	2347.75	5993.5	12229.75	44466.0	9882.0	1.009365

Table 1.16

The Fresh variable tends to have a lot of inconsistency across hotel and retail channels which can be seen in the high COV. The IQR for Hotel is much more for Hotel channel, that means the data is more wide spread across hotel channel. Also, the standard deviation for hotel is much more. With visualization the spread of the data would become clearer. The COV for Retail channel is less which can be seen in the following plot, more even distribution across retail.

```
df.hist(by = 'Channel', column = 'Fresh', figsize = (5,3));
```

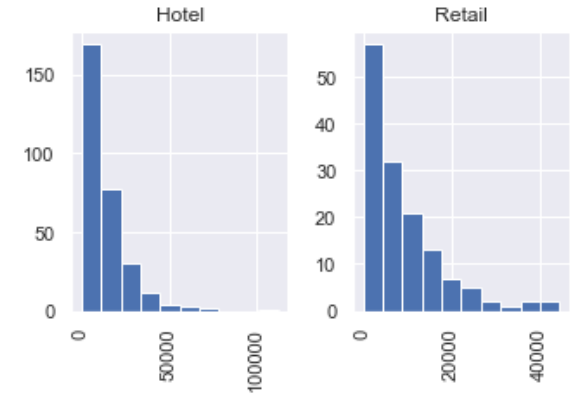

Fig1.14

2. Milk:

```
df2_c_milk.loc[:,'IQR'] = df2_c_milk['75%'] - df2_c_milk['25%']
df2_c_milk.loc[:,'COV'] = df2_c_milk['std']/df2_c_milk['mean']
df2_c_milk
```

Channel	count	mean	std	min	25%	50%	75%	max	IQR	COV
Hotel	298.0	3451.724832	4352.165571	55.0	1164.5	2157.0	4029.50	43950.0	2865.00	1.260867
Retail	142.0	10716.500000	9679.631351	928.0	5938.0	7812.0	12162.75	73498.0	6224.75	0.903246

Table 1.17

Looking at the COV the spread is more inconsistent across hotel channel. The IQR of retail is more which indicates the spread is more. For hotel there is large difference between min and max, same applies for Retail as well. Visualization would make it clearer. Retail is more even and also more widespread.

```
df.hist(by = 'Channel', column = 'Milk', figsize = (5,3));
```

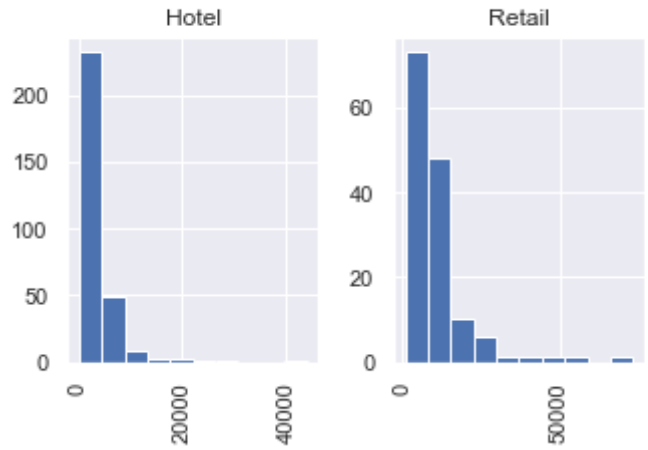

Fig 1.15

3. Grocery:

```
df2_c_groc.loc[:,'IQR'] = df2_c_groc['75%'] - df2_c_groc['25%']
df2_c_groc.loc[:,'COV'] = df2_c_groc['std']/df2_c_groc['mean']
df2_c_groc
```

Channel	count	mean	std	min	25%	50%	75%	max	IQR	COV
Hotel	298.0	3962.137584	3545.513391	3.0	1703.75	2684.0	5076.75	21042.0	3373.00	0.894849
Retail	142.0	16322.852113	12267.318094	2743.0	9245.25	12390.0	20183.50	92780.0	10938.25	0.751543

Table 1.17

The distribution of grocery in both channels seems better than the rest of the variables seen so far. COV<1, but still it is even better for Retail channel. IQR for retail is high pointing towards more spread. Also, the standard deviation for retail is on the higher side. For both hotel and Retail the difference between min and max is too wide. In hotel channel the values seem to concentrate more towards the left side, while for Retail the values go on up to a higher limit.

```
df.hist(by = 'Channel', column = 'Grocery', figsize = (5,3));
```

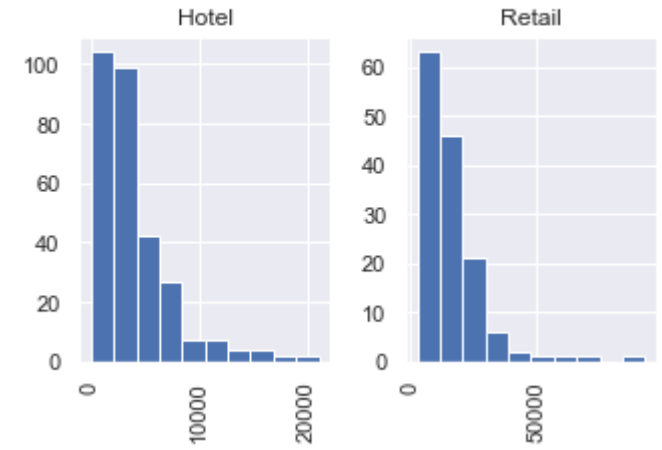

Fig 1.16

4. Frozen:

```
df2_c_froz.loc[:,'IQR'] = df2_c_froz.loc[:,'75%'] - df2_c_froz.loc[:,'25%']
df2_c_froz.loc[:,'COV'] = df2_c_froz.loc[:,'std']/df2_c_froz.loc[:,'mean']
df2_c_froz
```

Channel	count	mean	std	min	25%	50%	75%	max	IQR	COV
Hotel	298.0	3748.251678	5643.912500	25.0	830.00	2057.5	4558.75	60869.0	3728.75	1.505745
Retail	142.0	1652.612676	1812.803662	33.0	534.25	1081.0	2146.75	11559.0	1612.50	1.096932

Table 1.18

The frozen variable is more evenly distributed in retail channel, depicting the low COV. The COV for Hotel is much higher. The IQR for hotel has a larger value pointing towards a wider spread. The standard deviation is also much larger for hotel channel. For both regions, there is a huge gap between min and max values.

Visualization of Frozen across hotel and channel: -

```
df.hist(by = 'Channel', column = 'Frozen', figsize = (5,3));
```

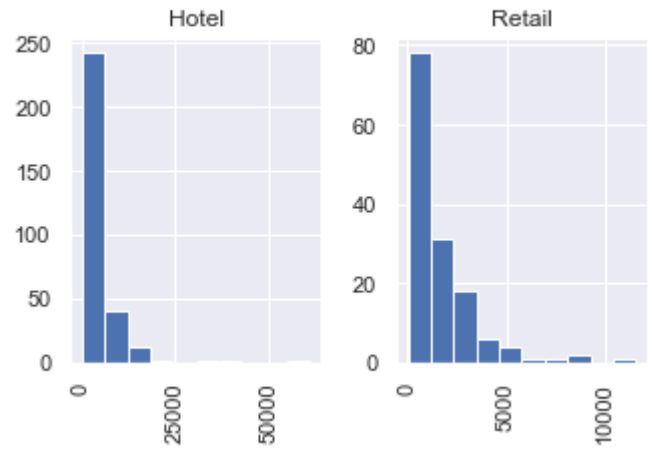

Fig 1.17

5. Detergents Paper:

```
df2_c_det.loc[:,'IQR'] = df2_c_det.loc[:,'75%'] - df2_c_det.loc[:,'25%']
df2_c_det.loc[:,'COV'] = df2_c_det.loc[:,'std'] / df2_c_det.loc[:,'mean']
df2_c_det
```

Channel	count	mean	std	min	25%	50%	75%	max	IQR	COV
Hotel	298.0	790.560403	1104.093673	3.0	183.25	385.5	899.5	6907.0	716.25	1.396596
Retail	142.0	7269.507042	6291.089697	332.0	3683.50	5614.5	8662.5	40827.0	4979.00	0.865408

Table 1.19

The variable is evenly distributed across retail channel compared to hotel channel, which can be derived from the low COV for Retail. The IQR is more for retail and too less for Hotel. The standard deviation is too high for hotel. The difference between min and max values for Retail are too high.

```
df.hist(by = 'Channel', column = 'Detergents_Paper', figsize = (5,3));
```

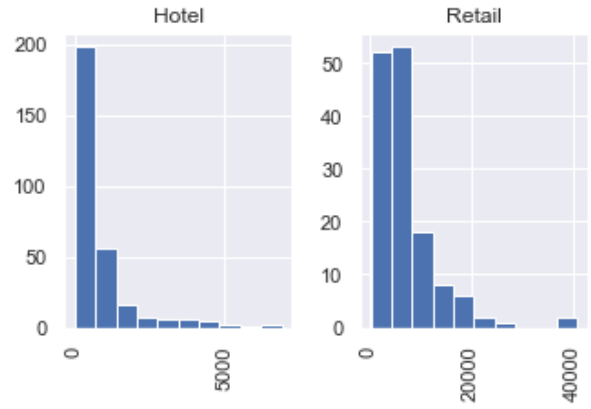

Fig1.18

6. Delicatessen:

```
df2_c_del.loc[:,'IQR'] = df2_c_del.loc[:,'75%'] - df2_c_del.loc[:,'25%']
df2_c_del.loc[:,'COV'] = df2_c_del.loc[:,'std'] / df2_c_del.loc[:,'mean']
df2_c_del
```

Channel	count	mean	std	min	25%	50%	75%	max	IQR	COV
Hotel	298.0	1415.956376	3147.426922	3.0	379.00	821.0	1548.0	47943.0	1169.00	2.222828
Retail	142.0	1753.436620	1953.797047	3.0	566.75	1350.0	2156.0	16523.0	1589.25	1.114267

Table 1.20

The variable delicatessen is way too inconsistent in hotel channel (COV 2.22), still high in Retail. IQR is higher for Retail channel, standard deviation higher in Hotel. Again, large difference between min and max values pointing towards way too uneven spread.

```
df.hist(by = 'Channel', column = 'Delicatessen', figsize = (5,3));
```

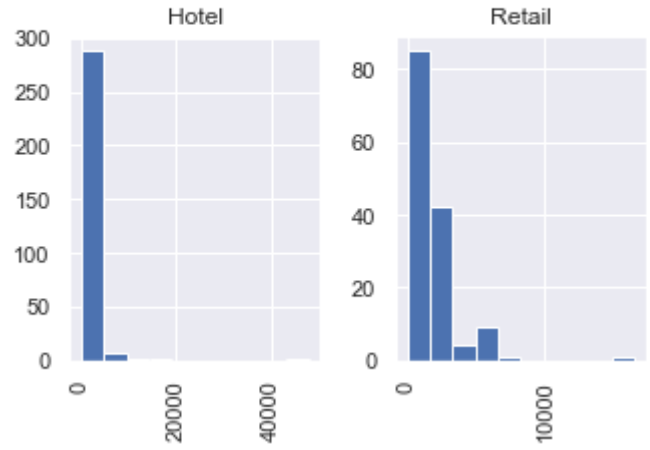

Fig 1.19

1.3 On the basis of descriptive measure of variability, which item shows the most inconsistent behaviour? Which items show the least inconsistent behaviour?

Inconsistency in the data can be very well checked using the Coefficient of variation, which was widely used in the previous problem. On the original data frame, describe function can be used to create a new data frame. On the newly created data frame two new columns “IQR” and “COV” can be introduced. Based on the COV values we can infer variation and consistency.

```
df_des = df.describe().T
df_des.loc[:,'IQR'] = df_des['75%'] - df_des['25%']
```

```
df_des.loc[:,'COV'] = df_des['std']/df_des['mean']
```

```
df_des[['std','mean','IQR','COV']].sort_values(by = "COV", ascending = False)
```

	std	mean	IQR	COV
Delicatessen	2820.105937	1524.870455	1412.00	1.849407
Detergents_Paper	4767.854448	2881.493182	3665.25	1.654647
Frozen	4854.673333	3071.931818	2812.00	1.580332
Milk	7380.377175	5796.265909	5657.25	1.273299
Grocery	9503.162829	7951.277273	8502.75	1.195174
Fresh	12647.328865	12000.297727	13806.00	1.053918
Total Spend	26356.301730	33226.136364	23858.75	0.793240

Table 1.21

Using the sort_values () function the data frame is sorted in descending order. It can be clearly inferred that COV for **Delicatessen** is highest: **1.89** while for **Fresh** it is the lowest:**1.05**. So, to conclude the variable Delicatessen is the most inconsistent, while the variable is the most consistent.

Further, to asses this visually, function sns.distplot(), can be used to get idea about inconsistency and unevenness. Subplot has been used to get idea in a single plot.

The **dist plot ()** gives a histogram along with a kernel density estimate curve.

```
print('\t\t\t\t\tDISTRIBUTION OF VARIABLES')

plt.figure(figsize = (12,12))
plt.subplot(3,2,1)
sns.distplot(df.Fresh)
plt.subplot(3,2,2)
sns.distplot(df.Milk)
plt.subplot(3,2,3)
sns.distplot(df.Grocery)
plt.subplot(3,2,4)
sns.distplot(df.Frozen)
plt.subplot(3,2,5)
sns.distplot(df.Detergents_Paper)
plt.subplot(3,2,6)
sns.distplot(df.Delicatessen)
plt.show()
```

The following graph clearly depicts the unevenness in the Delicatessen.

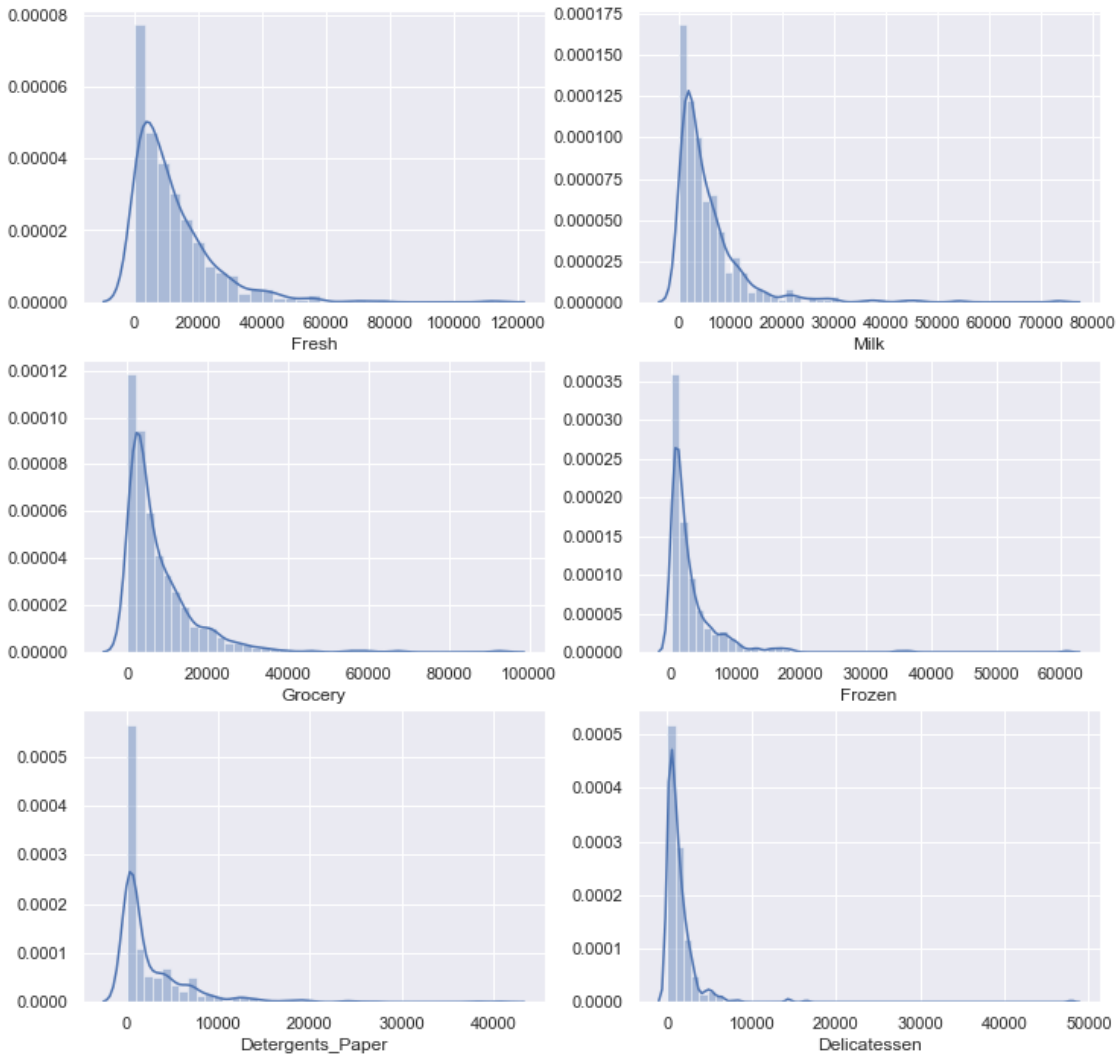

Fig 1.20

1.4 Are there any outliers in the data?

The best way to detect any outliers in the data is to generate boxplots. Outliers basically are the values lying beyond 1.5 x IQR + Q1 and 1.55 x IQR + Q3.

sns.boxplot() can be used along with sublots to get an idea about outliers for all the numerical variables in a go.

```
print('\t\t\t\t\t\tDETECTING OUTLIERS')
plt.figure(figsize = (15,8))
plt.subplot(2,3,1)
sns.boxplot(df.Fresh)
plt.subplot(2,3,2)
sns.boxplot(df.Milk)
plt.subplot(2,3,3)
sns.boxplot(df.Grocery)
plt.subplot(2,3,4)
sns.boxplot(df.Frozen)
plt.subplot(2,3,5)
sns.boxplot(df.Detergents_Paper)
plt.subplot(2,3,6)
sns.boxplot(df.Delicatessen)
plt.show()
```

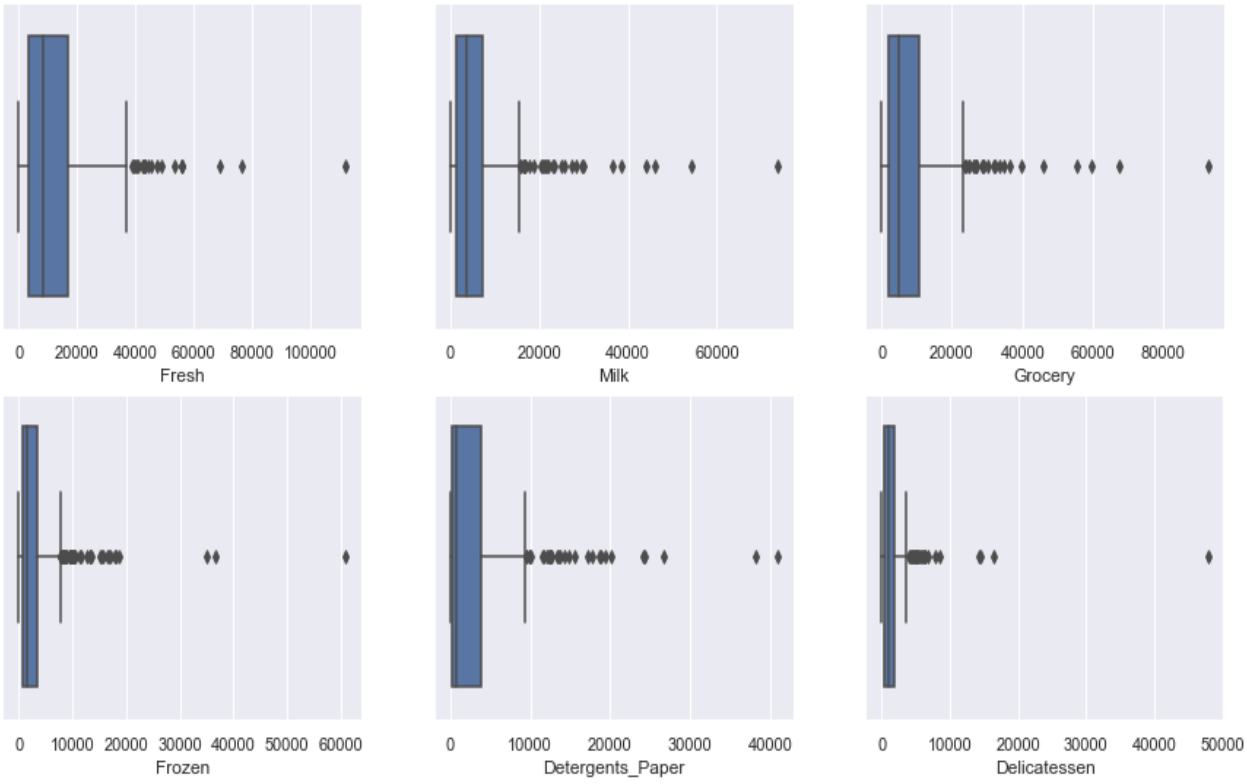

Fig 1.21

It is clearly evident from the box plot that all the variables have many outliers and at least there is one point which is too far away. These outliers cause variation.

1.5 On the basis of this report, what are the recommendations?

To start with let's have a look at client distribution of hotel and region. Creating a crosstab, it's found that Lisbon has 59 hotel and 18 Retail customers Oporto has 28 hotel and 19 Retail, while Other regions have 211 hotel and 105 retail customers.

In total there are 298 hotel and 142 retail channel customers. Region wise Other comprises of 316, Lisbon: 77 and Oporto: 47.

```
df_cross = pd.crosstab(df.Region, df.Channel, margins=True)
df_cross
```

Channel	Hotel	Retail	All
Region			
Lisbon	59	18	77
Oporto	28	19	47
Other	211	105	316
All	298	142	440

Table 1.22

```
plt.title('COUNT PLOT')
sns.countplot(df.Region, hue = df.Channel);
```

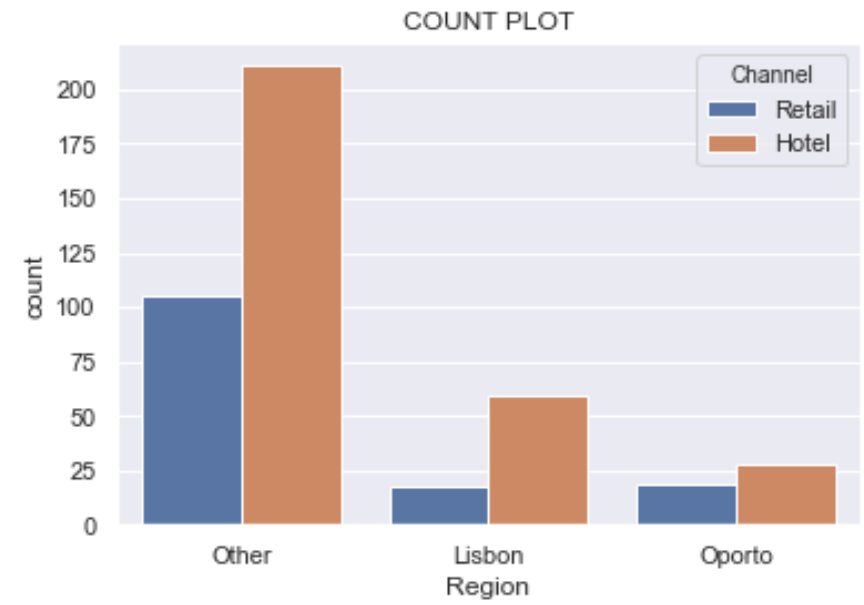

Fig 1.22

Insights and recommendation: -

- Coming to the expenditure trend, Region wise, other regions spend much more compared to Lisbon and Oporto. Channel wise Hotels spend more. Since the data is too inconsistent it would be wrong to infer a trend based on mean, but nonetheless median can be a decent option to explore expenditure trends.

- Hotels are spending a median amount of **21254** while Retailers spend **37139**.It's interesting to notice that in spite of being less in no **(142)** as compared to hotels, retails spend more. Although, the hotel clients spend more than retailers, the wholesaler should try to find more retail clients. The reason being that the **total spend variable is more consistent** in the **retail channel**. Hotels are subjected to more risk comparatively. Also, the median spend of retail is more than hotels, so it can be a good business decision to target the retailers.

- Comparing the expenditure trends based on the products we can draw the following insights: -
  - Hotels tend to spend more on Fresh products, which is obvious
  - Retail channels spend almost 1/3rd on grocery products. It would be a good decision to bring some offer on board for retailers to increase the sale.
  - This idea goes hand in hand with the previous insight as well.

- If we look at the data region wise, other tends to spend the most whereas Oporto has the least expenditure. Looking at the **COV for total expenditure Lisbon (.65) <Oporto (.73) < Other (.82)** Keeping in mind the inconsistency, the wholesaler can target customers from Lisbon.

- Lisbon spends a median of 25835, Oporto: 26953, Other 28029

- Looking at the customer numbers from Oporto and the median expenditure, it can be inferred that Oporto has a significant business potential (as there are only 47 customers). Keeping in mind the COV, there might be some rich class customers living in Oporto spending Lavishly.

- Comparing the expenditure based on products across region we can have the following insights: -
  - Lisbon customer base tend to spend more on Fresh Products
  - Oporto customer base spends more on Fresh products and Grocery.
  - While others spend more on Fresh and then Grocery.

- Business wise it would be great if the wholesaler could ensure free delivery of products in **LISBON and OPORTO**, as the customer base is quite a few. This could attract more customers.

- Looking at the customer distribution across region and channel we can derive the following: -
  - Lisbon has 59 hotel and 18 Retail customers
  - Oporto has 28 hotel and 19 Retail customers
  - Other regions have 211 hotel and 105 retail customers.

- Looking at the median expenditure, the retails spend more in Lisbon in spite of being less in no. Also, the COV for Lisbon was the least (.65). So, to add on to previous inference of adding to retail clients, the wholesaler should target **specifically retail clients from Lisbon**.

```
df_cross_med = pd.pivot_table(df, 'Total Spend' , index = ['Region'],
                              columns = ['Channel'], aggfunc = 'median')
df_cross_med
```

Channel	Hotel	Retail
Region		
Lisbon	20245	35324
Oporto	20867	29875
Other	18609	30124

- Hotels from Oporto should also be targeted as the spending capacity is much more there when compared to Lisbon. If we look at COV for OPORTO its quite high (.73). This brings to the fact that there are some really rich hotels in OPORTO ready to spend more. Specifically, those can be targeted.

- Observing the total spend trends it can be seen that more customers are sending in the range of **0 to 20000**, based on products, whereas there are almost none on the higher side. A customer should be labelled as per the spending capacity on order to tap more sales. For e.g. the customers can be labelled in 3 categories: **low expenditure medium expenditure and high expenditure**. In such a situation a customer labelled as high expenditure could be targeted with costly products in order to gain an increase revenue.

PROBLEM 2

Problem Statement: -

The Student News Service at Clear Mountain State University (CMSU) has decided to gather data about the undergraduate students that attend CMSU. CMSU creates and distributes a survey of 14 questions and receives responses from 62 undergraduates (stored in the ***Survey*** data set).

SOLUTION

The first step would be reading the data and storing it in a suitable variable say df. Next the head and tail can be seen using df.head() and df.tail() commands.

```
df = pd.read_csv('Survey-1.csv')
```

```
df.head()
```

	ID	Gender	Age	Class	Major	Grad Intention	GPA	Employment	Salary	Social Networking	Satisfaction	Spending	Computer	Text Messages
0	1	Female	20	Junior	Other	Yes	2.9	Full-Time	50.0	1	3	350	Laptop	200
1	2	Male	23	Senior	Management	Yes	3.6	Part-Time	25.0	1	4	360	Laptop	50
2	3	Male	21	Junior	Other	Yes	2.5	Part-Time	45.0	2	4	600	Laptop	200
3	4	Male	21	Junior	CIS	Yes	2.5	Full-Time	40.0	4	6	600	Laptop	250
4	5	Male	23	Senior	Other	Undecided	2.8	Unemployed	40.0	2	4	500	Laptop	100

Table 2.1

Extracting information about the data set using df.info ().

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 62 entries, 0 to 61
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   ID                 62 non-null     int64
 1   Gender             62 non-null     object
 2   Age                62 non-null     int64
 3   Class              62 non-null     object
 4   Major              62 non-null     object
 5   Grad Intention     62 non-null     object
 6   GPA                62 non-null     float64
 7   Employment         62 non-null     object
 8   Salary             62 non-null     float64
 9   Social Networking  62 non-null     int64
 10  Satisfaction       62 non-null     int64
 11  Spending           62 non-null     int64
 12  Computer           62 non-null     object
 13  Text Messages      62 non-null     int64
dtypes: float64(2), int64(6), object(6)
```

Using df.info() we can see that there are 62 non null values for all the column variables. Gender, Class, Major, Grad Intention, Employment, and computer are categorical variables while the rest are numerical.

We can also check the shape of the data frame and null values present if any, using df.shape and df.isnull().sum().

```
df.shape
```

```
(62, 14)
```

```
df.isnull().sum()
```

```
ID                  0
Gender              0
Age                 0
Class               0
Major               0
Grad Intention      0
GPA                 0
Employment          0
Salary              0
Social Networking   0
Satisfaction        0
Spending            0
Computer            0
Text Messages       0
dtype: int64
```

The data has 62 rows and 14 columns, with zero null values.

Moving further we can have a look at the categorical variables using .unique() function. The following code snippet shows the values of categorical variables.

```
print('Gender: ',df.Gender.unique())
print('\nClass: ',df.Class.unique())
print('\n Major: ', df.Major.unique())
print('\nGrad Intention: ',df['Grad Intention'].unique())
print('\nEmployment: ',df.Employment.unique())
print('\nComputer: ',df.Computer.unique())
```

```
Gender:  ['Female' 'Male']

Class:  ['Junior' 'Senior' 'Sophomore']

 Major:  ['Other' 'Management' 'CIS' 'Economics/Finance' 'Undecided'
 'International Business' 'Retailing/Marketing' 'Accounting']

Grad Intention:  ['Yes' 'Undecided' 'No']

Employment:  ['Full-Time' 'Part-Time' 'Unemployed']

Computer:  ['Laptop' 'Tablet' 'Desktop']
```

2.1. For this data, construct the following contingency tables (Keep Gender as row variable)

2.1.1. Gender and Major

The pd.crosstab() function with **margins = True** can be used to create the required contingency table. Margins = True includes a column **All** which holds the sum across rows and columns.

```
df_gen_maj = pd.crosstab(df.Gender, df.Major,margins='True')
```

```
df_gen_maj
```

Major Gender	Accounting	CIS	Economics/Finance	International Business	Management	Other	Retailing/Marketing	Undecided	All
Female	3	3	7	4	4	3	9	0	33
Male	4	1	4	2	6	4	5	3	29
All	7	4	11	6	10	7	14	3	62

Table 2.2

2.1.2. Gender and Grad Intention

```
df_gen_gradint = pd.crosstab(df.Gender, df['Grad Intention'],margins = True)
df_gen_gradint
```

Grad Intention Gender	No	Undecided	Yes	All
Female	9	13	11	33
Male	3	9	17	29
All	12	22	28	62

Table 2.3

2.1.3. Gender and Employment

```
df_gen_emp = pd.crosstab(df.Gender, df['Employment'], margins = 'True')
df_gen_emp
```

Employment Gender	Full-Time	Part-Time	Unemployed	All
Female	3	24	6	33
Male	7	19	3	29
All	10	43	9	62

Table 2.4

2.1.4. Gender and Computer

```
df_gen_comp = pd.crosstab(df.Gender, df.Computer, margins = 'True')
df_gen_comp
```

Computer	Desktop	Laptop	Tablet	All
Gender				
Female	2	29	2	33
Male	3	26	0	29
All	5	55	2	62

Table 2.5

2.2. Assume that the sample is representative of the population of CMSU. Based on the data, answer the following question:

2.2.1. What is the probability that a randomly selected CMSU student will be male?

It's a very straight forward question. We need to calculate probability of selecting a male student. It can be simply calculated as a ratio of total males and total students.

Following a simple approach, we can find the total no of students categorized as Male and Female using the value_counts() function applied over the Gender column.

```
df.Gender.value_counts()
```

```
Female    33
Male      29
Name: Gender, dtype: int64
```

Now we know the total males and also the total no of students which is simply 33+29, we can easily calculate the probability of selecting a random male student.

```
print('Probability that a randomly selected student is a male is = ', 29/62)
```

```
Probability that a randomly selected student is a male is =  0.46774193548387094
```

As calculated the probability that a randomly selected student is male is .4677 or 46.77%.

2.2.2. What is the probability that a randomly selected CMSU student will be female?

The required probability in this can be calculated similar to the previous question. The only difference being that, we need to find the probability for a female student. The total no of females in this case are **33.**

```
print('Probability that a randomly selected student is a female is = ', 33/62)
```

```
Probability that a randomly selected student is a female is =  0.532258064516129
```

The required probability that a randomly selected student is a female is .5322 or 53.22%

2.3. Assume that the sample is representative of the population of CMSU. Based on the data, answer the following question:

2.3.1. Find the conditional probability of different majors among the male students in CMSU.

Using the contingency table to solve this problem would be an ideal approach.

```
df_gen_maj
```

Major Gender	Accounting	CIS	Economics/Finance	International Business	Management	Other	Retailing/Marketing	Undecided	All
Female	3	3	7	4	4	3	9	0	33
Male	4	1	4	2	6	4	5	3	29
All	7	4	11	6	10	7	14	3	62

Table 2.6

Calculation of the required conditional probability can be understood as follows: -

P (Major | Male) = P (Major and Male) / P(Male)

It can also be put as: P(Major|Male) = n (Major and Male) / n(Male), here n stands the count value. It can be clearly seen that the total no of males is 29, so the denominator is fixed. The intersection part for each major and male intersection would change as we move across the row Male. Instead of computing the value for each major separately, for loop can be used. The list of majors is stored in a variable for traversing across row Male.

```
majors = df.Major.unique().tolist()
majors = sorted(majors)
majors
```

```
['Accounting',
 'CIS',
 'Economics/Finance',
 'International Business',
 'Management',
 'Other',
 'Retailing/Marketing',
 'Undecided']
```

```
for i in majors:
    print('Conditional Probability for major',i,'is: ' ,df_gen_maj.loc['Male'][i]/29,'\n')
```

```
Conditional Probability for major Accounting is:  0.13793103448275862

Conditional Probability for major CIS is:  0.034482758620689655

Conditional Probability for major Economics/Finance is:  0.13793103448275862

Conditional Probability for major International Business is:  0.06896551724137931

Conditional Probability for major Management is:  0.20689655172413793

Conditional Probability for major Other is:  0.13793103448275862

Conditional Probability for major Retailing/Marketing is:  0.1724137931034483

Conditional Probability for major Undecided is:  0.10344827586206896
```

The above results show the conditional probability of different majors among males. The values are tabulated below: -

Accounting	.1379
CIS	.0344
Economics/Finance	.1379
International Business	.0689
Management	.2068
Other	.1379
Retailing/Market	.1724
Undecided	.1034

Table 2.7

2.3.2 Find the conditional probability of different majors among the female students of CMSU.

Following the same approach as the previous question, this time the condition is for females.

P (Major | Female) = P (Major and Female) / P(Female)

P (Major | Female) = n (Major and Female)/n(Female)

Having a look at the contingency table: -

Major	Accounting	CIS	Economics/Finance	International Business	Management	Other	Retailing/Marketing	Undecided	All
Gender									
Female	3	3	7	4	4	3	9	0	33
Male	4	1	4	2	6	4	5	3	29
All	7	4	11	6	10	7	14	3	62

Table 2.8

The total no of females is **33**, that's the only change here rest the approach is similar to that of the previous question.

```
for i in majors:
    print('Conditional Probability of a female opting  major',i,'is: ', df_gen_maj.loc['Female'][i]/33,'\n')
```

```
Conditional Probability of a female opting  major Accounting is:  0.09090909090909091

Conditional Probability of a female opting  major CIS is:  0.09090909090909091

Conditional Probability of a female opting  major Economics/Finance is:  0.21212121212121213

Conditional Probability of a female opting  major International Business is:  0.12121212121212122

Conditional Probability of a female opting  major Management is:  0.12121212121212122

Conditional Probability of a female opting  major Other is:  0.09090909090909091

Conditional Probability of a female opting  major Retailing/Marketing is:  0.2727272727272727

Conditional Probability of a female opting  major Undecided is:  0.0
```

The conditional Probability for each major for a female is tabulated below.

Accounting	.0909
CIS	.0909
Economics/Finance	.2121
International Business	.1212
Management	.1212
Other	.0909
Retailing/Market	.2727
Undecided	0

Table 2.9

2.4. Assume that the sample is a representative of the population of CMSU. Based on the data, answer the following question:

2.4.1. Find the probability That a randomly chosen student is a male and intends to graduate.

The contingency table for Gender and Employment would give us the answer. The required probability is: P (Male and Intends to Graduate). It's a simple case of joint probability.

Grad Intention	No	Undecided	Yes	All
Gender				
Female	9	13	11	33
Male	3	9	17	29
All	12	22	28	62

Table 2.10

The intersection of Male and Intend to graduation is **17 and the total students is 62.**

```
print('Prob that a randomly selected student is a male and intends to graduate: ',17/62)
```

```
Prob that a randomly selected student is a male and intends to graduate:  0.27419354838709675
```

The probability that a selected student is male and intends to graduate is .27 or 27.419%.

2.4.2 Find the probability that a randomly selected student is a female and does NOT have a laptop

The answer lies in the gender and computer contingency table.

Computer	Desktop	Laptop	Tablet	All
Gender				
Female	2	29	2	33
Male	3	26	0	29
All	5	55	2	62

Table 2.11

The required probability here is P (Female and No Laptop). We need to find no of females with no laptop.

```
fem_no_lap = df_gen_comp.loc['Female']['All'] - df_gen_comp.loc['Female']['Laptop']
fem_no_lap
```

```
4
```

As the code snippet tells, the no of females with no laptops are **4** in no. against a total of **62.**

```
print('Probability that a student selected at random is a female and does not have a laptop is: ',
      fem_no_lap/62)
```

```
Probability that a student selected at random is a female and does not have a laptop is:  0.06451612903225806
```

Further calculating the probability, the probability that a randomly selected student is female and does not have a laptop is .064 or 6.45%.

2.5. Assume that the sample is representative of the population of CMSU. Based on the data, answer the following question:

2.5.1. Find the probability that a randomly chosen student is either a male or has full-time employment?

The required probability in this case is: -

P (Male or Full time employed) = P(Male) + P (Full time Employed) – P (Male and Full time employed)

Looking at the gender and employment contingency table: -

Employment	Full-Time	Part-Time	Unemployed	All
Gender				
Female	3	24	6	33
Male	7	19	3	29
All	10	43	9	62

Table 2.12

Either a male or has full time employment: - A : Prob of a male student B : Prob of Full time employment P(A or B) = P(A) + P(B) - P(A and B)

```
A = 29/62
B = 10/62
AB = 7/62
print('Probability that a randomly chosen student is either a male or has full-time employment:', A+B-AB)
```

```
Probability that a randomly chosen student is either a male or has full-time employment: 0.5161290322580645
```

The code snippet clearly calculates the desired probability. All the probabilities are calculated from the contingency table. P(Male) = no of Males / Total students, P(Full Time employed) = no of Full time employed / Total students, and P(Male and Full time) = (Intersection of Male and Full time) / Total.

The Probability that a randomly selected student is either a male or has full time employment is: 5161 or 51.61%.

2.5.2. Find the conditional probability that given a female student is randomly chosen, she is majoring in international business or management.

The required probability is: **P (International Business or Management | Female)** which can be calculated as: -

P (International Business | Female) + P (Management | Female)

Since both the events are mutually exclusive, the intersection part is 0.

Following the contingency table: -

Major Gender	Accounting	CIS	Economics/Finance	International Business	Management	Other	Retailing/Marketing	Undecided	All
Female	3	3	7	4	4	3	9	0	33
Male	4	1	4	2	6	4	5	3	29
All	7	4	11	6	10	7	14	3	62

Table 2.13

Using the above table, we can easily calculate the required conditional probabilities and add them up to get the answer. **For P (International business| Female) = n (Female and International business)/n(Females)**. Similarly, **P (Females | Management) = n (Females and Management)/n(Females)**. The values are found using the above table.

```
# P(F|International business)
p_f_int_bus = 4/33
```

```
# P(F|Management)
p_f_mngmnt = 4/33
```

```
print('Conditional probability for the given condition is: ',p_f_int_bus + p_f_mngmnt)
```

```
Conditional probability for the given condition is:  0.24242424242424243
```

The Required conditional probability is: .**24 or 24.24%.**

2.6. Construct a contingency table of Gender and Intent to Graduate at 2 levels (Yes/No). The Undecided students are not considered now and the table is a 2x2 table. Do you think the graduate intention and being female are independent events?

Creating a new data frame df1 which does not include the students with undecided graduation intent. Further creating a cross tab using pd.crosstab() applied to df1.

```
df1 = df[df['Grad Intention']!="Undecided"]
```

```
df_grad_int = pd.crosstab(df1.Gender, df1['Grad Intention'], margins = True)
df_grad_int
```

Grad Intention	No	Yes	All
Gender			
Female	9	11	20
Male	3	17	20
All	12	28	40

Table 2.14

To check if the given two events are independent, we need to confirm the following: -

If P (Yes and Female) = P (Yes). P(Female), we can say that the two events are independent.

```
prob_f_yes = 11/40
prob_f = 9/40
prob_yes = 28/40
```

```
if prob_f_yes == prob_f * prob_yes:
    print('Independent')
else:
    print('Not Independent')
```

```
Not Independent
```

As calculated, the two events are **not Independent.**

2.7. Note that there are four numerical (continuous) variables in the data set, GPA, Salary, Spending, and Text Messages.

Answer the following questions based on the data

2.7.1. If a student is chosen randomly, what is the probability that his/her GPA is less than 3?

The approach is to count the total no. of students with GPA<3 and then divide by total no of students.

Count() function can be used to calculate the total no of students in the GPA columns. To calculate the students with GPA less than three conditionality can be applied on column GPA.

```
tot_stud = df.GPA.count()
```

```
less_than_3 = len(df[df['GPA']<3])
```

```
print('Probability that a random student has GPA less than 3 is: ', less_than_3/tot_stud)
```

```
Probability that a random student has GPA less than 3 is:  0.27419354838709675
```

Here the required probability that a random student has GPA less than 3 is .2741 or 27.41 %

2.6.2. Find the conditional probability that a randomly selected male earns 50 or more. Find the conditional probability that a randomly selected female earns 50 or more.

In the first case it is asked to calculate the conditional probability that a randomly selected male earns more than 50. The probability can be formulated as: -

P (Earns>=50 | Male) = n (Earns > =50 and Male / n (Male)

```
total_males = len(df[df.Gender=="Male"])
total_males
```

```
29
```

The total no of males = 29. Finding no of males earning equal or more than 50.

```
# Male and earning more than 50
len(df[(df.Gender=="Male")&(df.Salary>=50)])
```

```
14
```

Male and earning more than 50 = 14.

```
print('The proability that a randomly selcted male earning 50 or more is: ',14/29)
```

```
The proability that a randomly selcted male earning 50 or more is:  0.4827586206896552
```

The required probability that a randomly selected male earning 50 or more is .4827 or 48.27%

Similarly finding the same for females.

P (earns >= 50 | Female) = n (earns >= 50 and Female) / n (Female)

```
total_females = len(df[df.Gender=="Female"])
total_females
```

```
33
```

Total Females = 33.

```
# Female and earning more than 50
len(df[(df.Gender=="Female")&(df.Salary>=50)])
```

```
18
```

Females earning 50 or more: 18.

```
print('Probability that a randomly selected female earns 50 or more is : ',18/33)
```

```
Probability that a randomly selected female earns 50 or more is :  0.5454545454545454
```

The required probability that a randomly selected female earns 50 or more is .5454 or 54.54

2.8. Note that there are four numerical (continuous) variables in the data set, GPA, Salary, Spending, and Text Messages. For each of them comment whether they follow a normal distribution. Write a note summarizing your conclusions.

To start with a new data frame is created which stores only the continuous variables. It is further transformed to describe data frame using describe() function. Further columns IQR and COV are added to check consistency and spread. Further a column of skewness is added.

```
df_cont = df.loc[:][['GPA','Salary','Spending','Text Messages']]
```

```
df_cont.head()
```

	GPA	Salary	Spending	Text Messages
0	2.9	50.0	350	200
1	3.6	25.0	360	50
2	2.5	45.0	600	200
3	2.5	40.0	600	250
4	2.8	40.0	500	100

Tale 2.15

```
df_cont = df_cont.describe().T
```

```
df_cont.loc[:,'IQR'] = df_cont['75%'] - df_cont['25%']
```

```
df_cont.loc[:,'COV'] = df_cont['std']/df_cont['mean']
```

```
skewness = [stats.skew(df.GPA), stats.skew(df.Salary), stats.skew(df.Spending),
            stats.skew(df['Text Messages'])]
```

```
df_cont.loc[:,'Skewness'] = skewness
df_cont
```

The skewness is calculated using the stats.skew() method.

	count	mean	std	min	25%	50%	75%	max	IQR	COV	Skewness
GPA	62.0	3.129032	0.377388	2.3	2.9	3.15	3.4	3.9	0.5	0.120609	-0.306937
Salary	62.0	48.548387	12.080912	25.0	40.0	50.00	55.0	80.0	15.0	0.248843	0.521677
Spending	62.0	482.016129	221.953805	100.0	312.5	500.00	600.0	1400.0	287.5	0.460470	1.547285
Text Messages	62.0	246.209677	214.465950	0.0	100.0	200.00	300.0	900.0	200.0	0.871070	1.264245

Table 2.16

To look at the distribution we can use the sns.barplot() method with kde = True which would give a visualization about the skewness as well.

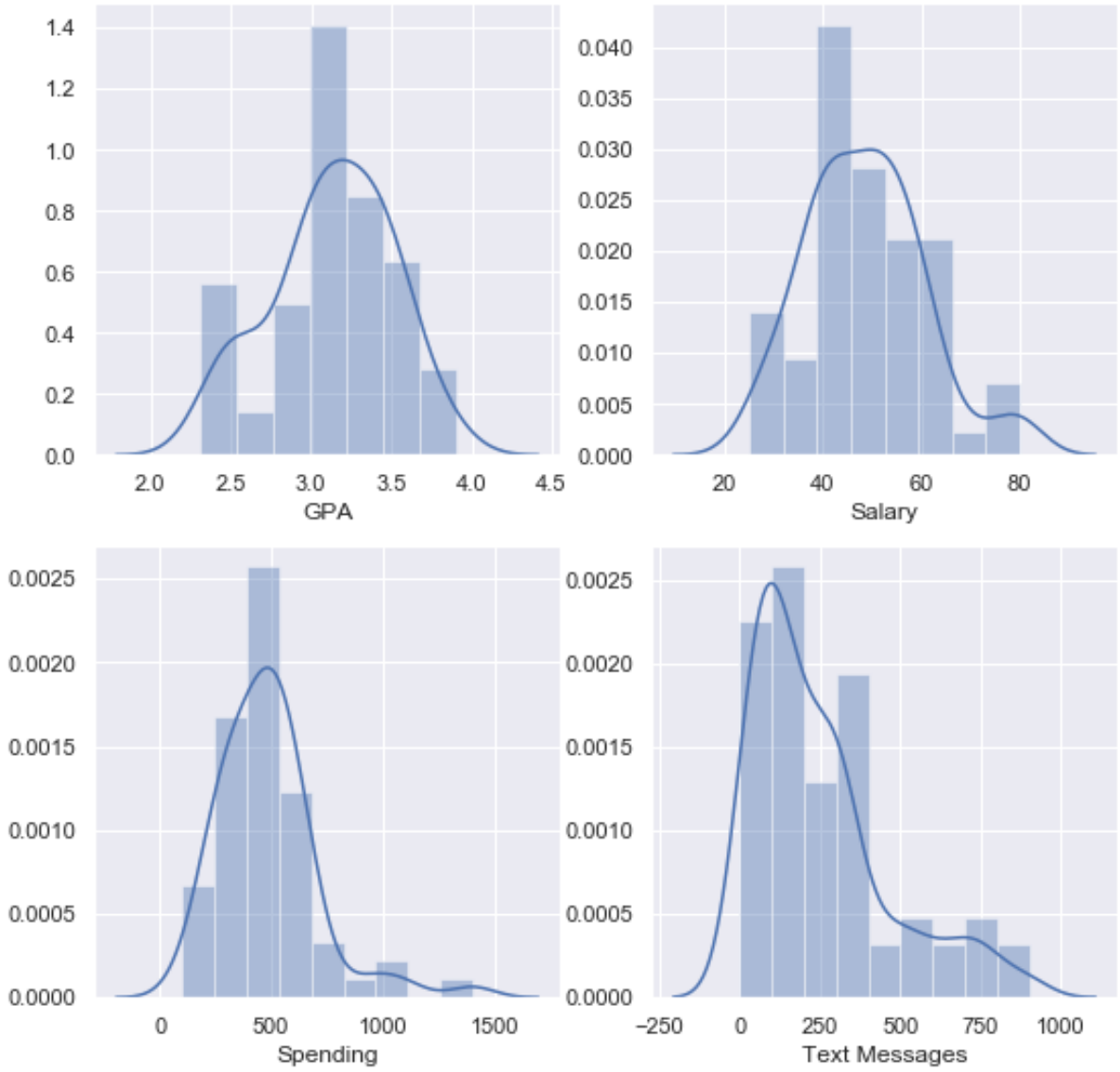

Fig 2.1

The **GPA** variable seems to be distributed Normally which is evident from the fact that it has a skewness of -.3. It is a bit left skewed. It is also very consistent with a COV of .12. Being a bit left skewed, median is greater than mean. Mode lies around 3.25.

Salary would have been completely normally distributed if there were no distortions towards right. Still it tends to a normal distribution, with a slightly right skewness (skewness = .52). This tells us that mean salary is greater than median. Mode lies between 40-60. The COV is also a bit on the higher side (.24) compared to GPA.

The **Spending** variable is highly right skewed. The skewness is 1.54. Mean Spending would be much higher than the median spending. Mode would lie near 500. COV increases a bit (.46).

The **Text Messages** is also right skewed. It has a skewness value of 1.26. Mean is greater than Median and the mode lies somewhere around 125. The COV is the highest for this variable (.87) indicating the max inconsistency.

Checking the empirical rule

The empirical rule says for a normal distribution 68% values lie in the range of first standard deviation from the mean, μ – σ and μ + σ, 95% of values lie in the range of two standard deviations from the mean, μ - 2σ and μ + 2σ, and 99.7% of values fall within first three standard deviations from the mean μ - 3σ and μ + 3σ.

For the given continuous variables, GPA and Salary are the closest to normal distribution. To check the empirical rule we can use the stat.norm.cdf() function. The limits will be μ – σ and μ + σ.

Test for GPA: -

```
gpa_lower_lim = df_cont.loc['GPA']['mean'] - df_cont.loc['GPA']['std']
gpa_upper_lim = df_cont.loc['GPA']['mean'] + df_cont.loc['GPA']['std']
```

```
# According to emperical rule 68% of data shoul lie within
# mean+-sigma limits
# can be done using stats.norm.cdf
print('Emperical test for GPA')
print(stats.norm.cdf(gpa_upper_lim,3.129032,0.377388) -
      stats.norm.cdf(gpa_lower_lim,3.129032,0.377388))
```

```
Emperical test for GPA
0.6826899957094594
```

As found out around **68.26 %** values lie in the given limit, which testifies the empirical rule.

Test for Salary: -

```
sal_upper_lim = df_cont.loc['Salary']['mean'] + df_cont.loc['Salary']['std']
sal_lower_lim = df_cont.loc['Salary']['mean'] - df_cont.loc['Salary']['std']
```

```
print('Emperical test for Salary: - ')
stats.norm.cdf(sal_upper_lim,48.548387,12.080912)-
stats.norm.cdf(sal_lower_lim,48.548387,12.080912)
```

```
Emperical test for Salary: -

0.682689500803201
```

For salary too in the **68.26 %** values fall within the given range, which proves the empirical rule.

PROBLEM 3

Problem Statement:

An important quality characteristic used by the manufacturers of ABC asphalt shingles is the amount of moisture the shingles contain when they are packaged. Customers may feel that they have purchased a product lacking in quality if they find moisture and wet shingles inside the packaging. In some cases, excessive moisture can cause the granules attached to the shingles for texture and colouring purposes to fall off the shingles resulting in appearance problems. To monitor the amount of moisture present, the company conducts moisture tests. A shingle is weighed and then dried. The shingle is then reweighed, and based on the amount of moisture taken out of the product, the pounds of moisture per 100 square feet are calculated. The company would like to show that the mean moisture content is less than 0.35 pound per 100 square feet.

SOLUTION

To start with importing the data set in a data frame df and checking the sample using head() function. Further, using info() and isnull() function more details can be extracted.

Data at a glance: -

```
df = pd.read_csv('A+&+B+shingles.csv')
df.head()
```

	A	B
0	0.44	0.14
1	0.61	0.15
2	0.47	0.31
3	0.30	0.16
4	0.15	0.37

Table 3.1

As stated in the problem, the two columns A and B hold the moisture content value.Checking info () using df.info(): -

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36 entries, 0 to 35
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   A       36 non-null     float64
 1   B       31 non-null     float64
dtypes: float64(2)
memory usage: 704.0 bytes
```

Checking for NULL values if any: -

```
df.isnull().sum()
```

```
A    0
B    5
dtype: int64
```

As we can see, there are **5 null values in column B**.

3.1 Do you think there is evidence that mean moisture contents in both types of shingles are within the permissible limits? State your conclusions clearly showing all steps.

To check the same, we need to define our null and alternate hypothesis first. The company's current status quo is that the mean moisture content is less than .35. This will form the basis of our NULL Hypothesis. If we need to refute this claim, the alternate hypothesis would be mean moisture content> .35.

NULL HYPOTHESIS: H0: μ =0.35

ALTERNATE HYPOTHESIS: **HA: μ >0.35**

We need to check the permissible levels for both samples A and B. We need to create separate data frames storing values of Column A and B separately.

Next on each of the individual values, we would be applying one sample t test, with alpha = .05.

```
dfA = df.A
dfB = df.B
```

For Shingles A: -

Since our alternate hypothesis states that HA > .35, it's a right tailed test. Using the function t_test_1samp we can find the value of t stats and p value. One thing to note here is that we are performing a one tailed test here and the p value returned by the function is for two tailed. So, we need to divide the p value by two to get the required value for one tailed test.

```
alpha = .05
```

```
t_statistic, p_value = ttest_1samp(dfA, .35)
```

```
print('\n P value: ', p_value/2,'\n T stat: ', t_statistic)
```

```
 P value:  0.07477633144907513
 T stat:  -1.4735046253382782
```

P value: .074

T stats: -1.47

Alpha: .05

We have taken the value of alpha as .05, and as the P value obtained here is more than alpha, **we fail to reject the null hypothesis.**

Conclusion:

- P value>Alpha, at 95% confidence level, we **FAILED TO REJECT THE NULL HYPOTHESIS**
- We can conclude that since at 95% confidence level we have failed to reject the null hypothesis we can say that the moisture content in Shingles A is within permissible limits.

For Shingles B: -

Since we know there are some missing values in the shingles B data set, we can use the parameter nan_policy = omit parameter to omit the missing values. The p value needs to be divided by two, in order to get value for a one tailed test.

```
t_statistic_B, p_value_B = ttest_1samp(dfB, .35, nan_policy='omit')
```

```
print('P value = ', p_value_B/2, '\n T stats: ', t_statistic_B)
```

```
P value =  0.0020904774003191826
 T stats:  -3.1003313069986995
```

P value: .002

T stats: -3.1003

Alpha: .05

Here the P value is less than alpha, hence **we can reject the null hypothesis with 95% confidence**.

Conclusion:

- P value < alpha, hence at 95% confidence we **REJECT THE NULL HYPOTHESIS**.
- Thus, we have evidence at 95% confidence that the mean moisture in shingles **B > .35 i.e. not within permissible limits.**

3.2 Do you think that the population mean for shingles A and B are equal? Form the hypothesis and conduct the test of the hypothesis. What assumption do you need to check before the test for equality of means is performed?

The problem can be solved by applying a **two-sample t test** on the dataset. It can be done using the method ttest_ind(). Let's start by defining the NULL and ALTERNATIVE hypothesis

NULL Hypothesis: μA = μB

ALTERNATE Hypothesis: μA ≠ μB

The only assumption we need here is the value of alpha. Also, one thing to notice here is that it is a two tailed test hence no need to divide the p value by 2.

Considering **alpha = .05**

Performing the 2-sample t test: -

```
t_statistic_2_sam, p_value_2_sam = ttest_ind(df.A, df.B, nan_policy='omit')
```

```
print('P value: ',p_value_2_sam,'\nT stats: ', t_statistic_2_sam)
```

```
P value:  0.2017496571835306
T stats:  1.2896282719661123
```

P value: .20

T stats: 1.28

Alpha: .05

It can be clearly seen the P value > alpha, hence **we fail to reject the null hypothesis.**

Conclusion:

- P value > alpha, therefore at 95% confidence level we fail to reject the null hypothesis.
- Hence, we can conclude that the **mean moisture content of Shingle A and Shingle B are equal.**