SQL & Python Small Business Data Cleaning and EDA

In this document, we show our analysis for the last two questions of this project and include plots to visualize our findings. As a refreshment, the last two questions in this EDA are:

7. Calculate the amount of deliveries for each month and each year.
8. Calculate how many deliveries the seller made each month and for what amount.

In [1]:

```
# importing packages
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [10]:

```
# # reading the .csv files
data_2019 = pd.read_excel(r"C:\Users\Mili\OneDrive\Documents\projects\coding\Busine
data_2020 = pd.read_excel(r"C:\Users\Mili\OneDrive\Documents\projects\coding\Busine
```

We first take a look at the data sets

In [11]:

```
data_2019.head()
```

Out[11]:

	Order number	Client ID	Product code	Date of delivery	Delivery amount
0	97058.0	7121.0	494843.0	2019-01-24	9565.0
1	2968.0	7167.0	111937.0	2019-01-29	18907.0
2	2968.0	7167.0	218889.0	2019-01-29	54132.0
3	45863.0	7136.0	495715.0	2019-02-07	28023.0
4	45863.0	7136.0	495716.0	2019-02-07	59120.0

In [13]:

```
data_2020.head()
```

Out[13]:

	Order number	Client ID	Product code	Date of delivery	Delivery amount
0	81318.0	7118.0	510984.0	2020-02-17	4771.0
1	29280.0	7138.0	510984.0	2020-02-19	34346.0
2	33418.0	7161.0	510984.0	2020-02-12	29026.0
3	81318.0	7118.0	510985.0	2020-02-17	37186.0
4	29280.0	7138.0	510985.0	2020-02-19	47580.0

7) Calculate the amount of deliveries for each month and each year

We first create pivot tables for each year where we include the mean and median delivery amount for each month.

In [15]:

```
# Answering question 7
data_2019['Month'] = data_2019['Date of delivery'].dt.month
data_2020['Month'] = data_2020['Date of delivery'].dt.month
# Find the mean and median delivery amount for each month and create a pivot table
pivot_2019 = data_2019.groupby('Month', as_index=False).agg({'Delivery amount':['me
pivot_2019
```

Out[15]:

	Month	Delivery amount	
		mean	median
0	1.0	27534.666667	18907.0
1	2.0	35677.272727	37466.0
2	3.0	40689.631579	40826.0
3	4.0	27219.529412	22065.0
4	5.0	36439.666667	40694.0
5	6.0	20957.000000	17516.0
6	7.0	54915.000000	54915.0
7	8.0	64893.000000	64893.0
8	9.0	26639.500000	26639.5
9	10.0	31758.727273	36369.0
10	12.0	33727.500000	42597.0

In [16]:

```
pivot_2020 = data_2020.groupby('Month', as_index=False).agg({'Delivery amount':['me
pivot_2020
```

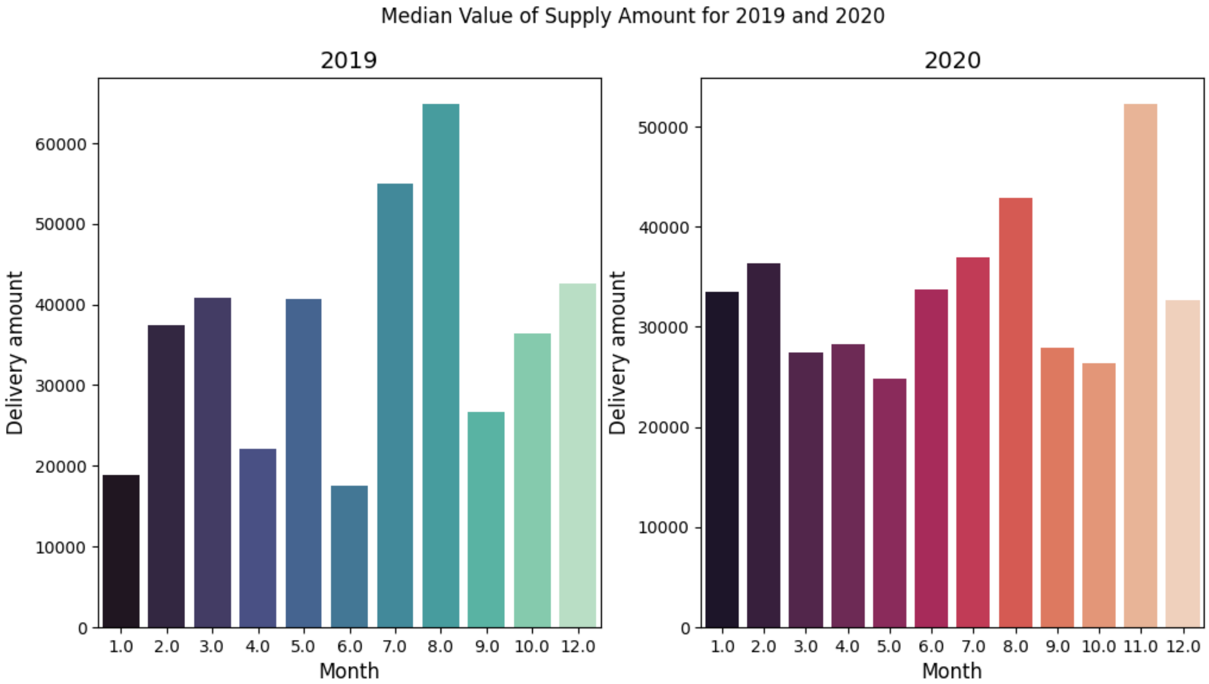
Out[16]:

	Month	Delivery amount	
		mean	median
0	1.0	32269.388889	33453.5
1	2.0	36417.900990	36392.0
2	3.0	30237.846154	27380.5
3	4.0	30941.346457	28267.0
4	5.0	26795.862069	24806.0
5	6.0	35112.833333	33687.0
6	7.0	37937.612500	36955.5
7	8.0	38144.854545	42860.0
8	9.0	30524.900000	27851.0
9	10.0	28478.500000	26398.0
10	11.0	52318.000000	52318.0
11	12.0	32221.333333	32666.5

Now using the calculated median from each pivot table, we plot it into the graph below for each year to determine if there is seasonality (find whether the data is cyclical).

In [37]:

```
fig, axs = plt.subplots(1, 2, figsize=(12, 6))
fig.suptitle('Median Value of Supply Amount for 2019 and 2020')
sns.barplot(x=pivot_2019['Month'], y=pivot_2019['Delivery amount', 'median'], ax=ax
axs[0].set_title('2019', fontsize=14)
axs[0].set_ylabel('Delivery amount', fontsize=12)
axs[0].set_xlabel('Month', fontsize=12)
sns.barplot(x=pivot_2020['Month'], y=pivot_2020['Delivery amount', 'median'], ax=ax
axs[1].set_title('2020', fontsize=14)
axs[1].set_ylabel('Delivery amount', fontsize=12)
axs[1].set_xlabel('Month', fontsize=12)
plt.show()
```



We can see that both graphs do not resemble each other, and that there are no visible patterns. Therefore, there is no seasonality.

8) Calculate how many deliveries the seller made each month and for what amount

We first obtain a summary of each data set:

In [20]:

```
# Answering question 8
data_2019.describe()
```

Out[20]:

	Order number	Client ID	Product code	Date of delivery	Delivery amount	M
count	542.000000	542.000000	542.000000	542	542.000000	542.00
mean	40986.944649	7137.667897	446526.763838	2019-04-03 00:31:52.915129088	35093.892989	3.40
min	2968.000000	7110.000000	111864.000000	2019-01-24 00:00:00	1267.000000	1.00
25%	21924.000000	7125.000000	497028.000000	2019-02-19 00:00:00	18886.750000	2.00
50%	40155.000000	7135.000000	497032.000000	2019-02-25 00:00:00	37466.000000	2.00
75%	50798.000000	7155.000000	497035.000000	2019-03-17 00:00:00	52643.750000	3.00
max	112601.000000	7167.000000	509369.000000	2019-12-26 00:00:00	65583.000000	12.00
std	27788.032290	18.422452	113410.076499	NaN	18956.664053	2.94

In [19]:

```
data_2020.describe()
```

Out[19]:

	Order number	Client ID	Product code	Date of delivery	Delivery amount	M
count	778.000000	778.000000	778.000000	778	1.048081e+06	778.00
mean	52321.366324	7137.523136	407363.991003	2020-05-23 07:53:49.820051456	3.355483e+04	5.20
min	3491.000000	7110.000000	111855.000000	2020-01-22 00:00:00	1.238000e+03	1.00
25%	33760.000000	7126.000000	237551.000000	2020-03-18 00:00:00	1.739400e+04	3.00
50%	52888.000000	7136.000000	511501.000000	2020-05-31 00:00:00	3.356000e+04	5.50
75%	65145.000000	7155.000000	518702.750000	2020-07-09 00:00:00	4.971000e+04	7.00
max	130090.000000	7167.000000	524468.000000	2020-12-24 00:00:00	6.589300e+04	12.00
std	29379.751231	15.984874	145409.383802	NaN	1.866508e+04	2.68

We can see that 2020 has a higher distribution for delivery amount compared to 2019. We are only interested in the month, Delivery amount and Data of delivery. We obtain summary statistics of Delivery Amount and Date of Delivery grouped by Month for 2019 and 2020:

In [26]:

```
#calculate the median of the number of deliveries and number of deliveries by month
med_2019 = data_2019.groupby(['Month'], as_index=False).agg({'Delivery amount':'med
#add in 2019 the data for November by the median of the year
med_2019.loc[11]=[11,37466.0, 15]
# do the same calculations to get the median number of delivaries and number of del
med_2020 = data_2020.groupby('Month', as_index=False).agg({'Delivery amount':'media
med_2019.describe()
```

Out[26]:

	Month	Delivery amount	Date of delivery
count	12.000000	12.000000	12.000000
mean	6.500000	36696.125000	46.416667
std	3.605551	14128.238898	104.030990
min	1.000000	17516.000000	2.000000
25%	3.750000	25495.875000	5.500000
50%	6.500000	37466.000000	16.500000
75%	9.250000	41268.750000	34.500000
max	12.000000	64893.000000	374.000000

In [27]:

```
med_2019
```

Out[27]:

	Month	Delivery amount	Date of delivery
0	1.0	18907.0	6.0
1	2.0	37466.0	374.0
2	3.0	40826.0	38.0
3	4.0	22065.0	34.0
4	5.0	40694.0	18.0
5	6.0	17516.0	6.0
6	7.0	54915.0	2.0
7	8.0	64893.0	2.0
8	9.0	26639.5	4.0
9	10.0	36369.0	22.0
10	12.0	42597.0	36.0
11	11.0	37466.0	15.0

In [28]:

```
med_2020.describe()
```

Out[28]:

	Month	Delivery amount	Date of delivery
count	12.000000	12.000000	12.000000
mean	6.500000	33586.250000	64.833333
std	3.605551	7893.848372	47.512359
min	1.000000	24806.000000	2.000000
25%	3.750000	27733.375000	26.750000
50%	6.500000	33060.000000	62.500000
75%	9.250000	36532.875000	85.250000
max	12.000000	52318.000000	156.000000

In [29]:

```
med_2020
```

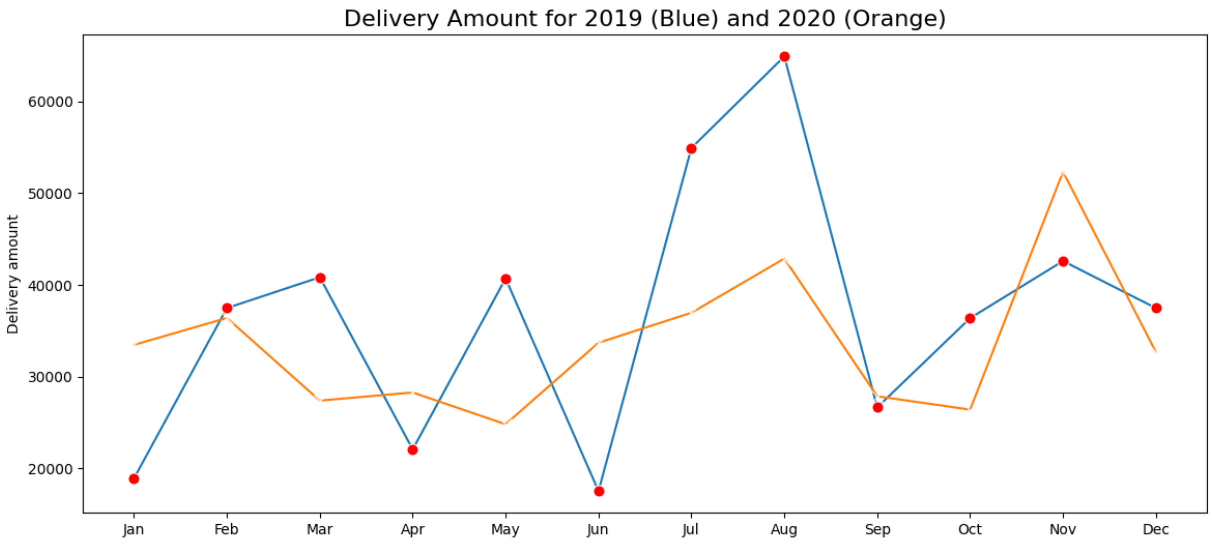
Out[29]:

	Month	Delivery amount	Date of delivery
0	1.0	33453.5	54
1	2.0	36392.0	101
2	3.0	27380.5	78
3	4.0	28267.0	127
4	5.0	24806.0	29
5	6.0	33687.0	156
6	7.0	36955.5	80
7	8.0	42860.0	55
8	9.0	27851.0	20
9	10.0	26398.0	70
10	11.0	52318.0	2
11	12.0	32666.5	6

We plot the delivery amount per month for each year:

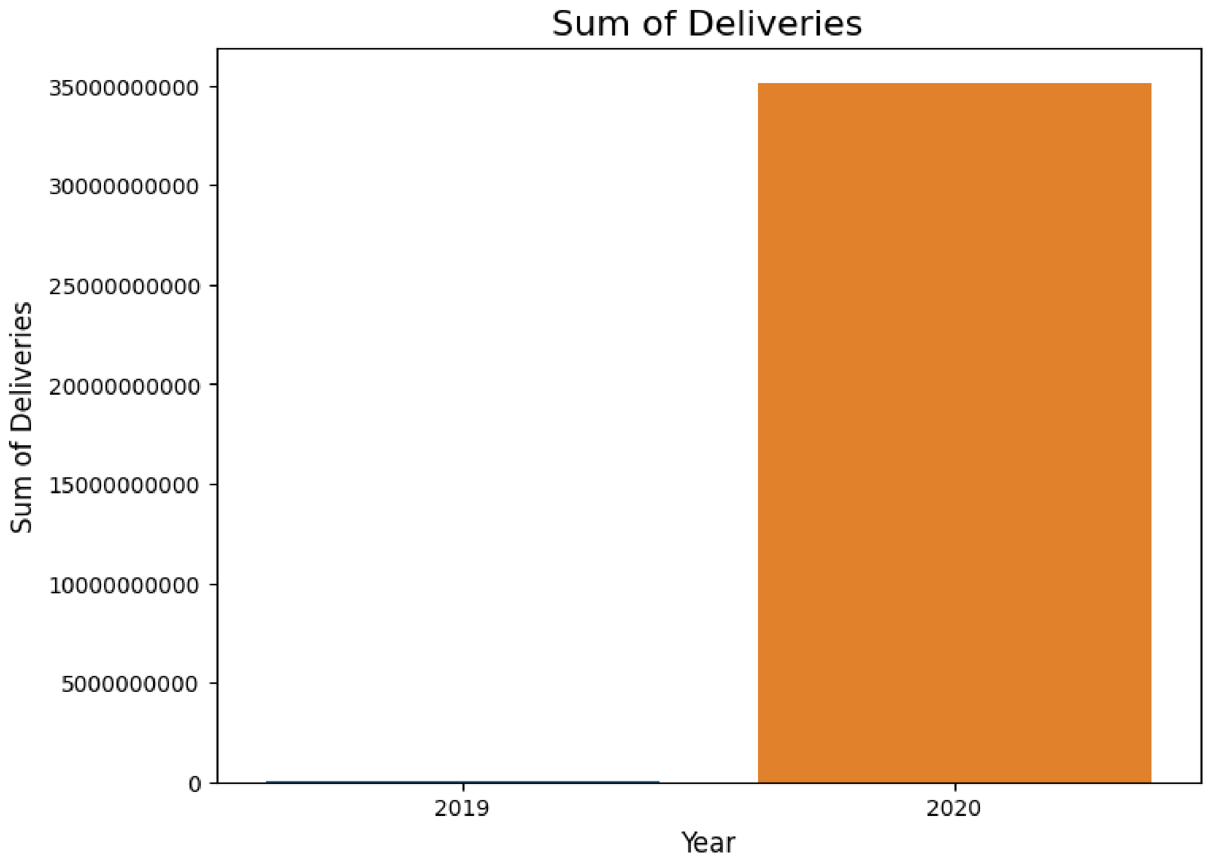
In [36]:

```
month = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov
plt.figure(figsize=(14, 6))
plt.title("Delivery Amount for 2019 (Blue) and 2020 (Orange)", fontsize=16)
sns.lineplot(y = med_2019['Delivery amount'], x=month, marker='o', markerfacecolor=
sns.lineplot(med_2020['Delivery amount'], marker='x', markerfacecolor='blue', marke
plt.show()
```



We can see in the plot above that there was a surge in delivery amount in the summer months of 2019 between June and September. On the other hand, 2020 had a surge in delivery amounts that was much smaller between October and December. We also plot the sum of deliveries for each year:

```
In [35]: plt.figure(figsize=(8, 6))
         sns.barplot(x = ['2019', '2020'], y=[data_2019['Delivery amount'].sum(), data_2020[
         plt.title("Sum of Deliveries", fontsize=16)
         plt.xlabel("Year", fontsize=12)
         plt.ylabel("Sum of Deliveries", fontsize=12)
         plt.ticklabel_format(style='plain', axis='y')
         plt.show()
```



The sum of deliveries by year is significantly smaller in 2019 compared to 2020. There could be factors as to why this is the case which could also explain why there is no seasonality which could be the price of individual items.