Question 1

In [5]:
```
import pandas as pd
import numpy as np
```

In [3]:
```
data=pd.read_csv('/Users/akshayamahesh/Downloads/ShopifyData.csv')
data.head()
```

Out[3]:

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

In [7]:
```
data.shape
```

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

In [8]:
```
data.isnull().sum()
```

Out[8]:
```
order_id          0
shop_id           0
user_id           0
order_amount      0
total_items       0
payment_method    0
created_at        0
dtype: int64
```

In [15]:
```
data.dtypes
```

Out[15]:
```
order_id           int64
shop_id            int64
user_id            int64
order_amount       int64
total_items        int64
payment_method    object
created_at        object
dtype: object
```

In [10]:
```
# counts the number of unique shops in our data
print("Numbers of unique shops: "+ str(len(data.shop_id.unique())))
```

```
Numbers of unique shops: 100
```

a. what could be going wrong with our calculation and a better way to evaluate this data

Let us analyse the `order_amount` column and analyse it's statistics. This will give us a better idea of how our dataset for this variable looks like

In [11]:
```
# calculating summary statistics of order_amount variable
data.order_amount.describe()
```

Out[11]:
```
count      5000.000000
mean       3145.128000
std       41282.539349
min          90.000000
25%         163.000000
50%         284.000000
75%         390.000000
max      704000.000000
Name: order_amount, dtype: float64
```

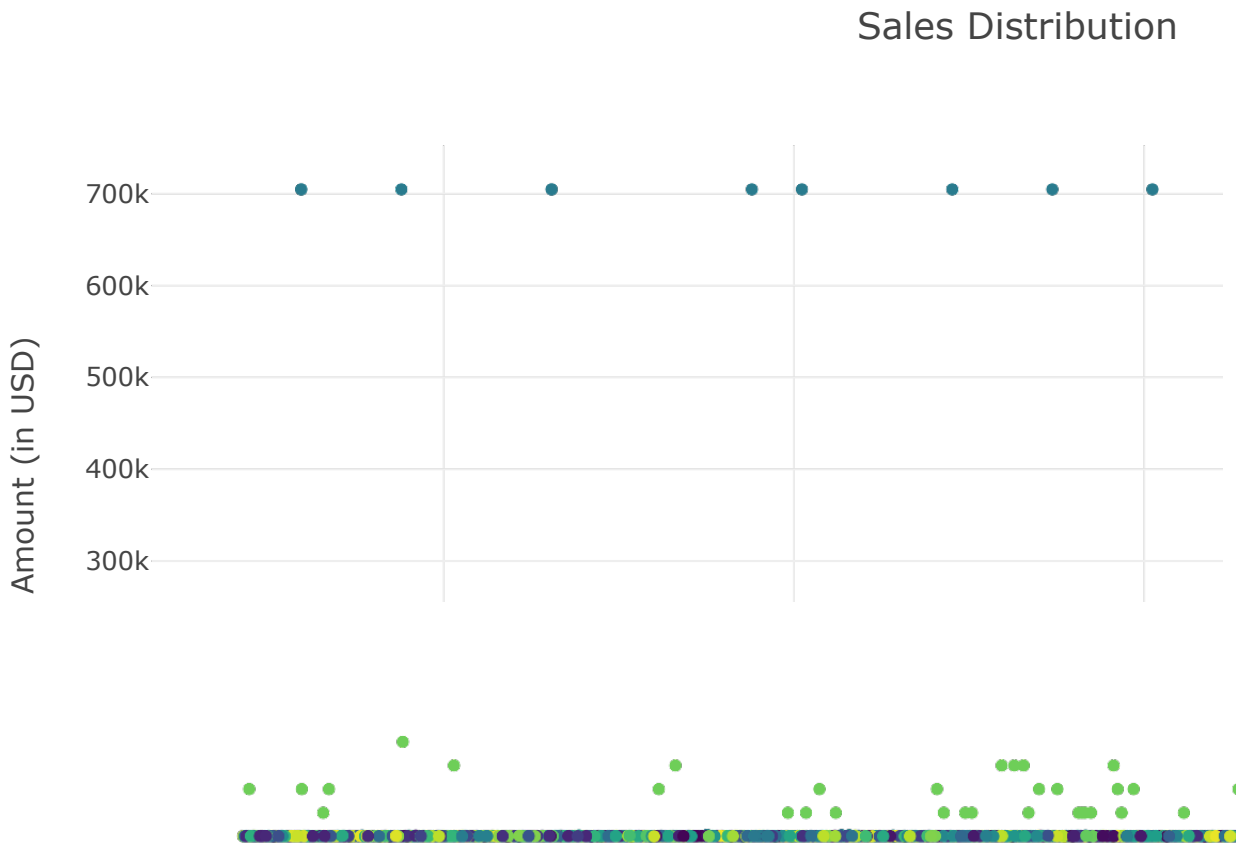
count gives the number of non-empty values. Therefore, we can cross verify this with 'data.isnull().sum()'
mean is 3145.128 USD
std here represents the standard deviation of the `order_amount` column. The value is 41282.53 USD which seems very large when compared to mean.
That is, we can find that there is a large spread of data and presence of potential ***outliers***.
Let us see if we can remove these outliers to see how our data exactly is.
Also, significant amount of data i.e, 50 percentile or the median is 284.00 USD (50% of data seems to be below 284.00 and 75% of data is below 390.00 USD)

In [23]:
```
#pip install plotly
```

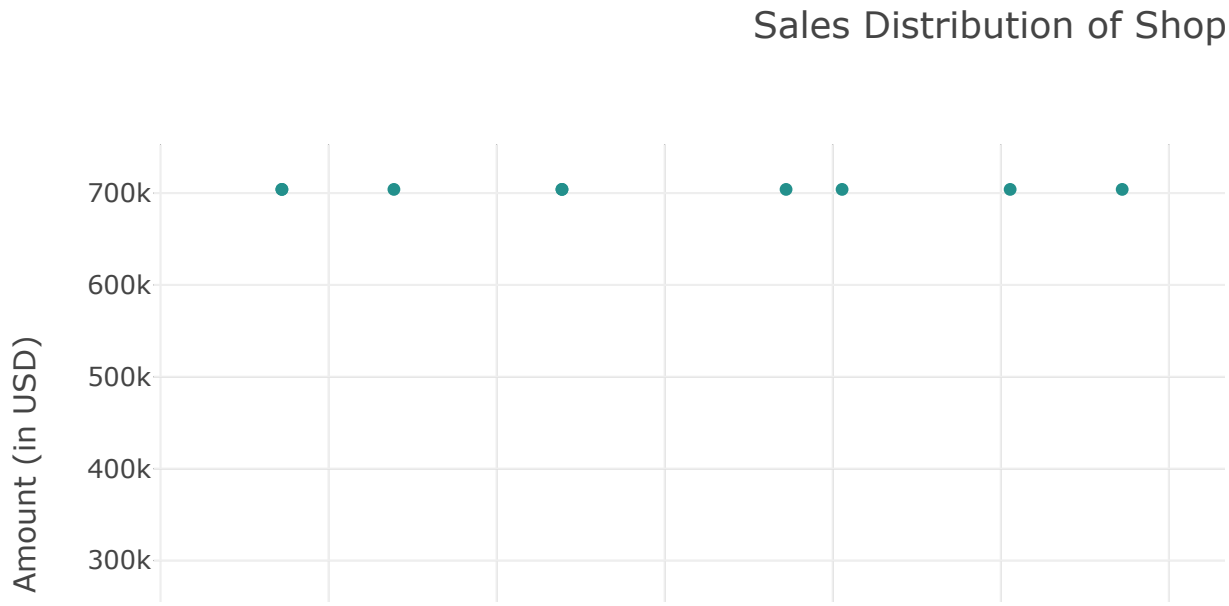
```
Collecting plotly
  Downloading plotly-5.8.0-py2.py3-none-any.whl (15.2 MB)
     |████████████████████████████████| 15.2 MB 4.7 MB/s eta 0:00:01
Collecting tenacity>=6.2.0
  Downloading tenacity-8.0.1-py3-none-any.whl (24 kB)
Installing collected packages: tenacity, plotly
Successfully installed plotly-5.8.0 tenacity-8.0.1
Note: you may need to restart the kernel to use updated packages.
```

In [24]:
```
import plotly.express as px
```

In [93]:
```
plot_fig = px.scatter(data, x="created_at",y="order_amount",
                  color="shop_id",
                  size_max=50,
                  log_x=False,
                  template="gridon",
                   labels={
                      "created_at": "Time",
                      "order_amount": "Order Amount (in USD)",
                  },
                  title="Sales Distribution" )
plot_fig.show()
```



In [94]:
```
# plot for shop 42
shop_42 = data[data['shop_id'] == 42]
plot_fig = px.scatter(shop_42, x="created_at",y="order_amount",
                      color="shop_id",
                      log_x=False,
                      size_max=60,
                      hover_data=["total_items"],
                      template="gridon",
                      labels={
                          "created_at": "Time",
                          "order_amount": "Order Amount (in USD)",
                      },
                      title="Sales Distribution of Shop 42" )
plot_fig.show()
```



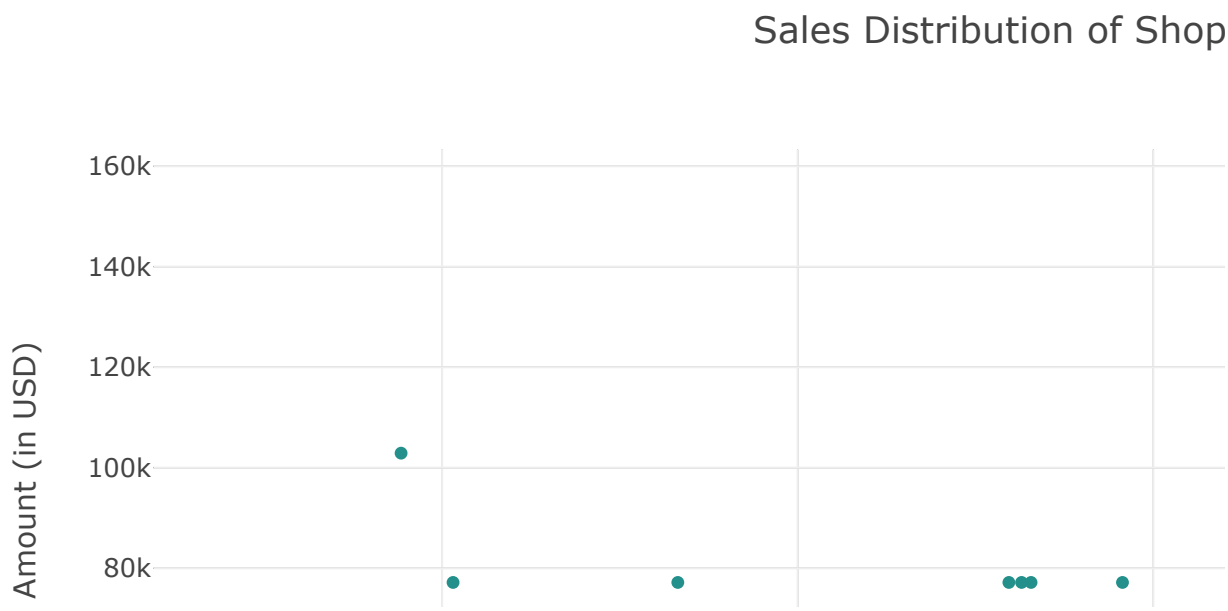
There is a constant purchase of 2000 items that every time amounts to 704k USD (almost on a regular basis). We therefore, need to have a close look of this to examine if these are outliers. Since there is a possibility that there can be a bulk order placement that results in such high `order_amount` we can say that this causes skewness in distribution of data yet not an outlier as such.

In [67]:
```
roup_by_Total = data.groupby('total_items')['order_amount'].sum().rena
otal_items_count = data.total_items.value_counts().rename_axis('total_
roup_by_Total
```

Out[67]:

	total_items	total
0	1	763777
1	2	1374394
2	3	1120803
3	4	277672
4	5	58470
5	6	161460
6	8	1064
7	2000	11968000

In [95]:
```
# plot for shop 78
shop_78 = data[data['shop_id'] == 78]
plot_fig = px.scatter(shop_78, x="created_at",
                  y="order_amount",
                  color="shop_id",log_x=False,
                       hover_data=["total_items"],
                  size_max=60, template="gridon",
                  labels={
                      "created_at": "Time",
                      "order_amount": "Order Amount (in USD)",
                  },
                  title="Sales Distribution of Shop 78" )
plot_fig.show()
```



This shows that data is distributed over a wide range. On randomly hovering over the data points we find that each pair of sneakers costs over 25k USD (maybe this shop sells expensive/high-end) products when compared to other shops. Therefore,this also causes skewness in data and is also not a potential outlier.

b.What metric would you report for this dataset?

Using the `order_amount` as a metric may not be appropriate here as the data evidently shows skewness. From visualization, we found that shop 42 sells bulk orders of 2000 items amounting to 704000 and shop 78 which sells high end products that typically seemed to start from 25k USD (which resulted in skewness).
I personally feel that using a single metric might lead to incorrect insights. But using them in a different way can help us get better insights.
We need to separate orders on the basis of the total items and see how much the quantity of items contribute to sales cost by grouping them.
Also, in cases where a single product is sold for a comparatively expensive amount we can strt by looking at the standard deviation and median sales of that to form better insights.

c. What is its value?

Let us now calculate the required metrics,

In [78]:
```
groupby_total_items = data.groupby('total_items')['order_amount'].mean
groupby_total_items
```

Out[78]:

	Total Items in Order	Average of Order Value
0	1	417.364481
1	2	750.215066
2	3	1191.076514
3	4	947.686007
4	5	759.350649
5	6	17940.000000
6	8	1064.000000
7	2000	704000.000000

In [80]:
```
upby('total_items')['order_amount'].std().rename_axis('Total Items in
upby('total_items')['order_amount'].median().rename_axis('Total Items
```

In [81]:
```
groupby_total_items['Median of Order Value'] = median
groupby_total_items['Standard Deviation'] = std_dev
```

In [82]:
```
groupby_total_items
```

Out[82]:

	Total Items in Order	Average of Order Value	Median of Order Value	Standard Deviation
0	1	417.364481	153.0	2593.090627
1	2	750.215066	306.0	4760.572162
2	3	1191.076514	459.0	7471.160149
3	4	947.686007	592.0	5977.632918
4	5	759.350649	765.0	161.174453
5	6	17940.000000	948.0	51153.864136
6	8	1064.000000	1064.0	NaN
7	2000	704000.000000	704000.0	0.000000

There is a single order in the dataset in which 8 items were ordered, that explains the NaN Standard deviation. The orders where 2000 items were ordered amounts to 704k USD that explains for the 0 standard deviation.

Therefore, grouping data on the basis of quantity of items purchased gives us more meaningful insights. Also, the median of order values gives us a good idea about how data is distributed over the 50th percentile. We can find what part of order values range below and above 50%.
Rather than looking at a single metric for analysis, comibining the Average Order Value (AOV), Median Order Value(MOV), and the Standard Deviation gives us a good insight about the distribution of order amounts. The variation or difference between the Median Order Value and Average of Order Value gives an approximate estimate of the skew in the data distribution. Additionally, the standard deviation gives information about how much our order values vary.