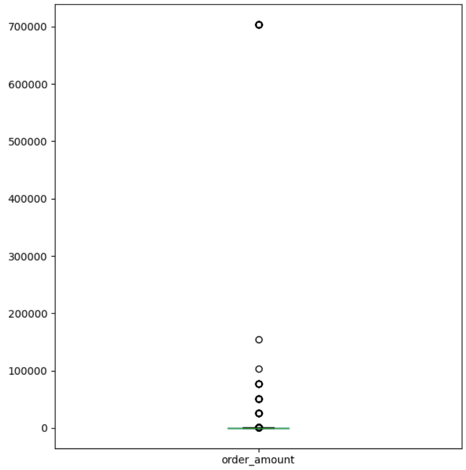
Daniel Matlin Shopify Winter 2021 Data Science Intern Challenge

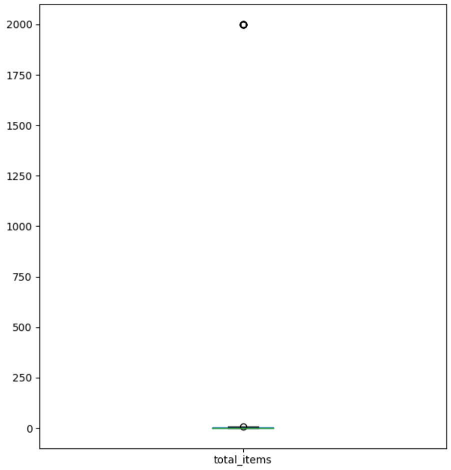
Question 1:

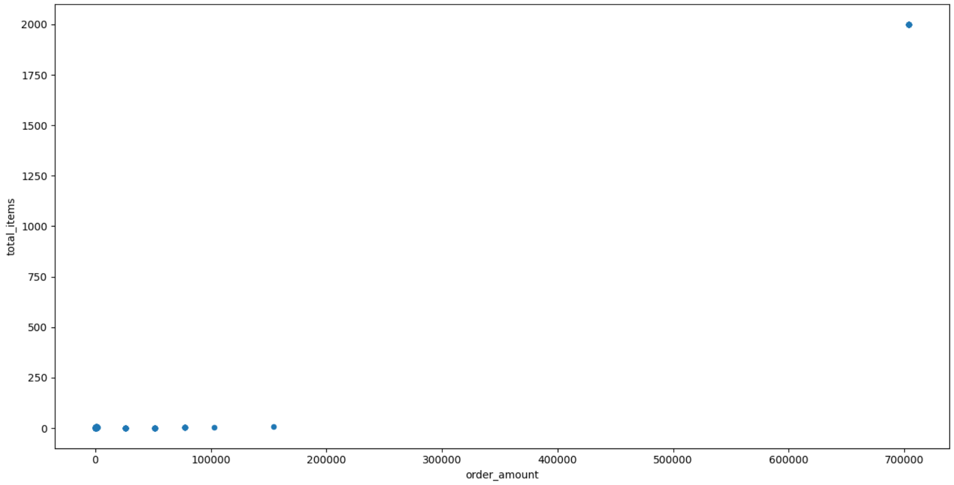
Note: I used Python with `pandas`*,* `matplotlib`*, and* `scipy` *in [this program](#) to answer this question.*

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

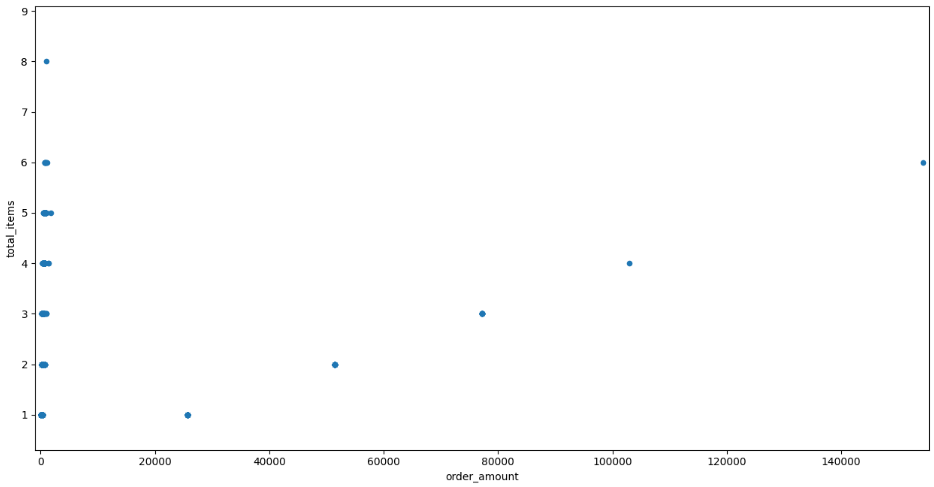
My first step was to look at various characteristics of the dataset's two quantitative columns (`order_amount` and `total_items`, which I determined correspond to order price and items ordered, respectively). I immediately noticed that for each column the median was far below the mean and a high max, indicating rightward skew. This was confirmed by the columns' skew values, both around +17, where a value greater than 1 is generally considered indicative of high rightward skew.
To be able to better understand the data, I plotted boxplots for the two quantitative columns, and a scatter plot of one column against the other (there should be a relationship between order price and items ordered), the plots are shown below.





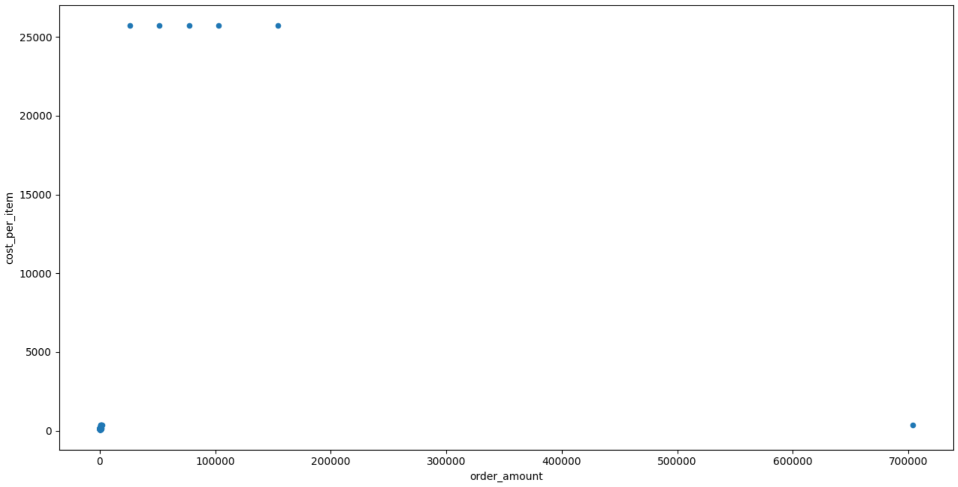


Immediately it is clear from the plots that there are extreme outliers in both columns skewing the results, which are most likely erroneous values. In particular there is one dot (or group of identically placed dots) with an enormous price (~=$700,000) and exceedingly high quantity of items ordered (~=2000). Assuming the most extreme outlier was removed, I zoomed in on the bottom-left of the scatterplot, noticing some evenly placed extreme points (enlarged plot below).



The diagonal set of points are clearly outliers on the order price axis. The even spacing of the dots brought me to realize that if each store sells just one product at one price, this series of extreme dots could represent the orders of one store which mispriced its product; in particular, the store could have accidentally reported figures 100 times the actuals, perhaps by adding a “00” for cents in a reporting tool that only considered whole-dollar prices. While this hypothesis seems very possible, it would be unreasonable to change those values without communication with the store in question. A different approach was necessary to get a better metric for the data.

Next, I decided to look at cost per item for each datapoint on the scatterplot, given by a simple division of the total cost column values by the quantity ordered column values:



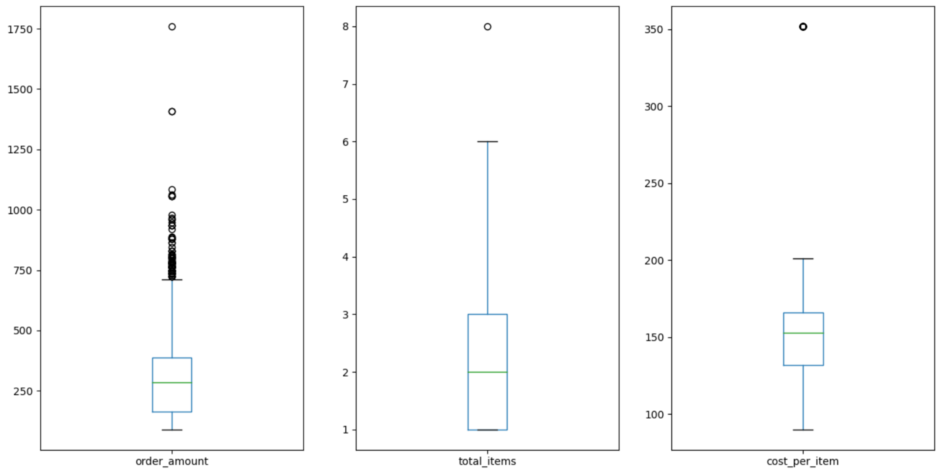
This plot supports the hypothesis that the diagonal points from the previous plot are from one store, as they all represent orders with the same price per item. Interestingly, the most extreme point from the first plot has a more realistic price per item, but is still an outlier given its enormous price.

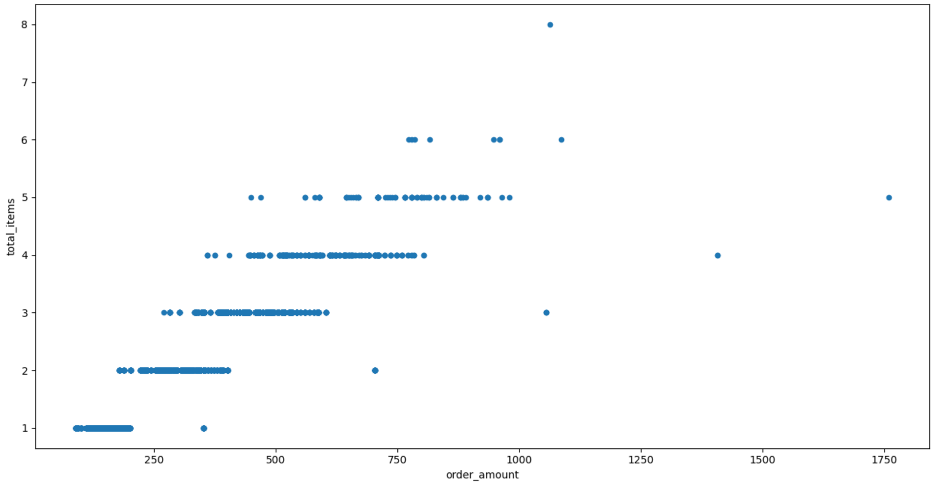
At this point I wrote a check to make sure there were exactly as many store/price-per-product pairs as stores, as the problem statement stated each store sells one type of shoe. This check passed with no problem.

My last step was to determine how to report a metric for this data, which is covered in the next subproblem.

b. What metric would you report for this dataset?

After deciding that data could not be correctively transformed without communication with the data sources, the next step was determining which outlying data to discard. I settled on keeping any data within 3.5 standard deviations of the means in both columns. Despite the high values of the means, a range of 3.5 standard deviations encompasses very nearly all the data but successfully removed the outliers. A set of boxplots and a scatterplot of the two original quantitative columns confirmed this:





It is evident that the new “outliers” of the dataset have much more realistic values. This resulted from the removal of only 63 datapoints of 5000. Given this satisfactory result, the final metric is the average of all values in the `order_amount` column after this outlier removal process.

c. What is its value?
The value of the trimmed average is $302.58, a much more realistic figure than the original.

Question 2:

a. How many orders were shipped by Speedy Express in total?

```
SELECT
      COUNT(*)
FROM
      Orders
NATURAL JOIN
      Shippers
WHERE
      ShipperName = 'Speedy Express';

=> 54
```

Before writing this query, I looked at the `Orders` and `Shippers` tables to check that they held all info necessary for the calculation. With the `NATURAL JOIN` operation, the two tables were joined by equality on the columns included in both tables: in this case, just the `ShipperID` column. The result of the `NATURAL JOIN` was a table with one row for each order, each row including info about the shipper of the represented order. All that was left was to set a condition to restrict the table to orders shipped by Speedy Express, and to count the rows in the resultant table.

b. What is the last name of the employee with the most orders?

```
SELECT
      LastName
FROM
      Orders
NATURAL JOIN
      Employees
GROUP BY
      EmployeeID
ORDER BY COUNT(*) DESC
LIMIT 1;

=> 'Peacock'
```

Similarly to the previous subproblem, I determined that the `Orders` and `Employees` tables had all the info I needed. I again used a `NATURAL JOIN` to add info about orders' employees to each row. Then I used the `GROUP BY` clause to aggregate the data by `EmployeeID` (a necessarily unique value, unlike `LastName`), ordered the aggregated data by count (descending) to sort employees from most to least orders, selected the employees' last name, and limited the result to the top row.

c. What product was ordered the most by customers in Germany?

```
SELECT
      ProductName
FROM
      Orders
NATURAL JOIN
      OrderDetails
NATURAL JOIN
      Customers
NATURAL JOIN
      Products
WHERE
      Country = 'Germany'
GROUP BY ProductID
ORDER BY SUM(Quantity) DESC
LIMIT 1;

=> 'Boston Crab Meat'
```

For this subproblem, I determined that I needed information from the `Orders`, `OrderDetails`, `Customers`, and `Products` tables. The `OrderDetails` table was necessary to join info about products with info about orders, as the `Orders` table did not have that information. As in the previous two subproblems, I used `NATURAL JOIN`s to combine all the tables into one, which contained one row for each product in an order (as well as info related to the customers for the given order). Next, I used a `WHERE` clause to restrict the table to rows corresponding to customers from Germany, then aggregated the table by the products' IDs and ordered them by the sum of their order quantities (descending) to sort products from most to least ordered in a way similar to the previous subproblem. Last, I had the query select the products' names and limited the result to the top row.

When writing this query, I made the assumption that there would be only one product ordered most based on the phrasing of the subproblem; this assumption holds true on the provided dataset. If the assumption could not be made, and a table of equally most-ordered products was to be returned, alternative queries could be constructed. In particular, a subquery that returned the maximum quantity ordered could then be used to restrict the table to records with only that quantity ordered.