Question 1.

a. The formula for AOV is the total revenue /number of orders. Two columns could be used to calculate this metric: order_amount (for revenue) and order_id (to get the number of orders).

If we simply take the sum of the order_amount column and divide it by the number of orders in R, we get:

```
> sum((shops$order_amount)/ length(shops$order_id))
```

$3,145.13, the number from the problem.

However, this metric is very skewed because there are plenty of orders where sneakers were bought in bulk. Additionally, there is at least one example of sneakers that had prices that were far above the others. The shop with the shop_id of 78 sold several single pairs of sneakers for **$25,725.**

If our goal is to get a better sense of what a single order (or single pair of shoes) costs, then we could use the sum of the total items sold instead of the number of orders. This takes care of the bulk orders we see in the data. Additionally, outliers like shop_id 78 could also be removed. But before removing any outliers, context is necessary.

b. Here is what a snippet of the table looks like.

```
> head(shops)
```

	order_id	shop_id	user_id	order_amount	total_items	payment_method	created_at
1:	1	53	746	224	2	cash	2017-03-13 12:36:56
2:	2	92	925	90	1	cash	2017-03-03 17:38:52
3:	3	44	861	144	1	cash	2017-03-14 04:23:56
4:	4	18	935	156	1	credit_card	2017-03-26 12:43:37
5:	5	18	883	156	1	credit_card	2017-03-01 04:35:11
6:	6	58	882	138	1	credit_card	2017-03-14 15:25:01

To get a sense of what a “single” order costs, we can use the total of the 2 highlighted columns. We can call this new metric “**sneaker/model cost**” (basically AOV but instead of orders we are looking at items).

c. There are a couple of different values or metrics that can be calculated using **sneaker cost** depending on what we want to achieve.

In R, we can do:
> sum((shops$order_amount)/ sum(shops$total_items))

The **overall average** (across all shops) **sneaker costs** around **$357.92.** This is a much more reasonable estimate of a pair of sneakers. It is still way too high for my taste, but it is far more reasonable than our original AOV.

Repeating the code above without the shop_id 78 gives us an **overall average sneaker cost** of **$307.01**.

However, according to this Shopify blog article, reporting a single measure of central tendency may not fully capture what is going on. Taking the **sneaker cost** for each store and then calculating the measures of central tendency may be more beneficial than just calculating an overall average cost for all stores.

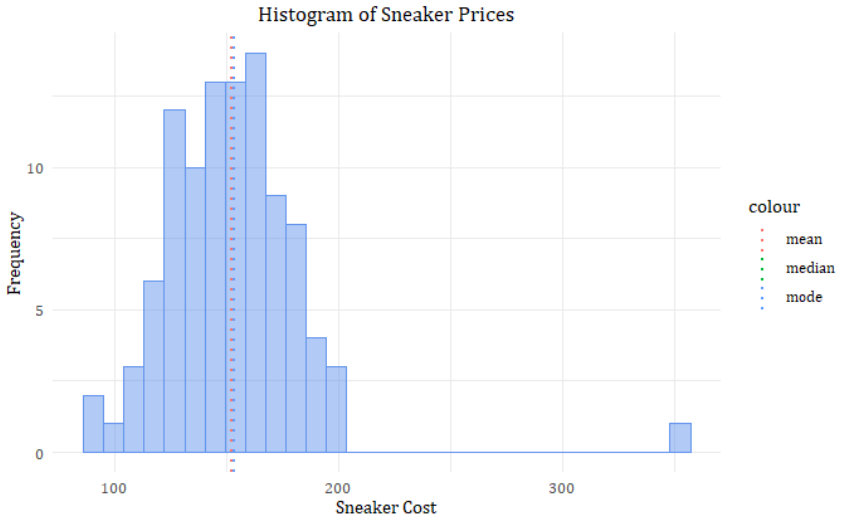
In R, we can take the **sneaker cost** for each shop and then calculate the mean, median, and mode.

```
> key_metrics <- shops %>%
  group_by(shop_id) %>%
  summarize(sneaker_cost = sum(order_amount)/ sum(total_items)) %>%
  ungroup() %>%
  summarize(mean_sneaker_cost = mean(sneaker_cost),
            median_sneaker_cost = median(sneaker_cost),
            modal_sneaker_cost= calc_mode(sneaker_cost))
```

Mean sneaker cost: $407.99
Median sneaker cost: $153
Modal sneaker cost: $153

This massive difference between the mean and the other metrics is caused by shop_id 78. This shows how sensitive means are to outliers. If we repeat the code above without shop_id 78, we get:

Mean sneaker cost: $152.26
Median sneaker cost: $153
Modal sneaker cost: $153



If I had to report a single value, I would choose $153.

Question 2

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

Speedy Express shipped 54 orders.

```
SELECT COUNT(OrderID)
FROM Orders
JOIN Shippers
ON Orders.ShipperID = Shippers.ShipperID
WHERE ShipperName = "Speedy Express";
```

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

With 40 orders, it looks like Peacock has the most orders.

```
SELECT LastName, COUNT(OrderID) FROM Orders
JOIN Employees
ON Orders.EmployeeID = Employees.EmployeeID
GROUP BY LastName
ORDER BY COUNT(OrderID) DESC;
```

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

Gorgonzola Telino was the most ordered product by customers in Germany.

```
SELECT ProductName, COUNT(*), Country FROM (
SELECT Products.ProductID, Products.ProductName, Orders.OrderID, Orders.CustomerID,
Customers.Country FROM Products
JOIN OrderDetails
ON Products.ProductID = OrderDetails.ProductID
JOIN Orders
ON OrderDetails.OrderID = Orders.OrderID
JOIN Customers
ON Orders.CustomerID = Customers.CustomerID
WHERE Country=  "Germany")
GROUP BY ProductName
ORDER BY COUNT(*) DESC;
```