Summer 2022 Data Science Intern Challenge

Amir Moghaddas Jafari

1/11/2022

Question 1

Regarding the definition of AOV based on *Shopify* blog, AOV is **the total monthly revenue divided by the number of orders in that month**. To start exploring data, I use SQL queries to extract some information. First, I start with calculating the whole average order amount in the table. "shoe_shops" is the name of the given data.

```
SELECT Avg(order_amount) AS avg_amount
FROM   shoe_shops;
```

Table 1: 1 records

avg_amount
3145.128

We can see that, this value is equal to the reported metric by Question.

Part a

With knowing the following principal, we start to explore the data:

- **Average of averages is NOT equal to the average of the whole when the groups have NOT the same size**
- **Garbage in, garbage out! We must attempt to clean the dataset before starting any investigation**

In the following chunks code, I will calculate the AOV for **each shoe shop** and show 10 of them below. I also included the unit price of shoes for the further experiments. Because each store only sells one model of shoes, the unit price for each store indicates the price (model) of shoes that it sells.

```
CREATE TABLE IF NOT EXISTS aov_shops AS
  SELECT shop_id,
         Count(order_id)                                         AS
         total_order,
         Sum(total_items)                                        AS
         sum_orders,
         Sum(order_amount)                                       AS
         total_amount,
         Sum(order_amount) / Sum(total_items)                    AS
         unit_price,
         Cast(Sum(order_amount) AS FLOAT) / Cast(Count(order_id)AS FLOAT) AS AOV
  FROM   shoe_shops
  GROUP  BY shop_id;
```

```
SELECT *
FROM   aov_shops
LIMIT  10;
```

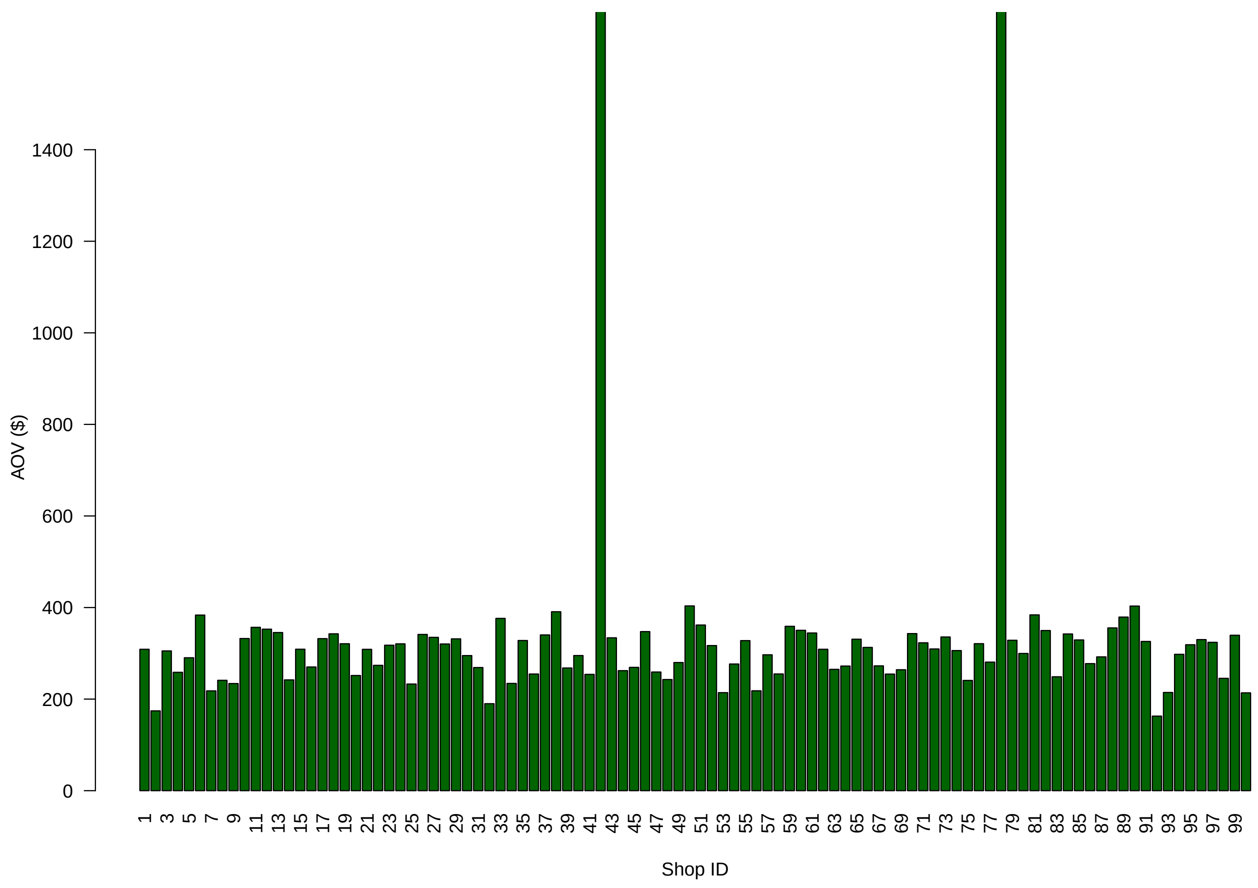
Table 2: Displaying records 1 - 10

shop_id	total_order	sum_orders	total_amount	unit_price	AOV
1	44	86	13588	158	308.8182
2	55	102	9588	94	174.3273
3	48	99	14652	148	305.2500
4	51	103	13184	128	258.5098
5	45	92	13064	142	290.3111
6	59	121	22627	187	383.5085
7	56	109	12208	112	218.0000
8	46	84	11088	132	241.0435
9	59	117	13806	118	234.0000
10	53	119	17612	148	332.3019

Now, I will import "aov_shops" table from the database into R workplace to get some insights from the dataset. The following figure is the bar chart plot of the AOV ($) for each shoe shop.

```r
aov_shops <- dbGetQuery(db, {
  "
  SELECT *
  FROM aov_shops "
})


aov <- c(aov_shops$AOV)
x_name <- c(aov_shops$shop_id)
barplot(
  aov,
  col = "darkgreen",
  xlab = "Shop ID",
  ylab = "AOV ($)",
  names.arg = x_name ,
  ylim = c(0, 1500) ,
  las = 2
)
```



We can see that, AOV of shops numbers 42 and 78 are suspicious due to the unusual high value of AOV comparing other shoe shops and also we know that shoes are not among of too expensive apparel such as jewelries. Lets extract the information about these 2 shops.

```sql
SELECT *
FROM   aov_shops
WHERE shop_id = 42 or shop_id = 78;
```

Table 3: 2 records

shop_id	total_order	sum_orders	total_amount	unit_price	AOV
42	51	34063	11990176	352	235101.49
78	46	88	2263800	25725	49213.04

There are 2 issues I would like to mention here:

- The total sold shoes for shop ID 42 is 34063 shoes. It means about 668 shoes has been sold through each order in average. This is not rational. We should find the problem here.
- The unit price of the shoes sold in shop ID 78 is $25,725 and it is far from reality. Since, we have 56 orders for these shoes and the shop already sold 88 pairs of them in a month (around 3 pairs a day), the shoes can not be this much expensive. We also should find the possible mistake here.

For Shop ID 42, I will start by grouping the orders based on the payment method.

```sql
SELECT Sum(total_items), payment_method
FROM   shoe_shops
```

```sql
WHERE shop_id = 42
GROUP BY payment_method;
```

Table 4: 3 records

Sum(total_items)	payment_method
22	cash
34016	credit_card
25	debit

After looking at the previous table, we can conclude that there is a clear problem for payments done through the credit carts.

```sql
SELECT *
FROM   shoe_shops
WHERE shop_id = 42 and payment_method = "credit_card";
```

Table 5: Displaying records 1 - 10

order_id	shop_id	user_id	order_amount	total_items	payment_method	created_at
16	42	607	704000	2000	credit_card	2017-03-07 4:00:00
41	42	793	352	1	credit_card	2017-03-24 14:15:41
61	42	607	704000	2000	credit_card	2017-03-04 4:00:00
309	42	770	352	1	credit_card	2017-03-11 18:14:39
410	42	904	704	2	credit_card	2017-03-04 14:32:58
521	42	607	704000	2000	credit_card	2017-03-02 4:00:00
939	42	808	1056	3	credit_card	2017-03-13 23:43:45
1105	42	607	704000	2000	credit_card	2017-03-24 4:00:00
1363	42	607	704000	2000	credit_card	2017-03-15 4:00:00
1437	42	607	704000	2000	credit_card	2017-03-11 4:00:00

As we can see (10 of 27 records has been shown), there are 17 orders created by user number 607 with exactly 2000 pairs of the shoes and created at the same time of day(4:00:00) . This is skeptical. This can be a fraud or an inserting data error or a database system flaw or lots of other possibilities.

Now, we want to extract some information from shop ID 78.

```sql
SELECT *
FROM   shoe_shops
WHERE shop_id = 78;
```

Table 6: Displaying records 1 - 10

order_id	shop_id	user_id	order_amount	total_items	payment_method	created_at
161	78	990	25725	1	credit_card	2017-03-12 5:56:57
491	78	936	51450	2	debit	2017-03-26 17:08:19
494	78	983	51450	2	cash	2017-03-16 21:39:35
512	78	967	51450	2	cash	2017-03-09 7:23:14
618	78	760	51450	2	cash	2017-03-18 11:18:42
692	78	878	154350	6	debit	2017-03-27 22:51:43
1057	78	800	25725	1	debit	2017-03-15 10:16:45
1194	78	944	25725	1	debit	2017-03-16 16:38:26

order_id	shop_id	user_id	order_amount	total_items	payment_method	created_at
1205	78	970	25725	1	credit_card	2017-03-17 22:32:21
1260	78	775	77175	3	credit_card	2017-03-27 9:27:20

Regarding the table above that shows 10 records of shop ID 78, the unit price shoes is skeptical. However, number of transactions and other information seem to be fine. I think there is a mistake in inserting the unit price of the shoe.

Summary: Based on our investigation I propose the following suggestions to clean up the dataset from posible mistakes.

- Delete records for user ID 607 that has created 17 orders by credit cart and each order included 2000 pairs of shoes and also all orders created at exactly 4:00:00.
- Change the shoes price of shop ID 78 from 257,25 to 257 dollars.

I use R work space to clean up the data.

```
shoe_shops <- dbGetQuery(db, {
  "
  SELECT *
  FROM shoe_shops "
})

shoe_shops_updated <- subset(shoe_shops, user_id != 607)

shoe_shops_updated$order_amount[shoe_shops_updated$shop_id == 78] <- shoe_shops_updated$order_amount[sh

# importing data from R to the database named "db"
dbWriteTable(db, "shoe_shops_updated", shoe_shops_updated, overwrite = TRUE)
```

After Cleaning up the dataset, we want to calculate AOV and demonstrate the bar chart graph again for the updated AOV.

```
CREATE TABLE IF NOT EXISTS aov_shops_updated AS
  SELECT shop_id,
         Count(order_id)                                                   AS
         total_order,
         Sum(total_items)                                                  AS
         sum_orders,
         Sum(order_amount)                                                 AS
         total_amount,
         Sum(order_amount) / Sum(total_items)                              AS
         unit_price,
         Cast(Sum(order_amount) AS FLOAT) / Cast(Count(order_id)AS FLOAT) AS AOV
  FROM   shoe_shops_updated
  GROUP  BY shop_id;
```

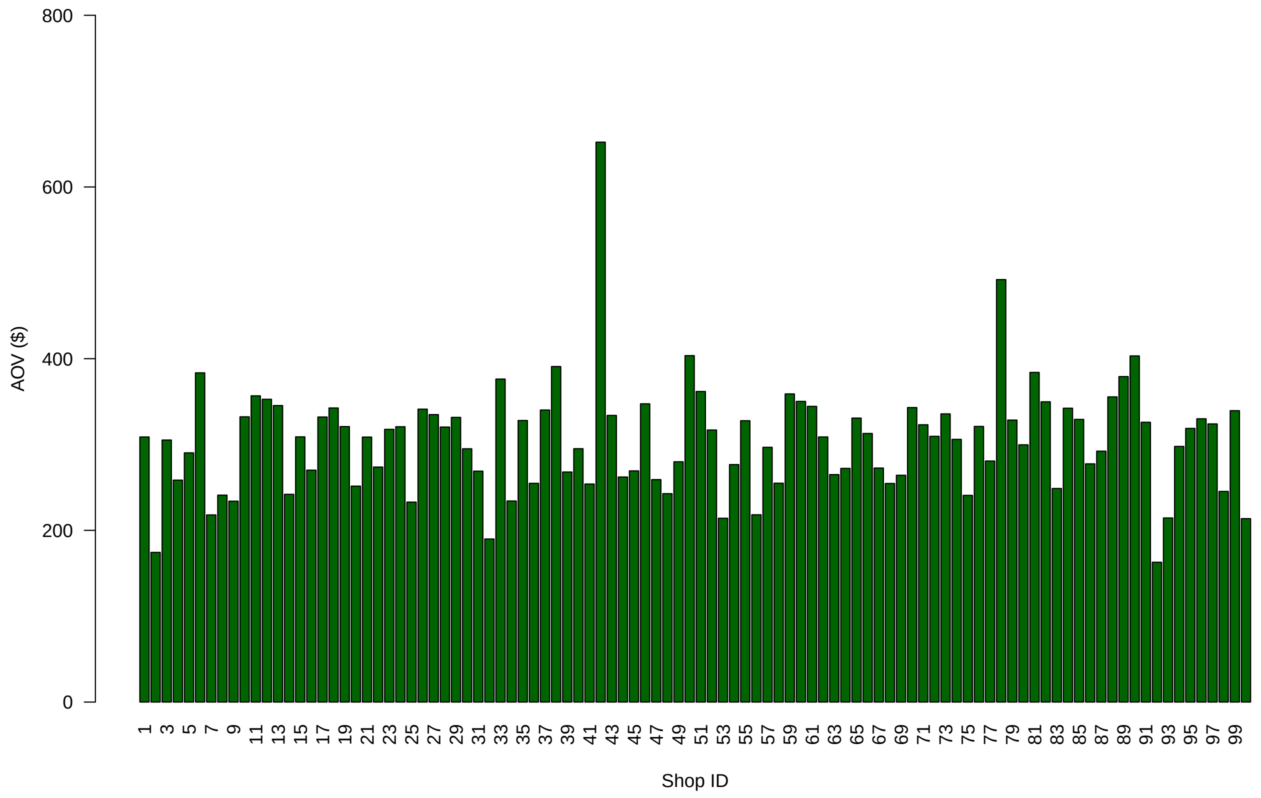
```
SELECT *
FROM   aov_shops_updated
LIMIT  10;
```

Table 7: Displaying records 1 - 10

shop_id	total_order	sum_orders	total_amount	unit_price	AOV
1	44	86	13588	158	308.8182
2	55	102	9588	94	174.3273
3	48	99	14652	148	305.2500
4	51	103	13184	128	258.5098
5	45	92	13064	142	290.3111
6	59	121	22627	187	383.5085
7	56	109	12208	112	218.0000
8	46	84	11088	132	241.0435
9	59	117	13806	118	234.0000
10	53	119	17612	148	332.3019

```
aov_shops_updated <- dbGetQuery(db, {
  "
  SELECT *
  FROM aov_shops_updated "
})


aov_updated <- c(aov_shops_updated$AOV)
x_name <- c(aov_shops_updated$shop_id)
barplot(
  aov_updated,
  col = "darkgreen",
  xlab = "Shop ID",
  ylab = "AOV ($)",
  names.arg = x_name ,
  ylim = c(0, 800) ,
  las = 2
)
```



Now we can see that we have a more uniform AOV bar chart.

Part b and c

We would like to introduce 2 metrics for the data set.

Metric 1 and its value There are two ways to calculate the average of AOV's:

1- Taking average of all order amounts (sum of all order amounts divided by number of orders/number of rows in the dataset)

2- Taking average of AOV's (sum of AOV's divided by number of shoe stores which is 100 in this case)

The fact that average of averages is not equal to average of whole when the groups don't have the same size would cause a small difference in the metric.

Following is the average of AOV's:

```sql
SELECT Avg(AOV)
FROM   aov_shops_updated;
```

Table 8: 1 records

Avg(AOV)
305.1324

And also following is the average of whole order amounts:

```sql
SELECT Avg(order_amount)
FROM   shoe_shops_updated;
```

Table 9: 1 records

Avg(order_amount)
304.3303

The small difference is due to the issue that I reported.

In my point of view, Since all shops sell one distinct item, **I would like to use this fact to expand the metric definition based on the shoes model rather than stores.** That being said, I prefer to work with the AOV for **each shop** that can be interpreted as AOV for **each model of shoe**. So, taking average of all AOV's would also give us a metric about the average shoe order value.

Shopify can use this metric to provide an excellent insight for business owners about the possible AOV (Average AOV) for the individuals who want to run a shoe store with Shopify and sell a particular model of shoe. Therefore, the *expected* money amount from each order created in a shoe shop that sells a particular model is 305.13 dollars.

$$metric\ 1 : 305.13\ dollars$$

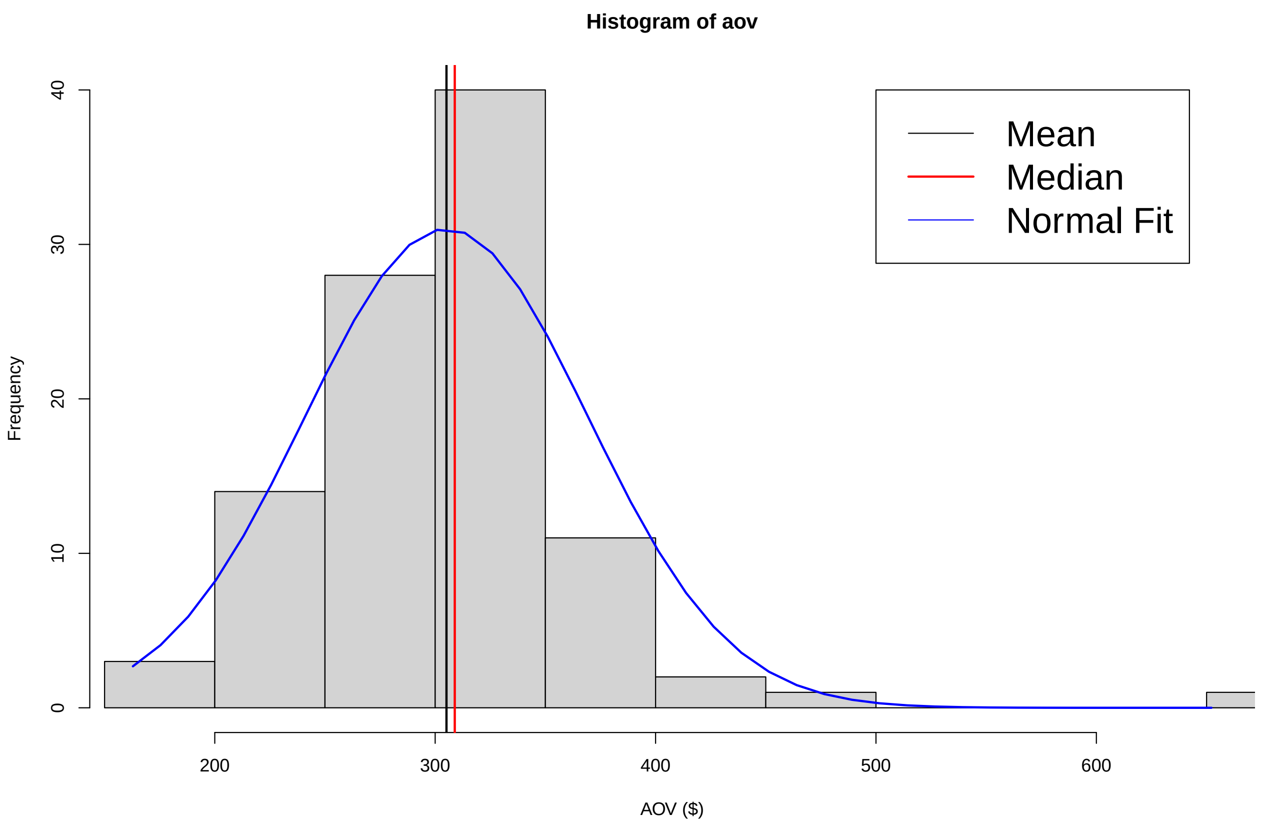
Metric 2 and its value (more advanced) This metric focuses on giving advice to shoe shop owners about the shoe models that possibly boost their sail. Interchangeably, we can interpret this metric as a range of shoe prices that boosts the sail because stores are selling one distinct model of shoes.

In statistics we can not just rely on the average because sometimes they do not give us a true picture of what is happening. I will graph the histogram of the AOV's.

```r
aov <- aov_shops_updated$AOV
xmin <- min(aov)
xmax <- max(aov)
h <-hist(aov, xlim = c(xmin, xmax), xlab = "AOV ($)", freq = TRUE)
abline(v = mean(aov), col = "black", lwd = 2)
abline(v = median(aov), col = "red", lwd = 2)

xfit <- seq(min(aov), max(aov), length = 40)
yfit <- dnorm(xfit, mean = mean(aov), sd = sd(aov))
# Scaling the Normal Fit
yfit <- yfit * diff(h$mids[1:2]) * length(aov)

lines(xfit, yfit, col = "blue", lwd = 2)
legend(500, 40, legend=c("Mean", "Median", "Normal Fit"),
       col=c("black", "red", "blue"), lwd=1:2, cex=2)
```



Since the normal curve (the blue curve) is a bit skewed and we have outliers in our dataset, it is better to report **median** instead of **mean**. The median is the **red** line on the histogram graph. I also showed **mean** with black line on the graph.

$$Median = 308.89\ dollars$$

We would like to find a range price of shoes that boosts the sail. We rank the Frequencies and pick the first two ranks. We can observe that between AOV of 250 and 350 dollars, we have the most occurred AOV. Now, lets find which shoes have the AOV between this range.

```
shoe_prices <- dbGetQuery(db, {
  "
  SELECT unit_price,
         aov
  FROM   aov_shops_updated
  WHERE  aov BETWEEN 250 AND 350"
})

price_range <- range(shoe_prices$unit_price)
price_range
```

```
## [1] 118 195
```

Summary: **The fact that shops are selling one model of shoes enable us to conclude that shoe prices (shoe models) between 118 and 195 dollars would boost the AOV** because the stores that selling shoes in this price range have had the most occurred AOV.

Suggestion: If Shopify had a conversationa-AI agent for their clients, it is a good idea to report this metric to shoe shop owners to boost their AOV since AOV is one of the most important metric for e-commerce.

Metric 2 : Selling Shoe models between 118 and 195 dollars would boost the AOV

Question 2

a

```
SELECT Count(*)
FROM   orders
       JOIN shippers
         ON shippers.shipperid = orders.shipperid
WHERE  shippers.shippername = 'Speedy Express';
```

Result : 54

b

```
SELECT employees.lastname,
       Count(orderid) AS num_orders
FROM   employees
       LEFT JOIN orders
              ON orders.employeeid = employees.employeeid
GROUP  BY employees.lastname,
          employees.firstname
ORDER  BY num_orders DESC;
```

We can see that Peacock has the most orders with 40 orders.

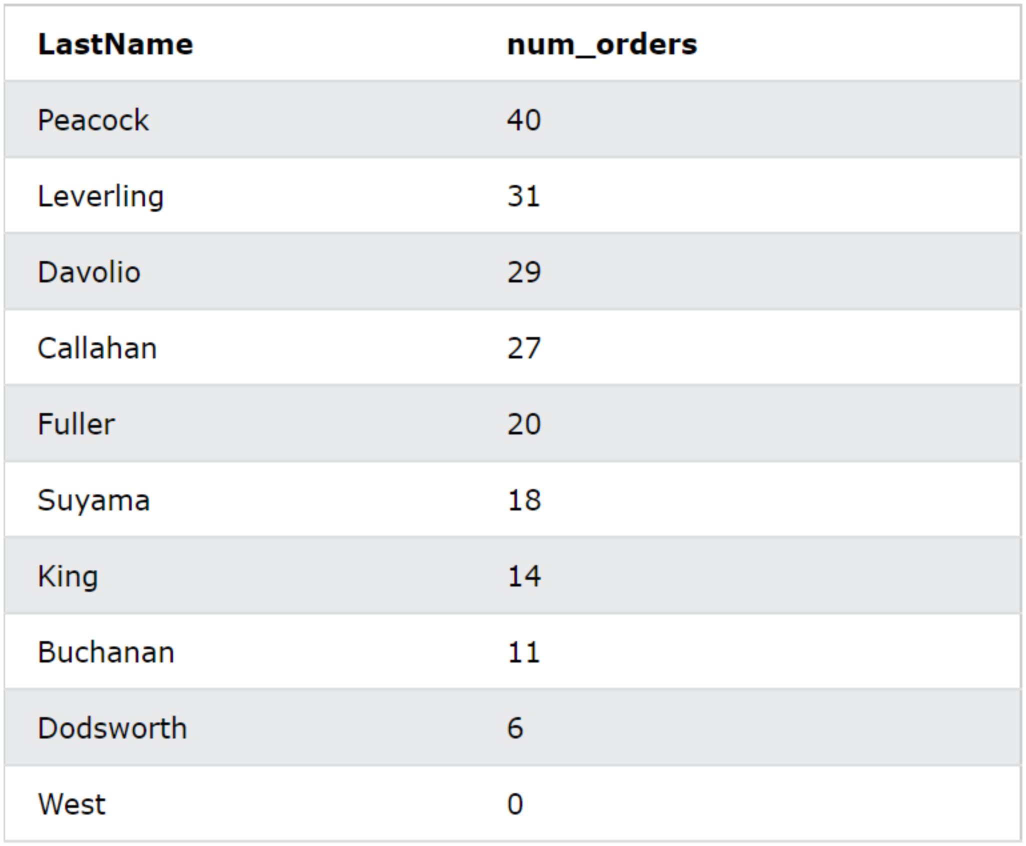
Note that: Employee name with last name of West had no order! I used left join to capture this.

c

```
SELECT products.productname,
       Sum(orderdetails.quantity) AS most_ordered
FROM   orders
       JOIN orderdetails
         ON orders.orderid = orderdetails.orderid
       JOIN customers
         ON orders.customerid = customers.customerid
       JOIN products
         ON orderdetails.productid = products.productid
WHERE  country = 'Germany'
GROUP  BY products.productname
ORDER  BY most_ordered DESC
LIMIT  1;
```

Result : Boston Crab Meat with 160 *total orders*

The End.



LastName	num_orders
Peacock	40
Leverling	31
Davolio	29
Callahan	27
Fuller	20
Suyama	18
King	14
Buchanan	11
Dodsworth	6
West	0

Figure 1: Image source: W3schools