Data Science – Data Preparation and Visualization [DPV]

Yoran de Weert (s1693859)
Selina Zwerver (s1690388)

December 9, 2020

Assignment 1: Facts and dimensions

Exercise 1.1

a) A fact has two components: a numerical property and a combination formula. In this case, the fact is the amount of participants, and the dimensions are the day of the week, year, location and number of events.

b) The conceptual star schema is shown in Figure 1.

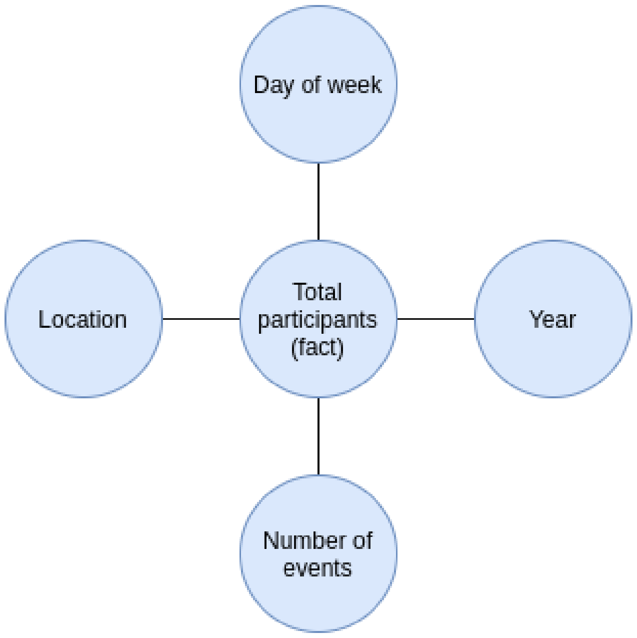

Figure 1: Star schema for the events happening in the city of Utrecht.

Exercise 1.2

a) The fact is the number of registrations, the dimensions are the institution, study, phase, year and sex.

b) The conceptual star schema is shown in Figure 2.

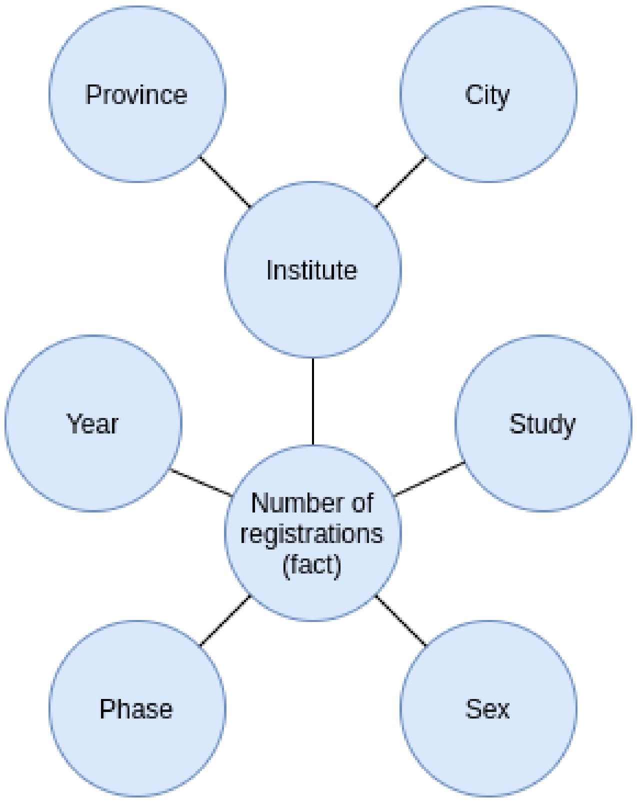

Figure 2: Star schema for the number of registered students in The Netherlands.

Assignment 2

R code

```r
library(readr)
library(dplyr)
library(DBI)
library("RPostgreSQL")

setwd("/home/selina/Desktop/DS/datascience/DPV/")
data0 <- read_delim(file = "BI_Raw_Data.csv",
                    delim=";",
                    col_names=TRUE,
                    col_types=NULL,
                    locale=locale(encoding="ISO-8859-1"))
# head(data0) # inspect first five rows

# Make table 'product'
product <- data0 %>%
  select(Product_Name, Product_Category) %>%
  rename(name = Product_Name, category = Product_Category) %>%
  arrange(name, category) %>%
  group_by(name, category) %>% # grouping the rows based on name and category
  distinct() %>% # retain only unique rows
  ungroup() %>%
  mutate(productid = row_number()) # add identifier

# head(product)

# Make table 'customer'
customer <- data0 %>%
  select(Customer_Name, Customer_Country) %>%
  rename(name = Customer_Name, country=Customer_Country) %>%
  arrange(name, country) %>%
  group_by(name, country) %>%
  distinct() %>%
  ungroup() %>%
  mutate(customerid = row_number())

# head(customer)

# Make table 'sales'
sales <- data0 %>%
  select(Order_Date_Day, Product_Name, Product_Category,
         Customer_Name, Customer_Country, Order_Price_Total)
sales <- sales %>%
  full_join(customer, by=c("Customer_Name"="name",
                          "Customer_Country"="country")) %>%
  select(-Customer_Name, -Customer_Country)
sales <- sales %>%
  full_join(product, by=c("Product_Name"="name",
                         "Product_Category"="category")) %>%
  select(-Product_Name, -Product_Category)

# Remove extra columns
sales <- sales[, !(colnames(sales) %in% c("salesid","category","country"))]

# head(sales)

drv <- dbDriver("PostgreSQL")
con <- dbConnect(drv, port=5432, host="bronto.ewi.utwente.nl",
                 dbname="dab_ds20211b_92", user="dab_ds20211b_92",
                 password="esgL4S/HT7fgFST+",
                 options="-c search_path=ass2")
dbWriteTable(con, "product", value = product, overwrite = T, row.names = F)
dbWriteTable(con, "customer", value = customer, overwrite = T, row.names = F)
dbWriteTable(con, "sales", value = sales, overwrite = T, row.names = F)

# Test to see if filled
# dbListTables(con)
# str(dbReadTable(con,"customer"))
# str(dbReadTable(con,"product"))

# Include the output of the following R-code:
dbGetQuery(con,
             "SELECT table_name FROM information_schema.tables
WHERE table_schema='ass2'") ## to get the tables from schema ass2
str(dbReadTable(con, c("ass2", "customer")))
str(dbReadTable(con, c("ass2", "product")))
str(dbReadTable(con, c("ass2", "sales")))
str(dbReadTable(con,"sales"))
```

Visualization

Appendix A shows the histograms for the values of customers and products, respectively. The most valued customers are the customers that have bought the most and hence, highest sales value. This results in the following top-5 most valued customers:

1. Save-a-lot Markets
2. Ernst Handel
3. QUICK-Stop
4. Rattlesnake Canyon Grocery
5. Hungry Owl All-Night Grocers

The other histogram shows the most sold products. This results in the following top-5 products:

1. Côte de Blaye
2. Raclette Courdavault
3. Camembert Pierrot
4. Tarte au sucre
5. Gnocchi di nonna Alice

Output of the piece of code

```
> str(dbReadTable(con, c("ass2", "customer")))
'data.frame':   89 obs. of  3 variables:
 $ name      : chr  "Alfreds Futterkiste" "Ana Trujillo Emparedados y helados"
                    "Antonio Moreno Taquer\xeda" "Around the Horn" ...
 $ country   : chr  "Germany" "Mexico" "Mexico" "UK" ...
 $ customerid: int  1 2 3 4 5 6 7 8 9 10 ...

> str(dbReadTable(con, c("ass2", "product")))
'data.frame':   77 obs. of  3 variables:
 $ name     : chr  "Alice Mutton" "Aniseed Syrup" "Boston Crab Meat"
                    "Camembert Pierrot" ...
 $ category : chr  "Meat/Poultry" "Condiments" "Seafood" "Dairy Products" ...
 $ productid: int  1 2 3 4 5 6 7 8 9 10 ...

> str(dbReadTable(con, c("ass2", "sales")))
'data.frame':   2155 obs. of  4 variables:
 $ orderdate : chr  "10-9-2009" "10-9-2009" "10-9-2009" "10-9-2009" ...
 $ sales     : num  244 586 586 586 1057 ...
 $ customerid: int  72 67 67 67 10 10 10 64 64 64 ...
 $ productid : int  48 7 32 65 20 68 70 7 8 54 ...

> str(dbReadTable(con,"sales"))
'data.frame':   2155 obs. of  4 variables:
 $ orderdate : chr  "10-9-2009" "10-9-2009" "10-9-2009" "10-9-2009" ...
 $ sales     : num  244 586 586 586 1057 ...
 $ customerid: int  72 67 67 67 10 10 10 64 64 64 ...
 $ productid : int  48 7 32 65 20 68 70 7 8 54 ...
```

Assignment 3

Picture of multi-dimensional model

idReturnStatus	returnstatus
0	Returned
1	Not Returned

Table 1: Database for ReturnStatus

productid	name	category	subcategory
1	"While you Were Out" ...	Office Supplies	Paper
2	*Staples* Highlighting Markers	Office Supplies	Pens & Art Supplies
3	*Staples* Letter Opener	Office Supplies	Scissors, Rulers and Trimmers
⋮	⋮	⋮	⋮

Table 2: Database for Product

customerid	name	province	region	segment
1	Aaron Bergman	Alberta	West	Corporate
2	Aaron Bergman	Nunavut	Nunavut	Corporate
3	Aaron Hawkins	British Columbia	West	Home Office
⋮	⋮	⋮	⋮	⋮

Table 3: Database for Customer

orderdate	customerid	productid	idReturnStatus	late	sales	orderquantity	profit	shippingcost
2009-01-01	915	862	1	not late	18036	32	-111.8	4.69
2009-01-01	916	862	1	not late	18036	32	-111.8	4.69
2009-01-01	1228	921	0	not late	87248	9	-342.91	35
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 4: Database for Sales

R code

```r
library(readr)
library(dplyr)
library(DBI)
library("RPostgreSQL")
library(lubridate)

setwd("/home/selina/Desktop/DS/datascience/DPV/")

```

```r
# Load data
data_main = read_delim(file = "SuperSales/SuperstoreSales_main.csv",
                       delim=";", col_names=TRUE, col_types=NULL,
                       locale=locale(encoding="ISO-8859-1"))

data_manager = read_delim(file = "SuperSales/SuperstoreSales_manager.csv",
                       delim=";", col_names=TRUE, col_types=NULL,
                       locale=locale(encoding="ISO-8859-1"))

data_returns = read_delim(file = "SuperSales/SuperstoreSales_returns.csv",
                       delim=";", col_names=TRUE, col_types=NULL,
                       locale=locale(encoding="ISO-8859-1"))

# Make table 'product'
product <- data_main %>%
  select("Product Name", "Product Category","Product Sub-Category" ) %>%
  rename(name="Product Name", category="Product Category",
  subcategory="Product Sub-Category") %>%
  arrange(name, category, subcategory) %>%
  group_by(name, category, subcategory) %>% # grouping the rows
  distinct() %>% # retain only unique rows
  ungroup() %>%
  mutate(productid = row_number()) # add identifier

# Make table 'customer'
customer <- data_main %>%
  select("Customer Name", "Province","Region", "Customer Segment" ) %>%
  rename(name="Customer Name", province="Province", region="Region",
         segment="Customer Segment") %>%
  arrange(name, province, region, segment) %>%
  group_by(name, province, region, segment) %>% # grouping the rows
  distinct() %>% # retain only unique rows
  ungroup() %>%
  mutate(customerid = row_number()) # add identifier

# Make table 'returnstatus'
idReturnStatus <- c(0,1)
returnstatus <- c("Returned", "Not Returned")
ReturnStatus <- data.frame(idReturnStatus, returnstatus,
                    stringsAsFactors=FALSE)

# Make table 'sales'
sales <- data_main %>%
  select("Order Date", Sales, "Order Quantity", "Unit Price", Profit,
         "Shipping Cost", "Customer Name", "Order ID", "Ship Date",
         "Product Name") %>%
  rename(orderdate = "Order Date", sales = Sales, orderquantity =
         "Order Quantity", unitprice = "Unit Price", profit = Profit,
         shippingcost = "Shipping Cost", shipdate = "Ship Date")

```

```r
# Combine data to get product/customer id
sales <- sales %>%
  full_join(customer, by = c("Customer Name" = "name")) %>%
  select( -"Customer Name")
sales <- sales %>%
  full_join(product, by = c("Product Name" = "name")) %>%
  select( -"Product Name")
sales <- sales %>%
  full_join(data_returns, by = c("Order ID" = "Order ID")) %>%
  select( -"Order ID")

# Add returnstatus
sales[c("Status")][is.na(sales[c("Status")])] <- "Not Returned"
sales <- sales %>%
  full_join(ReturnStatus, by = c("Status" = "returnstatus")) %>%
  select( -"Status")

# Determine the data types to see if change is needed
sapply(sales, class)

# Covert profit, shippingcost variables into numeric variables
sales$profit <- gsub(',', '.', sales$profit)
sales$shippingcost <- gsub(',', '.', sales$shippingcost)
sales$profit <- as.numeric(as.character(sales$profit), stringsAsFactors=FALSE)
sales$shippingcost <- as.numeric(as.character(sales$shippingcost),
                        stringsAsFactors=FALSE)

# Convert character dates into numeric dates
sales$orderdate <- dmy(sales$orderdate)
sales$shipdate <- dmy(sales$shipdate)

# Check if the product is shipped late or not
sales$diff_in_days<- difftime(sales$shipdate ,sales$orderdate , units = c("days"))
sales$Late <- ifelse(sales$diff_in_days>=3, "late", "not late")

# Remove unwanted columns from sales table
sales <- sales[, !(colnames(sales) %in% c("shipdate","province", "region",
            "segment", "diff_in_days", "category", "sub_category"))]

# Combine facts in sales table
sales <- sales %>%
  arrange(orderdate, customerid, productid, idReturnStatus, Late) %>%
  group_by(orderdate, customerid, productid, idReturnStatus, Late) %>%
  summarise(sales = sum(sales), orderquantity = sum(orderquantity),
            profit = sum(profit), shippingcost = sum(shippingcost)) %>%
  ungroup()


# Filling the data in PostgreSQL
drv <- dbDriver("PostgreSQL")
```

```
con <- dbConnect(drv, port=5432, host="bronto.ewi.utwente.nl",
                 dbname="dab_ds20211b_92", user="dab_ds20211b_92",
                 password="esgL4S/HT7fgFST+",
                 options="-c search_path=ass3")
dbWriteTable(con, "product", value = product, overwrite = T, row.names = F)
dbWriteTable(con, "customer", value = customer, overwrite = T, row.names = F)
dbWriteTable(con, "sales", value = sales, overwrite = T, row.names = F)
dbWriteTable(con, "ReturnStatus", value = ReturnStatus, overwrite = T, row.names = F)

# Output
dbGetQuery(con,
            "SELECT table_name FROM information_schema.tables
WHERE table_schema='ass3'") ## to get the tables from schema ass3
str(dbReadTable(con, c("ass3", "customer")))
str(dbReadTable(con, c("ass3", "product")))
str(dbReadTable(con, c("ass3", "ReturnStatus")))
str(dbReadTable(con, c("ass3", "sales")))
```

Dashboard visualisations

Appendix B shows the histograms for the lateness, profit and number of returns of each product category. The product categories that were most of the times late are the ones with the highest lateness count (see Figure 6). The 'Paper' category has the highest lateness count, followed by 'Binders and Binder Accessories' and 'Telephones and Communication' . The 'late' variable contains only 'late' and 'not late'. Therefore, it is not possible to determine which products were shipped really late (more than 2 days).

The product categories that made the most loss have a negative profit (see Figure 7). The most negative profit is made by the 'Tables' category, followed by 'Bookcases' and 'Scissors and Rulers'.

The product categories that were returned most have a high number of returned products (see Figure 8). The histogram shows that the 'Paper' category is returned most, followed by 'Binders and Binder Accessories' and 'Telephones and Communication'.

Output of the piece of code

```
> str(dbReadTable(con, c("ass3", "customer")))
'data.frame':   1832 obs. of  5 variables:
 $ name      : chr  "Aaron Bergman" "Aaron Bergman" "Aaron Hawkins"
                    "Aaron Hawkins" ...
 $ province  : chr  "Alberta" "Nunavut" "British Columbia" "Nova Scotia" ...
 $ region    : chr  "West" "Nunavut" "West" "Atlantic" ...
 $ segment   : chr  "Corporate" "Corporate" "Home Office" "Home Office" ...
 $ customerid: int  1 2 3 4 5 6 7 8 9 10 ...

> str(dbReadTable(con, c("ass3", "product")))
'data.frame':   1263 obs. of  4 variables:
 $ name        : chr  "\"While you Were Out\" Message Book, One Form per Page"
                      "*Staples* Highlighting Markers" "*Staples* Letter Opener" "
                      *Staples* Packaging Labels" ...
 $ category    : chr  "Office Supplies" "Office Supplies" "Office Supplies"
                      "Office Supplies" ...
 $ subcategory: chr  "Paper" "Pens & Art Supplies" "Scissors, Rulers and Trimmers"
                      "Labels" ...
 $ productid  : int  1 2 3 4 5 6 7 8 9 10 ...

> str(dbReadTable(con, c("ass3", "ReturnStatus")))
'data.frame':   2 obs. of  2 variables:
 $ idReturnStatus: num  0 1
 $ returnstatus  : chr  "Returned" "Not Returned"

> str(dbReadTable(con, c("ass3", "sales")))
'data.frame':   23353 obs. of  9 variables:
 $ orderdate     : Date, format: "2009-01-01" ...
 $ customerid    : int  915 916 1228 1229 1230 31 31 31 32 32 ...
 $ productid     : int  862 862 921 921 921 395 606 625 395 606 ...
 $ idReturnStatus: num  1 1 0 0 0 1 1 1 1 1 ...
 $ Late          : chr  "not late" "not late" "not late" "not late" ...
 $ sales         : num  18036 18036 87248 87248 87248 ...
 $ orderquantity : num  32 32 9 9 9 32 4 43 32 4 ...
 $ profit        : num  -112 -112 -343 -343 -343 ...
 $ shippingcost  : num  4.69 4.69 35 35 35 7.07 48.8 45 7.07 48.8 ...
```

Assignment 4

a) A good beta tester achieves a high score on completed games or levels in the game. A good gamer achieves a high score, in short time, after completing some levels or the whole game. A different good gamer, however, may also explore all options in the game which takes additional time. The latter option is preferable for a beta tester, however the problem is that 'bad' gamers may as well have a relatively long completion time. The difference is the ending score and hence, the score on the game should have the most impact on the 'goodness score'.

About the completion time, it is assumed that all gamers have the goal to finish the game in the shortest time possible with the highest score possible. This implies that a shorter time to completion leads to a higher 'goodness' score.

Preferably, testers devote a lot of time at once to a game; a gamer is able to really get into the game to gain experience and develop skills. If a gamer constantly switches between apps, the gamer needs some 'warm up' time each time it switches back to the game to get into the game. This implies that some qualitative time is lost each time the gamer starts playing again.

Combining these factors results into one goodness formula for a single game given in Equation 1.

$$g_i(s,c,t) = \alpha_1 \frac{s - s_{\min}}{s_{\max} - s_{\min}} + \alpha_2(1 - \frac{c - c_{\min}}{c_{\max} - c_{\min}}) + \alpha_3 \frac{t - t_{\min}}{t_{\max} - t_{\min}} \tag{1}$$

Where g_i is the goodness score on game i, s the achieved score of the completed game, c the completion time and t the average time of a gaming session playing the game. α_1, α_2 and α_3 are parameters to express the importance of each single feature, which may depend on the game type that is assessed. The variables are normalised to be able to combine the different features.

To obtain the complete goodness score, the experience of the gamer should be considered. In this case, it is simply assumed that the number of games played on Pear smartphones represents the experience of a gamer. The overall goodness G of a gamer is given by:

$$G = \sum_{i \in S} \beta_i g_i \tag{2}$$

Where S is the set of games played by the gamer. β_i is a parameter that indicates the relative gain in experience of playing game i.

b) The data warehouse of Pears has two purposes: 1) To find the best beta testers for a game and 2) to provide the developer with information on how much time the beta tester has devoted to beta testing the game. From these purposes, clear business questions can be derived:

- Which beta testers achieve the highest goodness score on a certain game?
- How much time do these testers spend on completing the game?
- How much time do these testers spend on playing the game?

c) The star schema is displayed in Figure 3 . The fact is the goodness score. The goodness score is based on the gamer and on the game of which the goodness score should be assessed.

The parameters α_1, α_2 and α_3 represent the importance of the considered features in the goodness score formula (Equation 1) and depend on the game that needs testing. The game developer should indicate an importance for the features 'completion time', 'score' and the 'length of a gaming session'. However, these 'importance' parameters will not be included in the star schema since this information cannot be found in the data warehouse.

Based on the parameters given by the game developer, the goodness score of a gamer can be determined. That information can be found in the data warehouse and will be included in the star schema. In the star schema, the completion time indicates the completion time of the whole game. The final score indicates the score when the gamer completed the score. The length of a gaming sessions represents the average time of a gaming session of the concerned gamer. In general, these three dimensions could have been divided into more features. For example, the dimension completion time could have been divided into completion time of certain levels or the completion time of the whole game. However, this is not shown in the star schema since only one feature of each dimension is considered in the calculation of the goodness score.

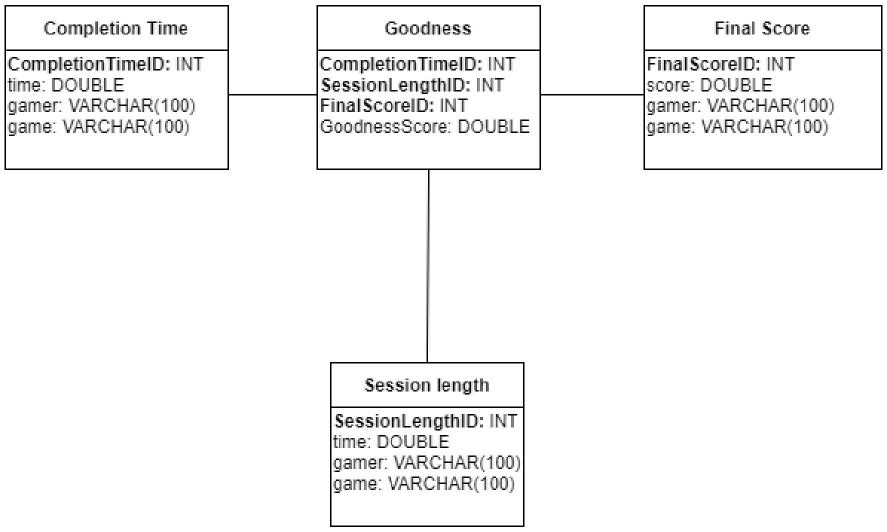

Figure 3: Star schema for the goodness score of a beta tester.

A Visualizations of assignment 2

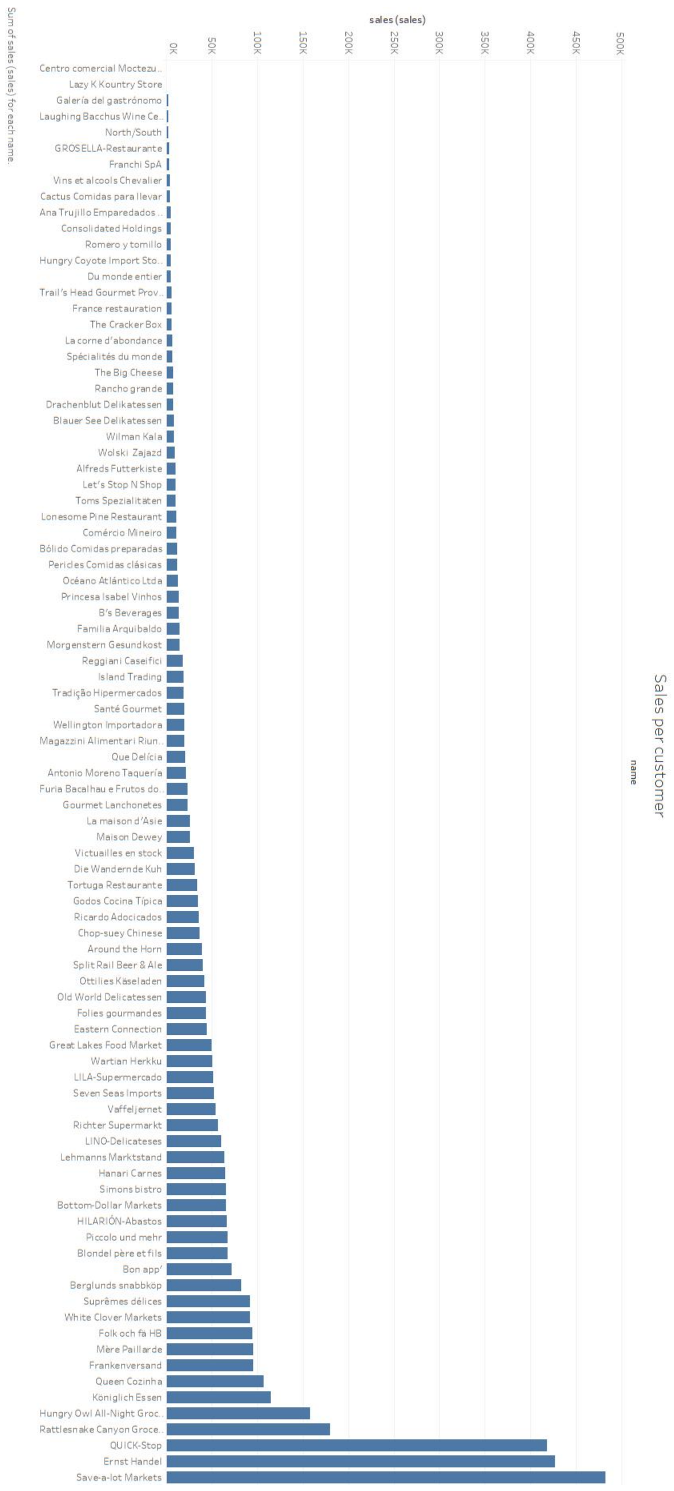

Figure 4: Histogram showing the sales per customer

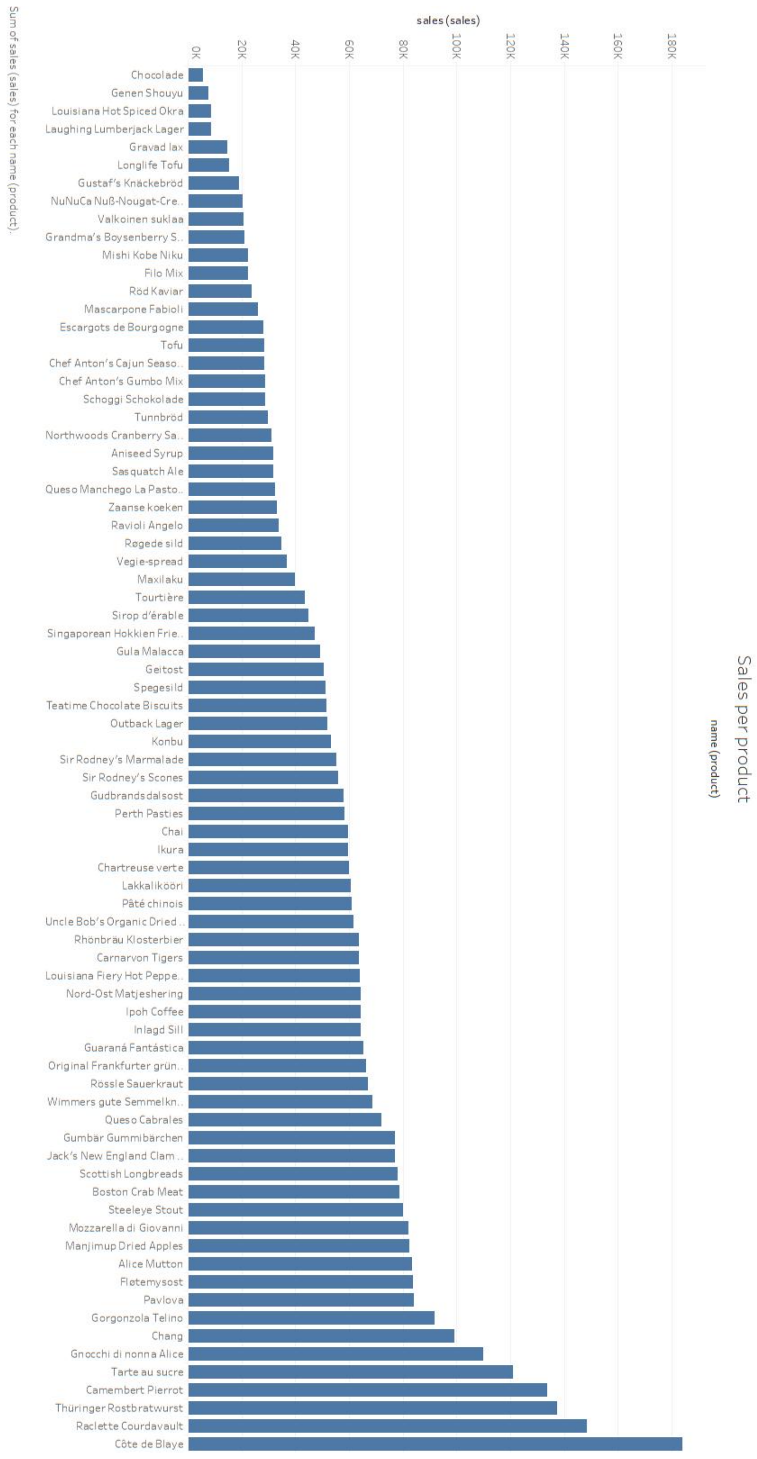

Figure 5: Histogram showing the sales per product

B Visualizations of assignment 3

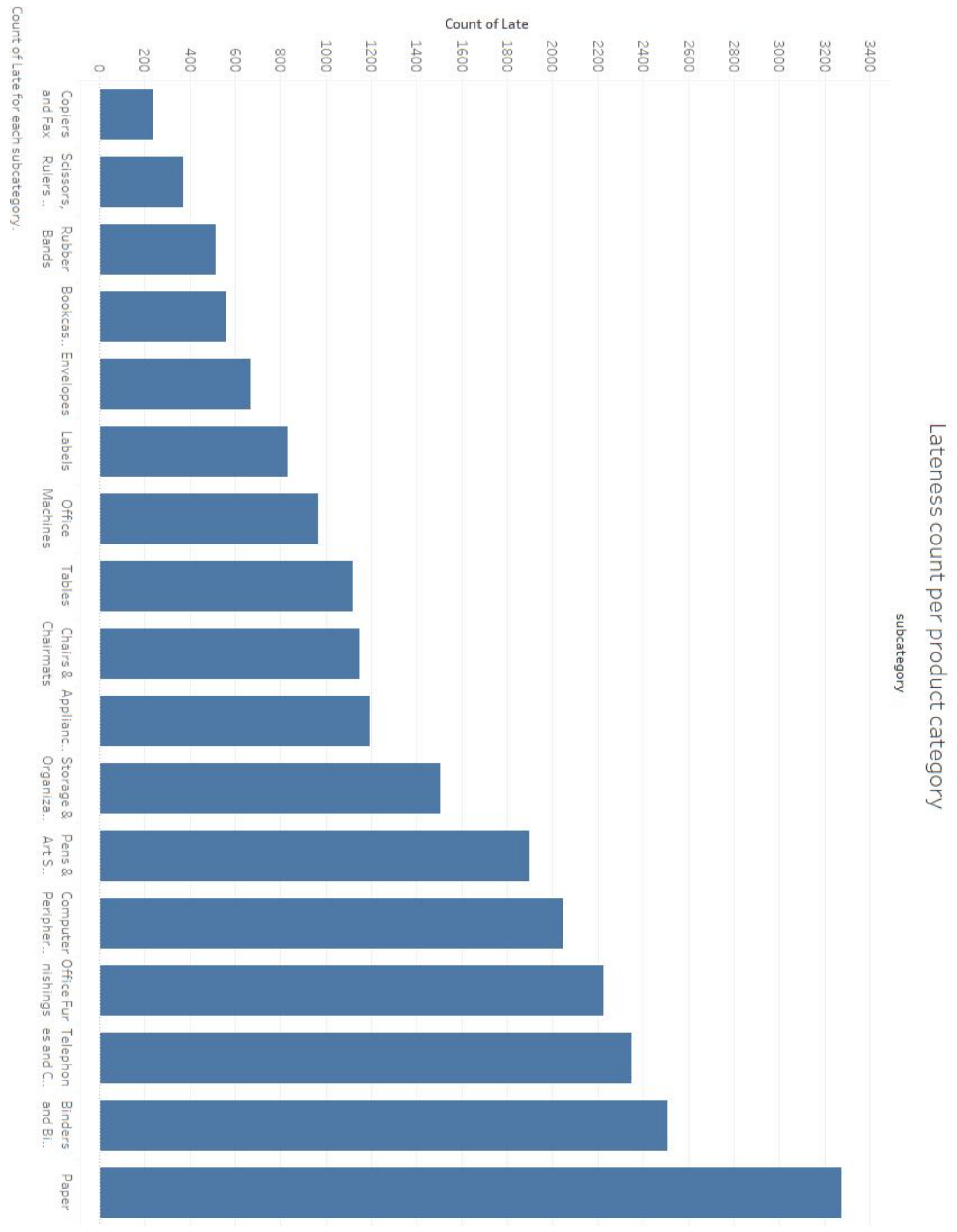

Figure 6: Histogram showing the number of products per product category that are too late

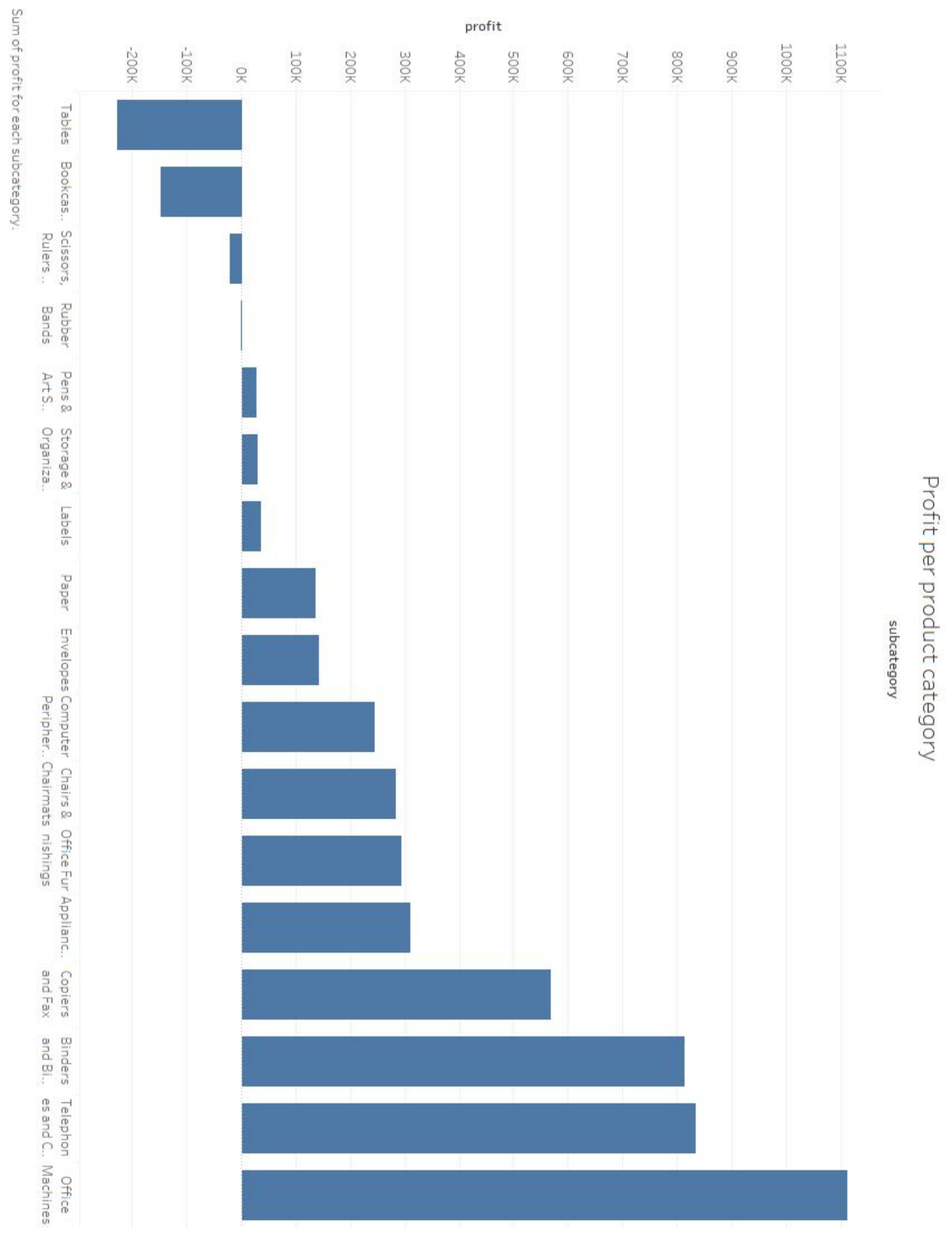

Figure 7: Histogram showing the profit per product category

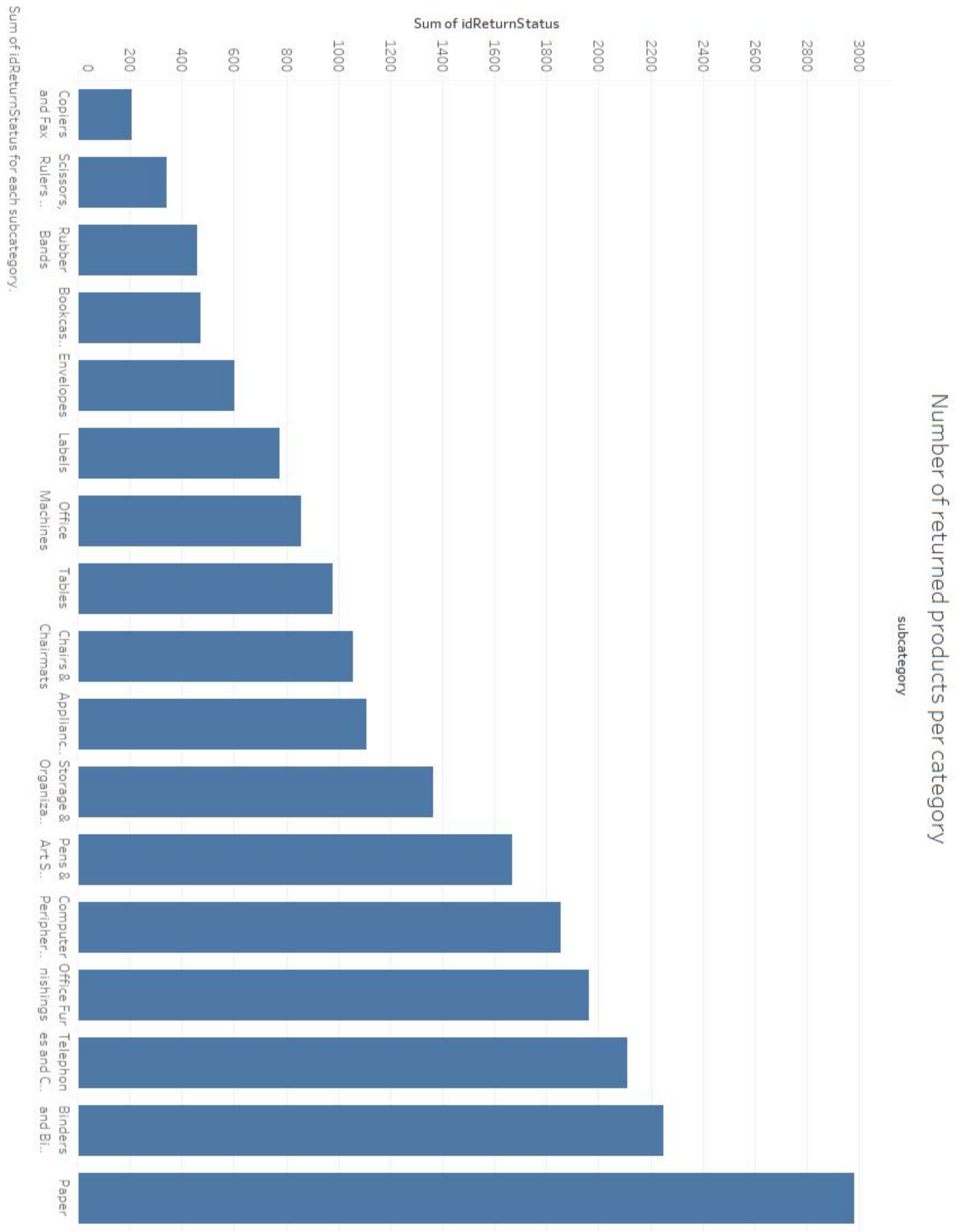

Figure 8: Histogram showing the number of products per product category that are returned