Product Analysis

Mukhtaar Ismail

2024-09-10

Install and load necessary packages.

I'll use "tidyverse" package because of there a lot of important packages include dplyr package wich we will use data manipulation.

```
require("tidyverse")
#install.packages("tidyverse")
#install.packages("naniar") # for dealing missing data (NA) data
#install.packages("SmartEDA")
#install.packages("ggcorrplot")

# Load necessary library
library("tidyverse")
library("lubridate")
library("magrittr")  # provides pipe operator, %>%, which allows you to chain functions together
library("naniar")    # exploring and dealing missing data
#library("dlookr")  # To visualizes the histogram of numeric data or relationship to specific categoric
library("SmartEDA")
library("ggcorrplot")
```

LET'S LOAD THE DATASET

```
# read the data
store <-read.csv("Data/DEPARTMENTAL_STORE.csv")
head(store)
```

```
##   UNIQUE_ID            PRODUCT_NAME COMPANY      PRODUCT_TYPE PRODUCT_CATEGORY
## 1     GSK01                 A Masks       A           hygiene             mask
## 2     GSK02 N hand sanitizer,350 ml       N           hygiene        sanitizer
## 3     GSK03       G Channa Dal, 1kG       G foodgrains&spices      pulses(dal)
## 4     GSK04  I Organic Raisins,100g       I      Organic food       Dry Fruits
## 5     GSK05              S Body oil       S   beauty products         bodycare
## 6     GSK06          S AloeverS Gel       S   beauty products         bodycare
##   COST_PRICE SELLING_PRICE QUANTITY_DEMANDED
## 1     160.00        200.00               890
## 2     248.00        400.00               800
## 3     162.00        180.00               456
## 4      77.14        133.00               100
## 5     494.50        593.40               111
## 6     445.05        534.06               111
```

Let's Manipulate and Transform data

let's glimpse the data to know more about the structure of the data

```
str(store)
```

```
## 'data.frame':    550 obs. of  8 variables:
##  $ UNIQUE_ID        : chr  "GSK01" "GSK02" "GSK03" "GSK04" ...
##  $ PRODUCT_NAME     : chr  "A Masks" "N hand sanitizer,350 ml" "G Channa Dal, 1kG" "I Organic Raisins
##  $ COMPANY          : chr  "A" "N" "G" "I" ...
##  $ PRODUCT_TYPE     : chr  "hygiene" "hygiene" "foodgrains&spices" "Organic food" ...
##  $ PRODUCT_CATEGORY : chr  "mask" "sanitizer" "pulses(dal)" "Dry Fruits" ...
##  $ COST_PRICE       : num  160 248 162 77.1 494.5 ...
##  $ SELLING_PRICE    : num  200 400 180 133 593 ...
##  $ QUANTITY_DEMANDED: int  890 800 456 100 111 111 360 52 27 353 ...
```

```
glimpse(store)
```

```
## Rows: 550
## Columns: 8
## $ UNIQUE_ID         <chr> "GSK01", "GSK02", "GSK03", "GSK04", "GSK05", "GSK06"~
## $ PRODUCT_NAME      <chr> "A Masks", "N hand sanitizer,350 ml", "G Channa Dal,~
## $ COMPANY           <chr> "A", "N", "G", "I", "S", "S", "S", "G", "Z", "A", "Z~
## $ PRODUCT_TYPE      <chr> "hygiene", "hygiene", "foodgrains&spices", "Organic ~
## $ PRODUCT_CATEGORY  <chr> "mask", "sanitizer", "pulses(dal)", "Dry Fruits", "b~
## $ COST_PRICE        <dbl> 160.00, 248.00, 162.00, 77.14, 494.50, 445.05, 270.0~
## $ SELLING_PRICE     <dbl> 200.000, 400.000, 180.000, 133.000, 593.400, 534.060~
## $ QUANTITY_DEMANDED <int> 890, 800, 456, 100, 111, 111, 360, 52, 27, 353, 500,~
```

```
dim(store)
```

```
## [1] 550   8
```

There is 550 rows and 8 variables. 'COMPANY' and 'PRODUCT_TYPE' variables are chr type instead of factor, let's convert into factor type.

```
store$COMPANY <- as.factor(store$COMPANY)
store$PRODUCT_TYPE <- as.factor(store$PRODUCT_TYPE)

# let's view the data type
glimpse(store) # both now are factor type
```

```
## Rows: 550
## Columns: 8
## $ UNIQUE_ID         <chr> "GSK01", "GSK02", "GSK03", "GSK04", "GSK05", "GSK06"~
## $ PRODUCT_NAME      <chr> "A Masks", "N hand sanitizer,350 ml", "G Channa Dal,~
## $ COMPANY           <fct> A, N, G, I, S, S, S, G, Z, A, Z, I, E, N, A, J, Z, Z~
## $ PRODUCT_TYPE      <fct> hygiene, hygiene, foodgrains&spices, Organic food, b~
## $ PRODUCT_CATEGORY  <chr> "mask", "sanitizer", "pulses(dal)", "Dry Fruits", "b~
## $ COST_PRICE        <dbl> 160.00, 248.00, 162.00, 77.14, 494.50, 445.05, 270.0~
## $ SELLING_PRICE     <dbl> 200.000, 400.000, 180.000, 133.000, 593.400, 534.060~
## $ QUANTITY_DEMANDED <int> 890, 800, 456, 100, 111, 111, 360, 52, 27, 353, 500,~
```

Now the two variables are factor type.

Adding columns in data

The data contains details of products from May, 2020, a period marked by covid-19. there is no date variable, so let's create datetime column from May 1 to May 30 2020 hour interval. I used 550 values since their is 744 hours in the month to match rows of the data.

```
# Define start and end dates
start_date <- ymd_hms("2020-05-01 00:00:00")
end_date <- ymd_hms("2020-05-31 23:00:00")

# Generate sequence of datetime values in hourly intervals
datetime_sequence <- seq(start_date, end_date, by = "hour")

#create datetime column in store
store <- mutate(store, Datetime = datetime_sequence[1:550])
head(store)
```

```
##   UNIQUE_ID            PRODUCT_NAME COMPANY      PRODUCT_TYPE PRODUCT_CATEGORY
## 1     GSK01                 A Masks       A           hygiene             mask
## 2     GSK02 N hand sanitizer,350 ml       N           hygiene        sanitizer
## 3     GSK03       G Channa Dal, 1kG       G foodgrains&spices      pulses(dal)
## 4     GSK04  I Organic Raisins,100g       I      Organic food       Dry Fruits
## 5     GSK05              S Body oil       S   beauty products         bodycare
## 6     GSK06          S AloeverS Gel       S   beauty products         bodycare
##   COST_PRICE SELLING_PRICE QUANTITY_DEMANDED            Datetime
## 1     160.00        200.00               890 2020-05-01 00:00:00
## 2     248.00        400.00               800 2020-05-01 01:00:00
## 3     162.00        180.00               456 2020-05-01 02:00:00
## 4      77.14        133.00               100 2020-05-01 03:00:00
## 5     494.50        593.40               111 2020-05-01 04:00:00
## 6     445.05        534.06               111 2020-05-01 05:00:00
```

```
# add column to show the profit
store <- store %>% mutate(PROFIT= SELLING_PRICE - COST_PRICE)

# add column to show the profit percent
store <- store %>% mutate(PROFIT_PERCENT= (PROFIT / COST_PRICE) *100)

# add column to show the net profit
store <- store %>% mutate(NET_PROFIT=PROFIT*QUANTITY_DEMANDED)

head(store)
```

let's add one more columns to show profit, profit percent, and net profit of the products.

```
##   UNIQUE_ID            PRODUCT_NAME COMPANY      PRODUCT_TYPE PRODUCT_CATEGORY
## 1     GSK01                 A Masks       A           hygiene             mask
## 2     GSK02 N hand sanitizer,350 ml       N           hygiene        sanitizer
## 3     GSK03       G Channa Dal, 1kG       G foodgrains&spices      pulses(dal)
## 4     GSK04  I Organic Raisins,100g       I      Organic food       Dry Fruits
```

```
## 5     GSK05              S Body oil       S   beauty products         bodycare
## 6     GSK06          S AloeverS Gel       S   beauty products         bodycare
##   COST_PRICE SELLING_PRICE QUANTITY_DEMANDED            Datetime PROFIT
## 1     160.00        200.00               890 2020-05-01 00:00:00  40.00
## 2     248.00        400.00               800 2020-05-01 01:00:00 152.00
## 3     162.00        180.00               456 2020-05-01 02:00:00  18.00
## 4      77.14        133.00               100 2020-05-01 03:00:00  55.86
## 5     494.50        593.40               111 2020-05-01 04:00:00  98.90
## 6     445.05        534.06               111 2020-05-01 05:00:00  89.01
##   PROFIT_PERCENT NET_PROFIT
## 1       25.00000   35600.00
## 2       61.29032  121600.00
## 3       11.11111    8208.00
## 4       72.41379    5586.00
## 5       20.00000   10977.90
## 6       20.00000    9880.11
```

Rearrarange Columns Order Using dplyr's Relocate Function.

Rearranging columns in a meaningful order can make analyses simpler. Let's put datetime column in the first column.

```
store <- store %>% relocate(Datetime, .before = UNIQUE_ID)
head(store)
```

```
##              Datetime UNIQUE_ID            PRODUCT_NAME COMPANY
## 1 2020-05-01 00:00:00     GSK01                 A Masks       A
## 2 2020-05-01 01:00:00     GSK02 N hand sanitizer,350 ml       N
## 3 2020-05-01 02:00:00     GSK03       G Channa Dal, 1kG       G
## 4 2020-05-01 03:00:00     GSK04  I Organic Raisins,100g       I
## 5 2020-05-01 04:00:00     GSK05              S Body oil       S
## 6 2020-05-01 05:00:00     GSK06          S AloeverS Gel       S
##        PRODUCT_TYPE PRODUCT_CATEGORY COST_PRICE SELLING_PRICE QUANTITY_DEMANDED
## 1           hygiene             mask     160.00        200.00               890
## 2           hygiene        sanitizer     248.00        400.00               800
## 3 foodgrains&spices      pulses(dal)     162.00        180.00               456
## 4      Organic food       Dry Fruits      77.14        133.00               100
## 5   beauty products         bodycare     494.50        593.40               111
## 6   beauty products         bodycare     445.05        534.06               111
##   PROFIT PROFIT_PERCENT NET_PROFIT
## 1  40.00       25.00000   35600.00
## 2 152.00       61.29032  121600.00
## 3  18.00       11.11111    8208.00
## 4  55.86       72.41379    5586.00
## 5  98.90       20.00000   10977.90
## 6  89.01       20.00000    9880.11
```

DEALING WITH MISSING VALUE

Identifying the Pattern of Missing Values When exploring a new dataset, it's worthwhile to identify the pattern of missing values. The `summary()` includes the number of missing values for each column along with the summary statistics.

```r
summary(store)
```

```
##     Datetime                       UNIQUE_ID         PRODUCT_NAME
##  Min.   :2020-05-01 00:00:00   Length:550         Length:550
##  1st Qu.:2020-05-06 17:15:00   Class :character   Class :character
##  Median :2020-05-12 10:30:00   Mode  :character   Mode  :character
##  Mean   :2020-05-12 10:30:00
##  3rd Qu.:2020-05-18 03:45:00
##  Max.   :2020-05-23 21:00:00
##
##     COMPANY               PRODUCT_TYPE PRODUCT_CATEGORY     COST_PRICE
##  S      : 97   beauty products  :100   Length:550         Min.   :  9.00
##  G      : 84   foodgrains&spices: 91   Class :character   1st Qu.: 72.97
##  Z      : 66   Packed Food      : 84   Mode  :character   Median :180.06
##  A      : 57   Organic food     : 82                      Mean   :216.53
##  B      : 53   snacks           : 64                      3rd Qu.:320.54
##  I      : 36   hygiene          : 54                      Max.   :912.00
##  (Other):157   (Other)          : 75
##  SELLING_PRICE    QUANTITY_DEMANDED     PROFIT       PROFIT_PERCENT
##  Min.   :  10.0   Min.   :   2.0    Min.   :  0.66   Min.   : 5.00
##  1st Qu.:  98.5   1st Qu.: 120.0    1st Qu.: 18.71   1st Qu.:15.00
##  Median : 229.0   Median : 316.0    Median : 41.16   Median :20.00
##  Mean   : 274.9   Mean   : 327.2    Mean   : 58.41   Mean   :29.82
##  3rd Qu.: 422.0   3rd Qu.: 500.0    3rd Qu.: 80.81   3rd Qu.:31.58
##  Max.   :1200.0   Max.   :1100.0    Max.   :288.00   Max.   :90.00
##
##    NET_PROFIT
##  Min.   :   223.4
##  1st Qu.:  4299.4
##  Median :  9350.0
##  Mean   : 12092.5
##  3rd Qu.: 13661.9
##  Max.   :136800.0
##
```

```r
# find to of missing values
sum_missing_values <- sum(is.na(store))
print(paste("The total of missing values are:", sum_missing_values))  # 0
```

```
## [1] "The total of missing values are: 0"
```

```r
# summary of missing variables
miss_var_summary(store)
```

```
## # A tibble: 12 x 3
##    variable          n_miss pct_miss
##    <chr>              <int>    <num>
##  1 Datetime               0        0
##  2 UNIQUE_ID              0        0
##  3 PRODUCT_NAME           0        0
##  4 COMPANY                0        0
##  5 PRODUCT_TYPE           0        0
```

```
##  6 PRODUCT_CATEGORY       0        0
##  7 COST_PRICE             0        0
##  8 SELLING_PRICE          0        0
##  9 QUANTITY_DEMANDED      0        0
## 10 PROFIT                 0        0
## 11 PROFIT_PERCENT         0        0
## 12 NET_PROFIT             0        0
```

As shown the result, there is no missing values. If a column has a large percentage of missing values, then you may want to consider dropping that column all together. However, if you are really interested in the effect of that column, then a better idea is to remove the observations that have missing values.

Identifying Distinct Values Using dplyr's Distinct Function

The `distinct` function returns only the unique values from a column.

```
dim(distinct(store))
```

```
## [1] 550  12
```

As shown the result, all the values are unique.

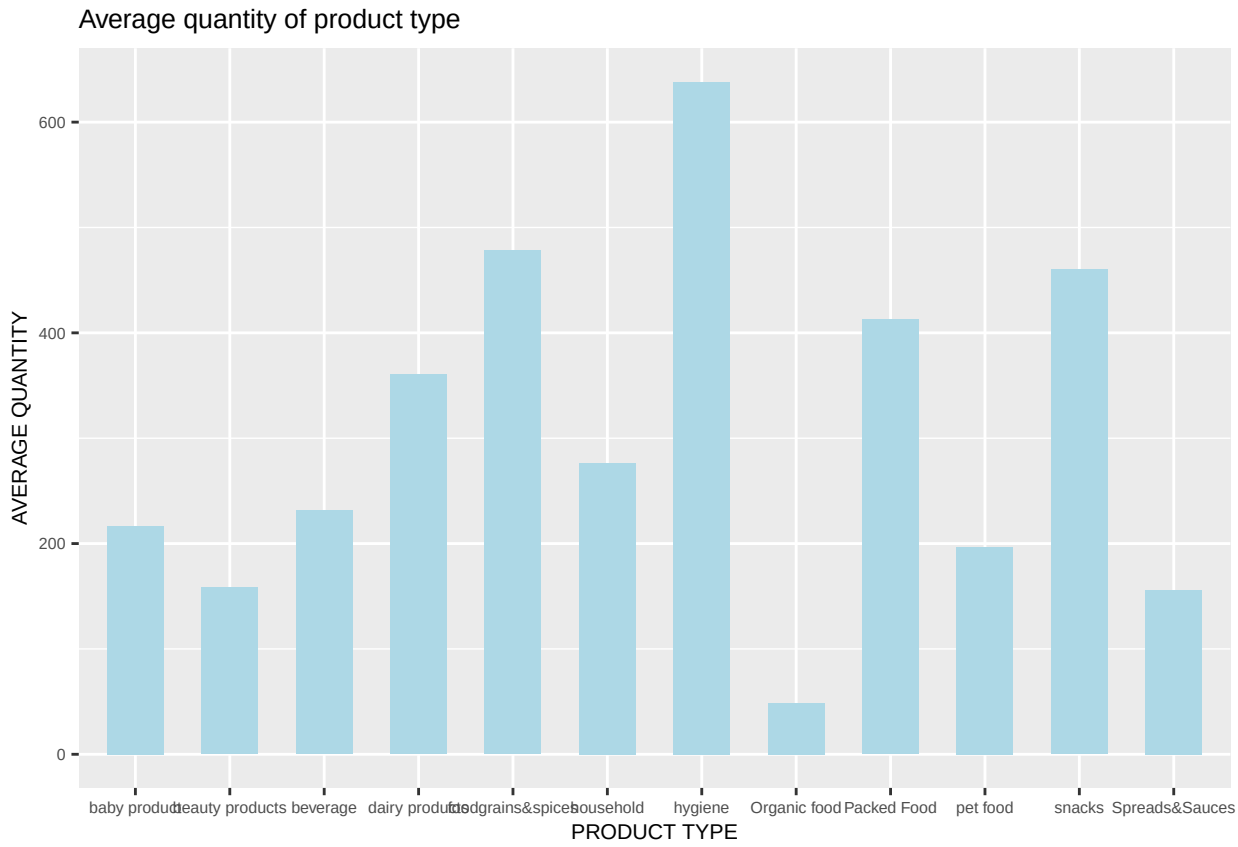
Save the updates file

```
# Save up the updates for reusing. Uncomment if want to run
#write.table(store, file = "Updated_store_data.csv", sep=",")
```

Let's Spot patterns and problems using graphs and visualizations

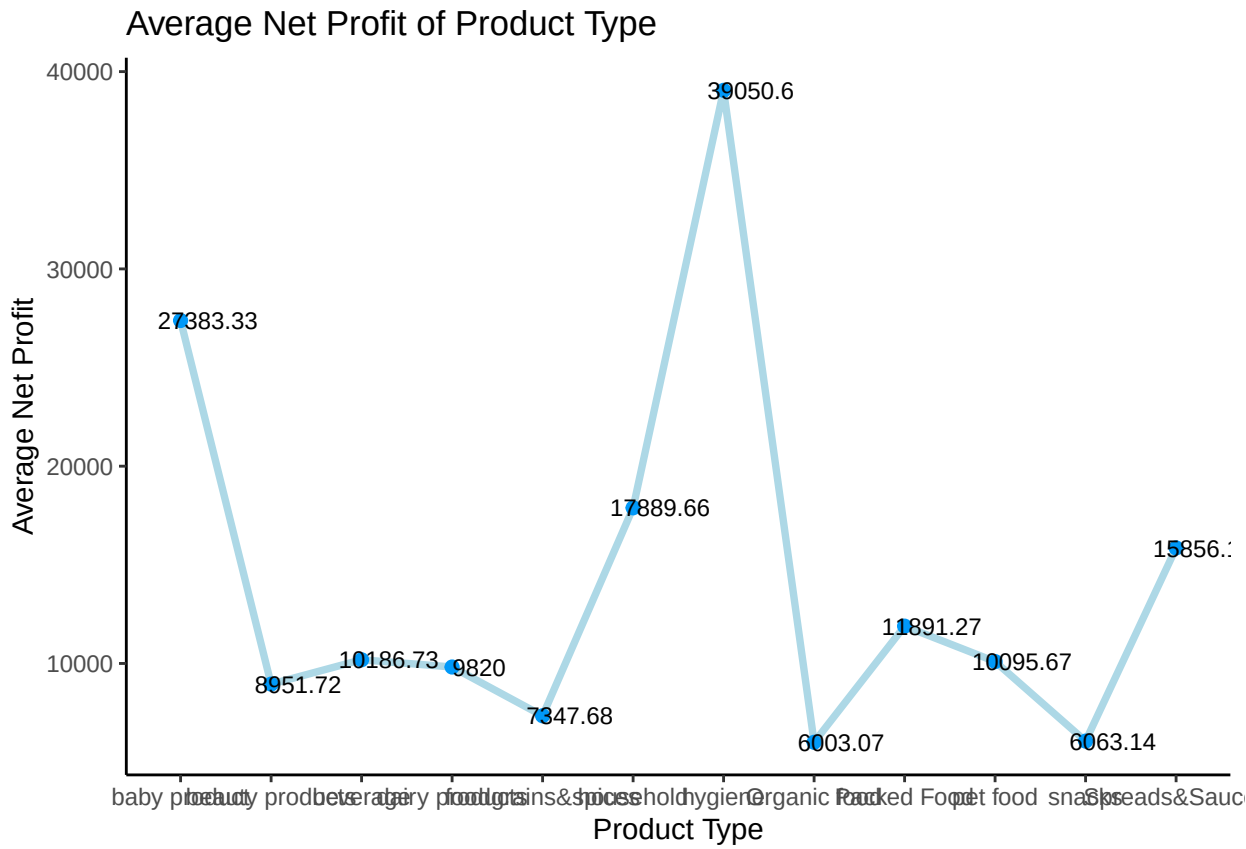
What is the AVERAGE_QUANTITY of PRODUCT_TYPE

```
#PLOT FOR AVERAGE_QUANTITY & PRODUCT_TYPE
store %>% group_by(PRODUCT_TYPE) %>%
  summarise(AVERAGE_QUANTITY=mean(QUANTITY_DEMANDED),
            AVERAGE_NET_PROFIT=mean(NET_PROFIT)) %>%
  ggplot(aes(x=PRODUCT_TYPE, y=AVERAGE_QUANTITY))+geom_col(width=0.6, fill="lightblue")+
  labs(x="PRODUCT TYPE", y="AVERAGE QUANTITY", title = "Average quantity of product type")+
  theme(text= element_text(size=8))
```



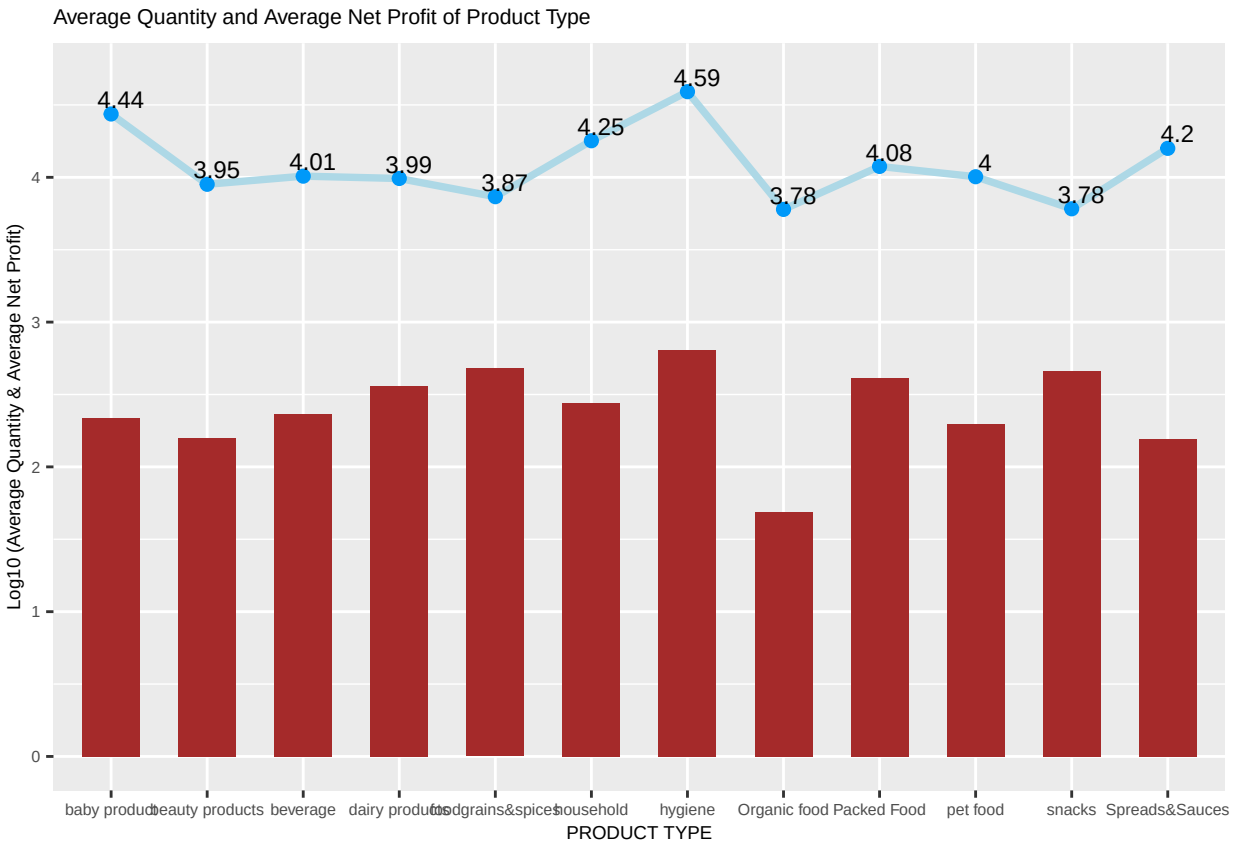
Let's plot the Average Net Profit of Product Type

```
store %>% group_by(PRODUCT_TYPE) %>%
  summarise(AVERAGE_NET_PROFIT=mean(NET_PROFIT)) %>%
  ggplot(aes(x=PRODUCT_TYPE, y= AVERAGE_NET_PROFIT, group = 3))+
  geom_line(color="lightblue", linewidth=1.3)+
  geom_point(color = "#0099f9", size = 2)+
  theme(text= element_text(size=7))+
  geom_text(aes(label = round(AVERAGE_NET_PROFIT,2)), nudge_x = 0.3,
            nudge_y = 0.2,
            size = 3)+
  theme_classic()+
  labs(x="Product Type", y="Average Net Profit", title = "Average Net Profit of Product Type")
```



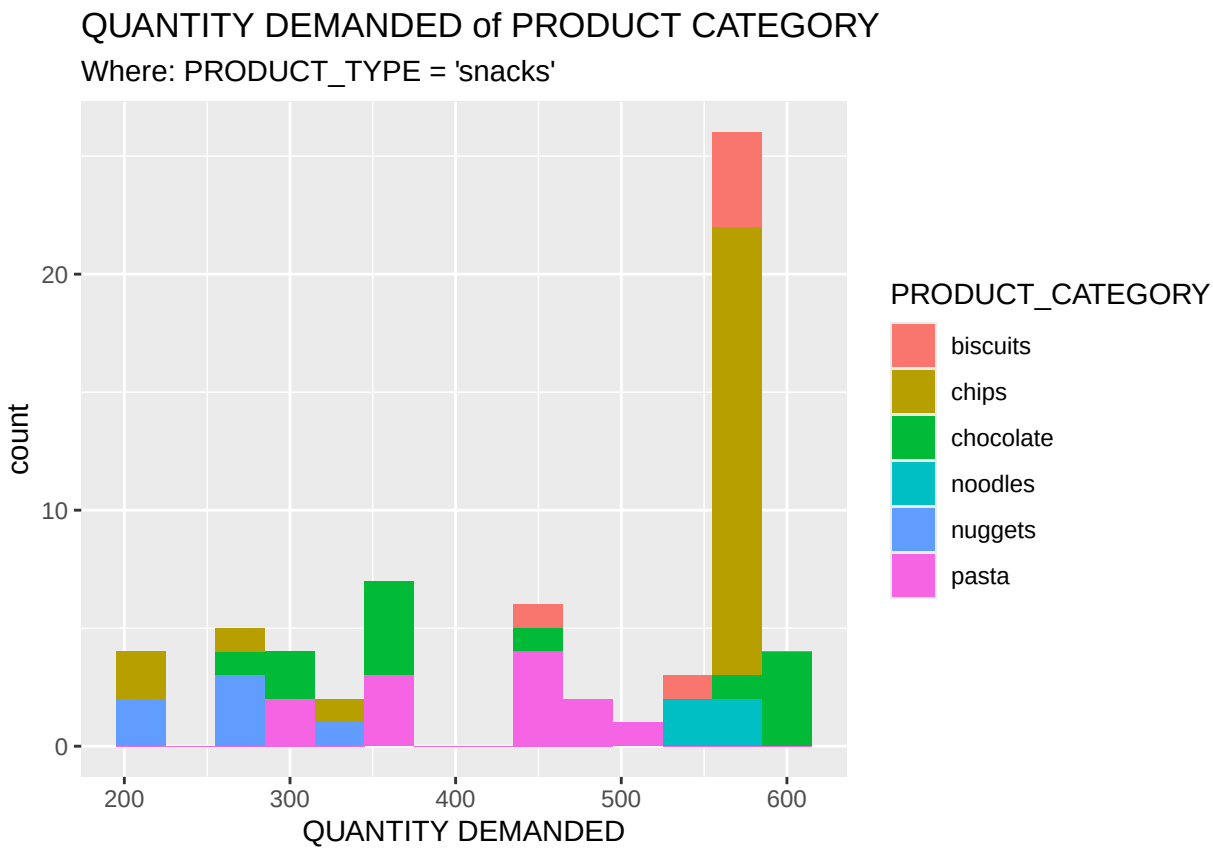
Let's plot the Average Quantity and Average Net Profit of Product Type

```
#Average quantity and AVERAGE NET PROFIT  of product type
store %>% group_by(PRODUCT_TYPE) %>%
  summarise(AVERAGE_QUANTITY=mean(QUANTITY_DEMANDED),
            AVERAGE_NET_PROFIT=mean(NET_PROFIT)) %>%
  ggplot(aes(x=PRODUCT_TYPE))+
  geom_col(aes(y = log10(AVERAGE_QUANTITY)), width=0.6, fill="brown")+
  geom_line(aes(y=log10(AVERAGE_NET_PROFIT), group=1),color="lightblue", linewidth=1.3)+
  geom_point(aes(y = log10(AVERAGE_NET_PROFIT)),color = "#0099f9", size = 2)+
  geom_text(aes(y=log10(AVERAGE_NET_PROFIT),label =  round(log10(AVERAGE_NET_PROFIT),2)),
            nudge_x = 0.1,
            nudge_y = 0.1,
            size = 3)+
  theme(text= element_text(size=7))+
  labs(x="PRODUCT TYPE", y="Log10 (Average Quantity & Average Net Profit)", title = "Average Quantity a
```



Plot histogram for QUANTITY DEMANDED of PRODUCT CATEGORY where PRODUCT TYPE is "snacks"

```
# HISTOGRAM FOR QUANTITY_DEMANDED OF PRODUCT_CATEGORY WHERE PRODUCT_TYPE IS "snacks"
store %>%
  filter(PRODUCT_TYPE == "snacks") %>%
  ggplot(aes(x=QUANTITY_DEMANDED, fill=PRODUCT_CATEGORY))+geom_histogram(binwidth = 30)+
  labs(x="QUANTITY DEMANDED", title = "QUANTITY DEMANDED of PRODUCT CATEGORY",
       subtitle="Where: PRODUCT_TYPE = 'snacks'")
```

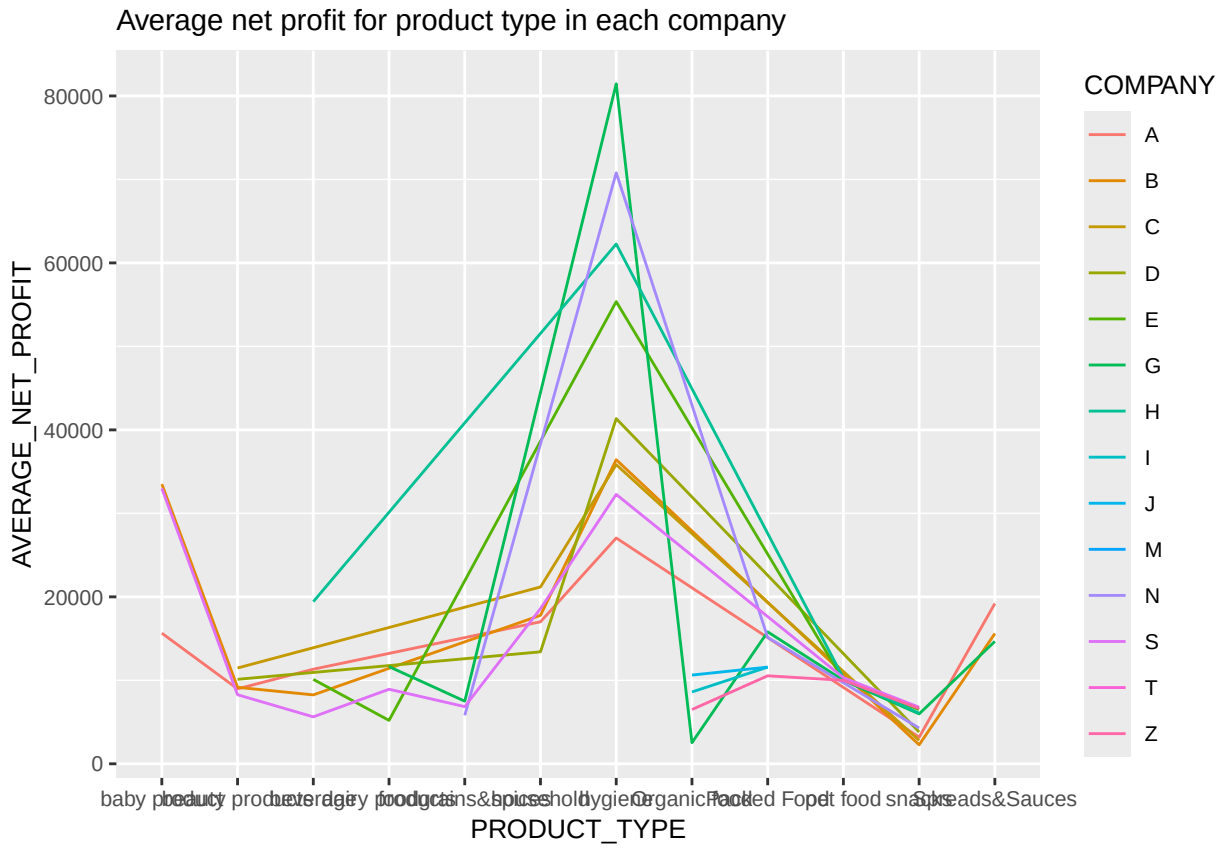


Plot average net profit for product type in each company

```
# PLOT FOR AVERAGE_NET_PROFIT & COMPANY

store %>% group_by(PRODUCT_TYPE, COMPANY) %>%
  summarise(AVERAGE_NET_PROFIT=mean(NET_PROFIT, na.rm=TRUE)) %>%
  ggplot(aes(x=PRODUCT_TYPE, y=AVERAGE_NET_PROFIT, group=COMPANY, colour = COMPANY))+
  geom_line()+ theme(text= element_text(size=9.5))+
  ggtitle("Average net profit for product type in each company")
```

```
## `summarise()` has grouped output by 'PRODUCT_TYPE'. You can override using the
## `.groups` argument.
```



Let's make a pie chart for each HYGIENE PRODUCT'S QUANTITY DEMANDED

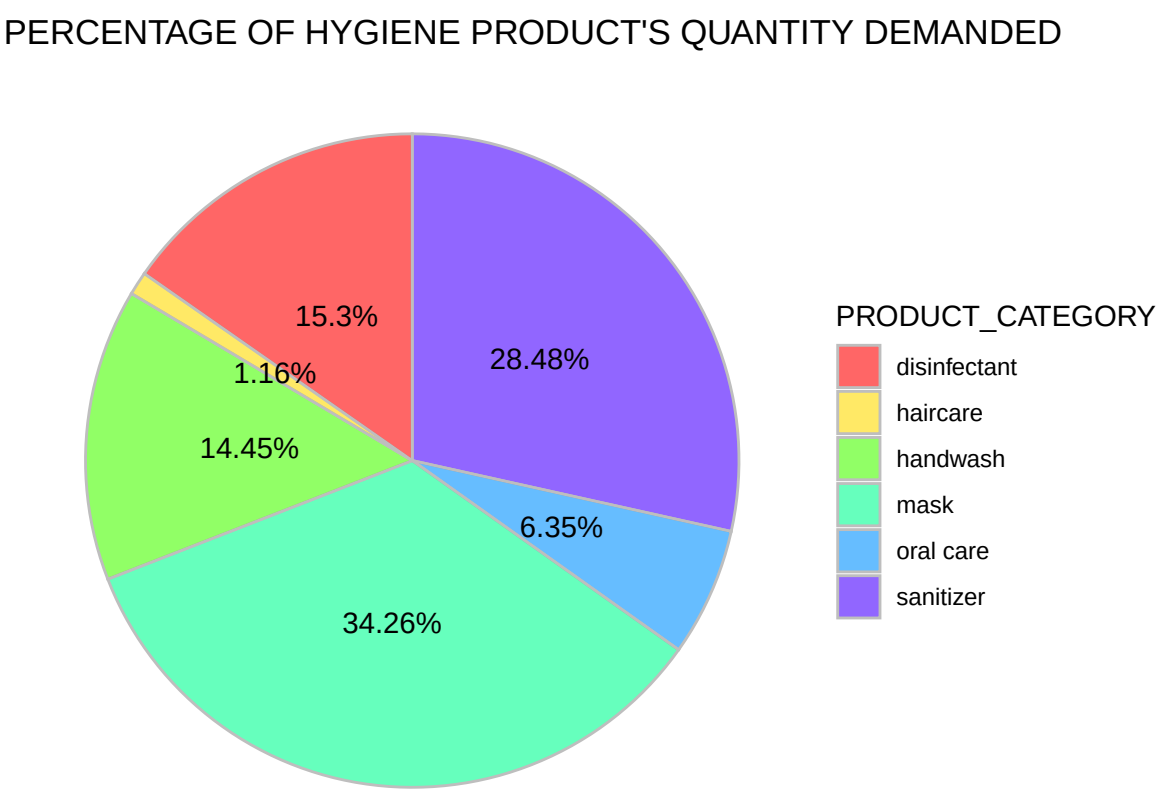
```
#LET'S PREPARE REQUIRD DATA

hygiene_prod <-  store %>% filter(PRODUCT_TYPE=="hygiene")%>%
  group_by(PRODUCT_CATEGORY)%>%
  summarise(QUANTITY_DEMANDED=sum(QUANTITY_DEMANDED))

# LET'S CALCULATE PERCENTAGE OF EACH PRODUCT

hygiene_prod_perc <- hygiene_prod %>%
  arrange(desc(PRODUCT_CATEGORY)) %>%
  mutate(percentage=round(QUANTITY_DEMANDED*100/sum(QUANTITY_DEMANDED),2)) %>%
  mutate(y_pos = cumsum(percentage)-0.5*percentage)

# LET'S CREATE THE PIE CHART
hygiene_prod_perc %>% ggplot(aes(x="", percentage, fill = PRODUCT_CATEGORY))+
  geom_bar(width = 1, stat = "identity", color="grey", alpha=0.6)+
  coord_polar("y", start = 0)+
  geom_text(aes(y=y_pos, label = paste0(percentage, "%")), color="black")+
  scale_fill_manual(values = rainbow(7))+
  ggtitle("PERCENTAGE OF HYGIENE PRODUCT'S QUANTITY DEMANDED")+
  theme_void()
```

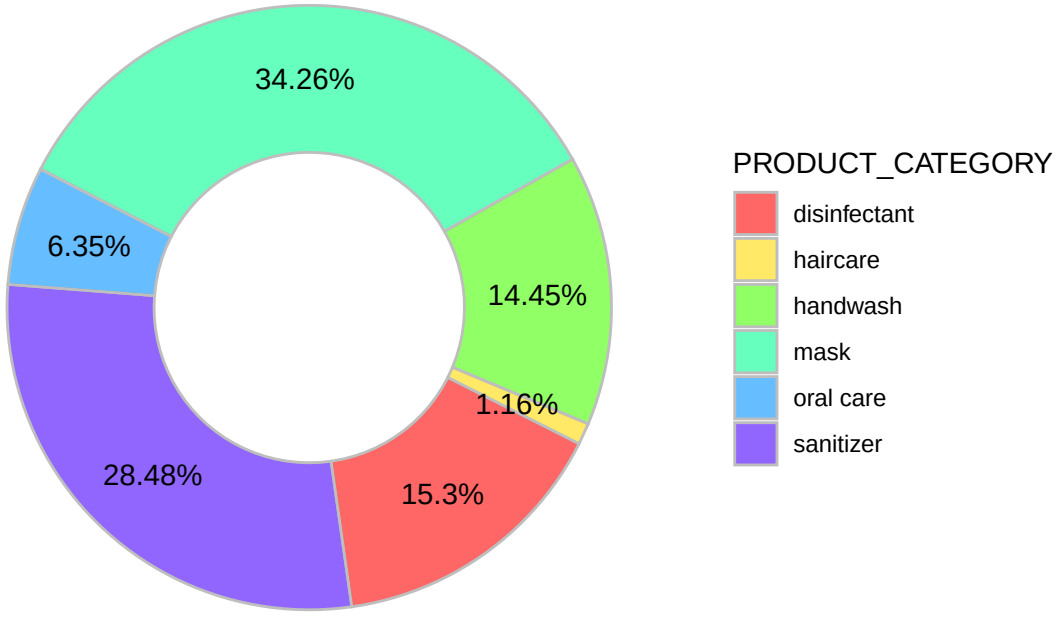


Plot a donut chart for same data

```
# LET'S MAKE A DONUT FOR THE SAME DATA

hygiene_prod_perc %>% ggplot(aes(x=2, percentage, fill = PRODUCT_CATEGORY))+
  geom_bar(stat = "identity", color="grey", alpha=0.6)+
  coord_polar(theta = "y", start = 3)+
  geom_text(aes(y=y_pos, label = paste0(percentage, "%")), color="black")+
  ggtitle("A DONUT CHART OF HYGIENE PRODUCT'S QUANTITY DEMANDED")+
  scale_fill_manual(values = rainbow(7))+ theme_void() + xlim(0.6,2.6)
```

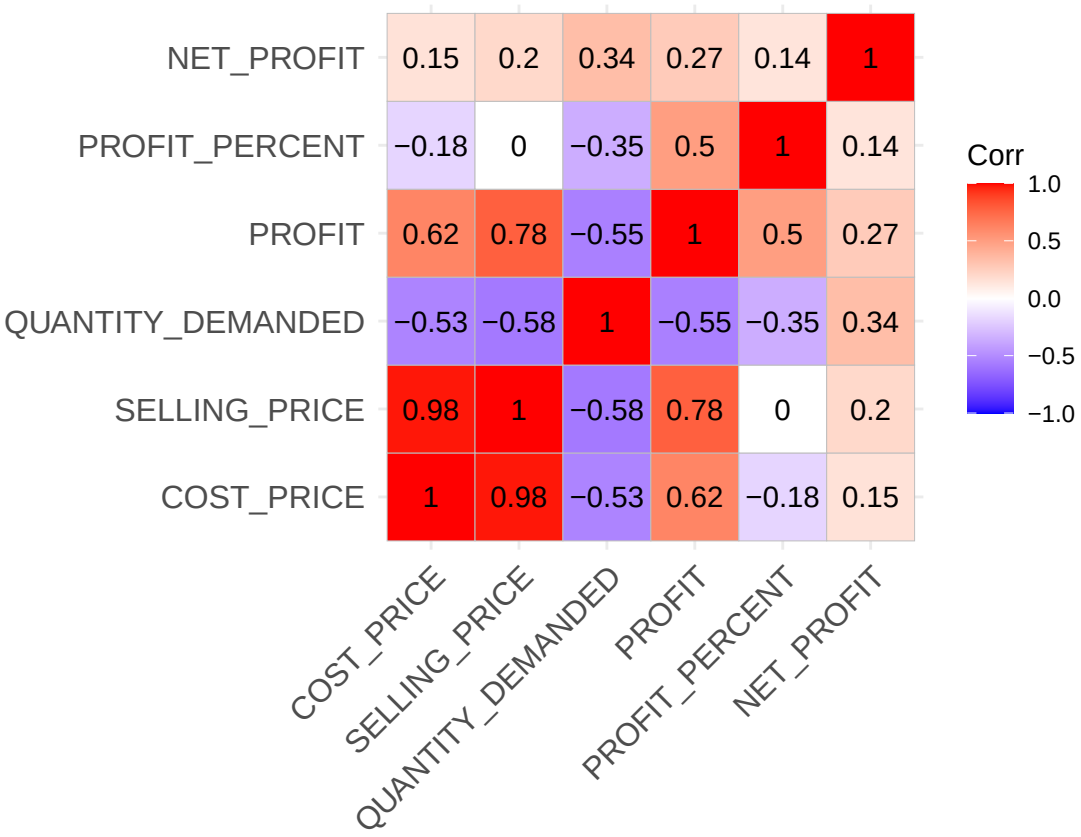
A DONUT CHART OF HYGIENE PRODUCT'S QUANTITY DEMANDED



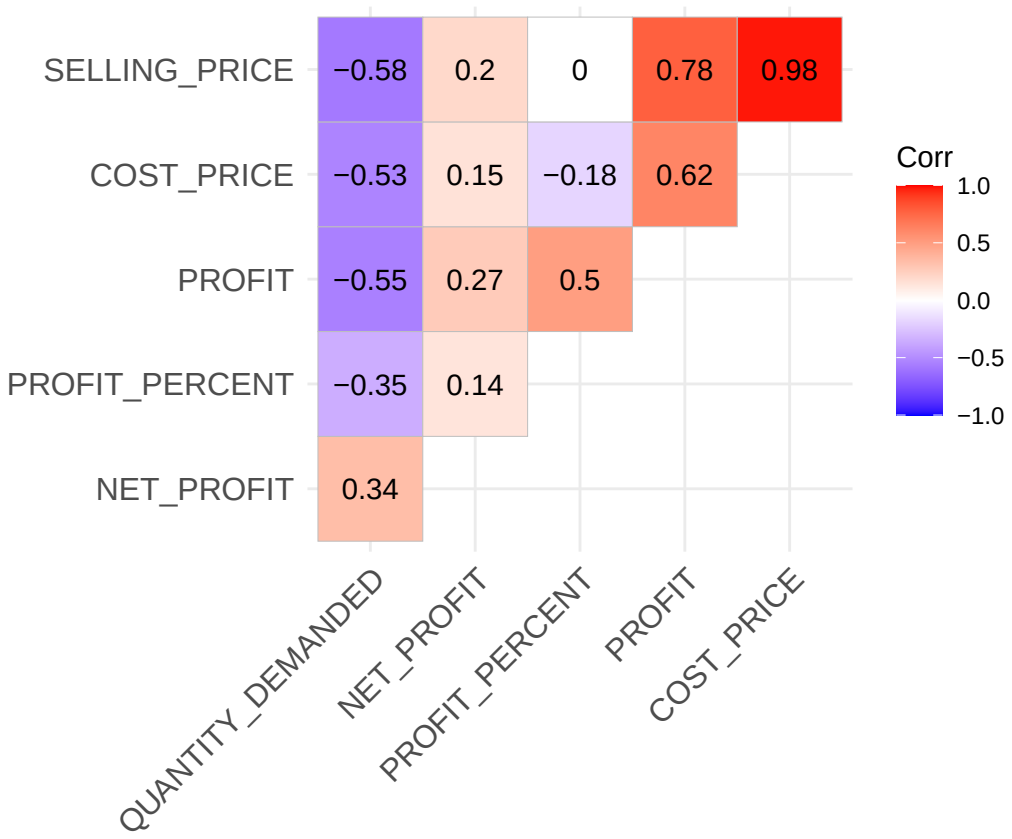
Correlation refers to the statistical concept that studies the relationship between two quantitative variables. Let's find and plot the correlation martix of numerical variables using ggcorrplot package.

```
numeric_var<- select_if(store, is.numeric) # get numerical features
r_corr<-cor(numeric_var, use = "complete.obs")

#PLOT THE CORRELATION MATRIX (HEAT MAP)
ggcorrplot(r_corr, lab = TRUE)
```




```r
# PLOT THE SORTED UPPER TRIANGLE HEAT MAP
ggcorrplot(r_corr,hc.order = TRUE, type = "upper", lab = TRUE)
```



Let's know the distribution of some numerical variables.

```
# plot histogram & density of cost_price
ggplot(store, aes(x= COST_PRICE))+
  geom_histogram(aes(y=..density..), binwidth=30, colour="black", fill="white")+
  geom_density(alpha=.2, fill="#FF6666")+
  ggtitle("Distribution & Density of cost price")
```

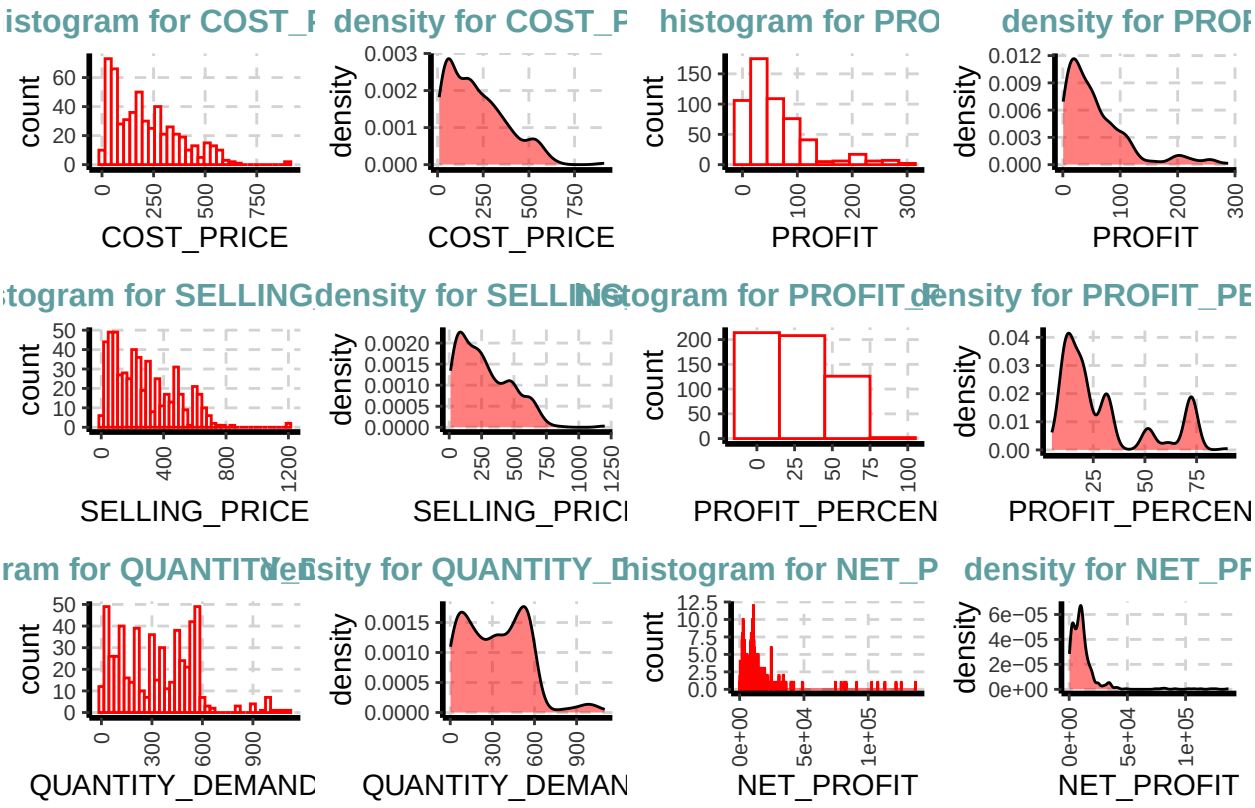
```
## Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(density)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

Density and Distribution of numerical variables

```
numeric_var<- colnames(select_if(store, is.numeric))

ExpTwoPlots(store, iv_variables = numeric_var,
            lp_geom_type = "histogram",
            lp_arg_list=list(fill = "white",color = "red",  binwidth=30),
            rp_geom_type= "density",
            rp_arg_list = list(alpha=0.5, fill="red"),
            page = c(3,2)
            )
```

```
## $`0`
```



Let's dive more about statistics in numerical variables using ExpNumStat() function in smartEDA package.

```
ExpNumStat(store)
```

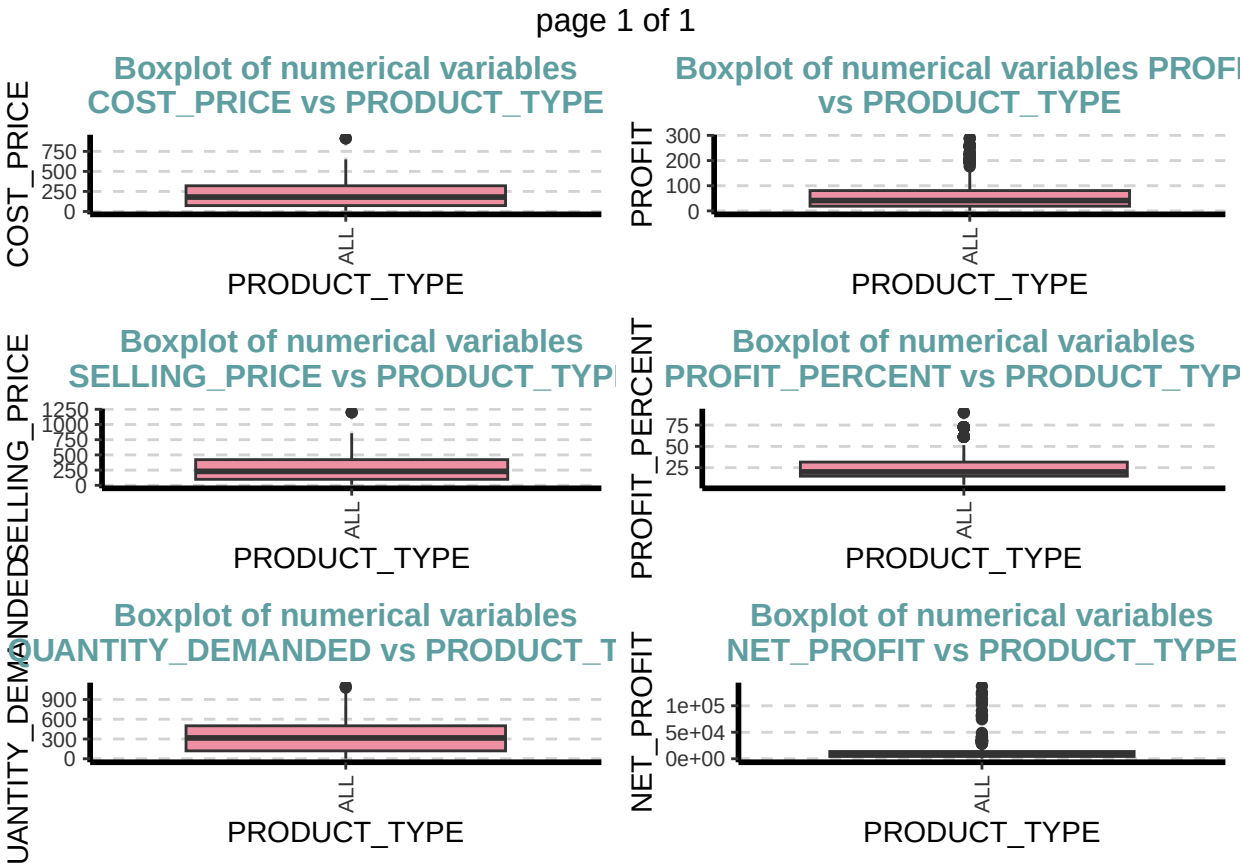
```
##              Vname Group  TN nNeg nZero nPos NegInf PosInf NA_Value
## 1        COST_PRICE   All 550    0     0  550      0      0        0
## 6        NET_PROFIT   All 550    0     0  550      0      0        0
## 4            PROFIT   All 550    0     0  550      0      0        0
## 5    PROFIT_PERCENT   All 550    0     0  550      0      0        0
## 3 QUANTITY_DEMANDED   All 550    0     0  550      0      0        0
## 2     SELLING_PRICE   All 550    0     0  550      0      0        0
##   Per_of_Missing        sum    min    max      mean  median        SD    CV
## 1              0  119091.37   9.00    912   216.530  180.06   161.829 0.747
## 6              0 6650871.60 223.44 136800 12092.494 9350.00 15220.272 1.259
## 4              0   32126.38   0.66    288    58.412   41.16    56.666 0.970
## 5              0   16402.13   5.00     90    29.822   20.00    21.719 0.728
## 3              0  179944.00   2.00   1100   327.171  316.00   224.432 0.686
## 2              0  151217.75  10.00   1200   274.941  229.00   202.085 0.735
##        IQR Skewness Kurtosis
## 1  247.570    0.876    0.469
## 6 9362.496    4.807   28.880
## 4   62.103    1.775    3.195
## 5   16.579    1.095   -0.240
## 3  380.000    0.564    0.230
## 2  323.500    0.837    0.547
```

The distribution of all numeric variables are right-tail distribution. As shown histogram and density in the above graph, there is other two ways I can confirm this claim. First, all mean of numerical variables are greater than their median, this indicates that the distribution is right-tailed. Secondly, all the skewness values are greater than 0, this shows this right-tailed distribution. There is highly positive skewness variables such NET_PROFIT, PROFIT_PERCENT , and PROFIT. and moderate skewness variables include COST_PRICE, SELLING_PRICE, and QUANTITY_DEMANDED.

In addition, the Kurtosis indicates the presence of outliers. Until now, there is one variable which have high Kurtosis value (NET_PROFIT). This shows there is more outliers in this variable, but I'll confirm by plotting boxplot using smartEDA package.

```
ExpNumViz(store, target = "PRODUCT_TYPE",type = 3, Page = c(3,2), gtitle = "Boxplot of numerical variabl
```

```
## $`0`
```



As the result above shows, NET_PROFIT, PROFIT_PERCENT , and PROFIT have more outliers than other variables. Specialy, NET_PROFIT variable contain most outliers compared with others. This confirm the fact that NET_PROFIT has high Kurtosis value (28.8).

Note: If you are preparing this data for modeling, you need to solve and transform some of issues in variables like skewness of numerical variables and remove outliers.