```
install.packages("data.table")

#### Load required libraries
library(data.table)
library(ggplot2)
library(ggmosaic)
library(readr)
library(readxl)

####loading and assign the data files to data.tables
filePath <- "intership/"
transactionData <- read_excel("intership/QVI_transaction_data.xlsx")
transactionData <- data.table(transactionData)
customerData <- fread(paste0(filePath,"QVI_purchase_behaviour.csv"))

#### Examine transaction data

str(transactionData)
head(transactionData)
view(transactionData)

#### Examine customer data
str(customerData)
head(customerData)
view(customerData)


# finding missing values using short sum()function.
sum(is.na(transactionData)) # result: 0
sum(is.na(customerData)) # result: 0

#### Convert DATE column to a date format
transactionData$DATE <- as.Date(transactionData$DATE, origin = "1899-12-30")


#### Examining PROD_NAME
summary(transactionData$PROD_NAME)
head(transactionData$PROD_NAME)
transactionData[, .N, PROD_NAME]

productWords <- data.table(unlist(strsplit(unique(transactionData[, PROD_NAME]), " ")))
setnames(productWords, 'words')

###let's remove all words with digits and special characters such as '&' from our set of product words
```

```
productWords <- productWords[grepl("\\d", words) == FALSE, ]

# Remove special characters
productWords <- productWords[grepl("[:alpha:]", words), ]

###counting the number of times a word appears and sorting first 10 highest.
head(productWords[, .N, words][order(N, decreasing = TRUE)], 10)


# remove the salsa product
transactionData[, SALSA := grepl("salsa", tolower(PROD_NAME))]
transactionData <- transactionData[SALSA == FALSE, ][, SALSA := NULL]

# summarizing the data
summary(transactionData)

# Filter the dataset to find the out-liers.
transactionData[PROD_QTY == 200, ]

#Let's see if the customer has had other transactions.
transactionData[LYLTY_CARD_NBR == 226000, ]


# Filter out the customer based on the loyalty card number
transactionData <- transactionData[LYLTY_CARD_NBR != 226000, ]

#### Re-examine transaction data
summary(transactionData)


#### Count the number of transactions by date
transactionData[, .N, by = DATE]      #[order(DATE)], sort if you need.



#### Create a sequence of dates and join this the count of transactions by date
allDates <- data.table(seq(as.Date("2018/07/01"), as.Date("2019/06/30"), by = "day"))
setnames(allDates, "DATE")
transactions_by_day <- merge(allDates, transactionData[, .N, by = DATE], all.x = TRUE)
### Setting plot themes to format graphs
theme_set(theme_bw())
theme_update(plot.title = element_text(hjust = 0.5))

# Plot transactions over time
```

```
ggplot(transactions_by_day, aes(x = DATE, y = N)) +
 geom_line(col = "orange") +
 labs(x = "Day", y = "Number of transactions", title = "Transactions over time") +
 scale_x_date(breaks = "1 month") +
 theme(axis.text.x = element_text(angle = 90, vjust = 0.5))


#### Filter to December and look at individual days
ggplot(transactions_by_day[month(DATE) == 12, ], aes(x = DATE, y = N)) +
 geom_line() +
 labs(x = "Day", y = "Number of transactions", title = "Transactions over time") +
 scale_x_date(breaks = "1 day") +
 theme(axis.text.x = element_text(angle = 90, vjust = 0.5))


#### We can work this out by taking the digits that are in PROD_NAME
transactionData[, PACK_SIZE := parse_number(PROD_NAME)]

# Let's check if the pack sizes look sensible
transactionData[, .N, PACK_SIZE][order(PACK_SIZE, decreasing = TRUE)]


#Plot a histogram showing the number of transactions by pack size.
options(scipen=999) # turn off scientific notations like 1e+05
hist(transactionData[, PACK_SIZE], col = "green",border = "red" , xlab = "PACK  SIZE", ylab = "Total no of chips purchased", main = "HISTOGRAM OF NO. OF CHIPS PURCHASED ACCORDING TO THEIR PACK SIZES")

#### Brands
transactionData[, BRAND := toupper(substr(PROD_NAME, 1, regexpr(pattern = ' ', PROD_NAME) - 1))]

# Checking brands
transactionData[, .N, by = BRAND][order(-N)]


#### Clean brand names
transactionData[BRAND == "RED", BRAND := "RRD"]
#other similar brands
transactionData[BRAND == "SNBTS", BRAND := "SUNBITES"]
transactionData[BRAND == "INFZNS", BRAND := "INFUZIONS"]
transactionData[BRAND == "WW", BRAND := "WOOLWORTHS"]
transactionData[BRAND == "SMITH", BRAND := "SMITHS"]
transactionData[BRAND == "NCC", BRAND := "NATURAL"]
transactionData[BRAND == "DORITO", BRAND := "DORITOS"]
```

```
transactionData[BRAND == "GRAIN", BRAND := "GRNWVES"]

# Checking
transactionData[, .N, by = BRAND][order(BRAND)]



#### Examining customer data
str(customerData)
head(customerData)

## Examining key values
customerData[, .N, by = LIFESTAGE][order(-N)]
customerData[, .N, by = PREMIUM_CUSTOMER][order(-N)]

#### Merge transaction data to customer data
data <- merge(transactionData, customerData, all.x = TRUE)

#Check for missing customer details
sum(is.null(data))  #use also; colSums(is.na(data)), it's perfect than that one.

# Save dataset as a csv.
write.csv(data,file="intership/QVI_data.csv")
```

```
#### Total sales by LIFESTAGE and PREMIUM_CUSTOMER
sales <- data[, .(SALES = sum(TOT_SALES)), .(LIFESTAGE, PREMIUM_CUSTOMER)]
# create plot
p <- ggplot(data = sales) +
  geom_mosaic(aes(weight = SALES, x = product(PREMIUM_CUSTOMER, LIFESTAGE) , fill =
PREMIUM_CUSTOMER)) +
  labs(x = "Lifestage", y = "Premium customer flag", title = "Proportion of sales") + theme(axis.text.x =
element_text(angle = 50, vjust = 0.5, size = 10))

# Plot and label with proportion of sales
p +
  geom_text(data = ggplot_build(p)$data[[1]], aes(x = (xmin + xmax)/2 , y = (ymin + ymax)/2, label =
as.character(paste(round(.wt/sum(.wt),3)*100, '%'))))


#### Number of customers by LIFESTAGE and PREMIUM_CUSTOMER
customers <- data[, .(CUSTOMERS = uniqueN(LYLTY_CARD_NBR)), .(LIFESTAGE,
PREMIUM_CUSTOMER)][order(-CUSTOMERS)]
labels <- c("A", "b", "c", "D", "e", "f", "g")
# Create plot
p <- ggplot(data = customers) + geom_mosaic(aes(weight = CUSTOMERS, x =
product(PREMIUM_CUSTOMER, LIFESTAGE), fill = PREMIUM_CUSTOMER)) + labs(x = "Lifestage", y =
"Premium customer flag", title = "Proportion of customers") + theme(axis.text.x = element_text(angle =
90, vjust = 0.5))+scale_x_productlist(labels = labels )

p + geom_text(data = ggplot_build(p)$data[[1]], aes(x = (xmin + xmax)/2 , y = (ymin + ymax)/2, label =
as.character(paste(round(.wt/sum(.wt),3)*100,'%'))))


#### Average number of units per customer by LIFESTAGE and PREMIUM_CUSTOMER
avg_units <- data[, .(AVG = sum(PROD_QTY)/uniqueN(LYLTY_CARD_NBR)), .(LIFESTAGE,
PREMIUM_CUSTOMER)][order(-AVG)]

ggplot(data = avg_units, aes(weight = AVG, x = LIFESTAGE, fill = PREMIUM_CUSTOMER)) +
geom_bar(position = position_dodge()) +
  labs(x = "Lifestage", y = "Avg units per transaction", title = "Units per customer") + theme(axis.text.x =
element_text(angle = 90, vjust = 0.75, size = 7))

#### Average price per unit by LIFESTAGE and PREMIUM_CUSTOMER
avg_price <- data[, .(AVG = sum(TOT_SALES)/sum(PROD_QTY)), .(LIFESTAGE,
PREMIUM_CUSTOMER)][order(-AVG)]
#### Create plot
```

```
ggplot(data = avg_price, aes(weight = AVG, x = LIFESTAGE, fill = PREMIUM_CUSTOMER)) +
geom_bar(position = position_dodge()) + labs(x = "Lifestage", y = "Avg price per unit", title = "Price per
unit") + theme(axis.text.x = element_text(angle = 90, vjust = 0.5))


#### young singles and couples
pricePerUnit <- data[, price := TOT_SALES/PROD_QTY]
t.test(data[LIFESTAGE %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES") &
PREMIUM_CUSTOMER == "Mainstream", price]
    , data[LIFESTAGE %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES") &
PREMIUM_CUSTOMER != "Mainstream", price]
    , alternative = "greater")
```

Answer for the t-test

```
Welch Two Sample t-test

data:  data[LIFESTAGE %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES
") & PREMIUM_CUSTOMER == "Mainstream", price] and data[LIFESTAGE %in% c("YOUN
G SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES") & PREMIUM_CUSTOMER != "Mainstre
am", price]
t = 37.624, df = 54791, p-value < 2.2e-16
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 0.3187234       Inf
sample estimates:
mean of x mean of y
 4.039786  3.706491
```

```
### Deeping dive into specific customer segments for insights
segment1 <- data[LIFESTAGE == "YOUNG SINGLES/COUPLES" & PREMIUM_CUSTOMER ==
"Mainstream",]
other <- data[!(LIFESTAGE == "YOUNG SINGLES/COUPLES" & PREMIUM_CUSTOMER ==
"Mainstream"),]

### Brand
quantity_segment1 <- segment1[, sum(PROD_QTY)]
quantity_other <- other[, sum(PROD_QTY)]
quantity_segment1_by_brand <- segment1[, .(targetSegment =
sum(PROD_QTY)/quantity_segment1), by = BRAND]
quantity_other_by_brand <- other[, .(other = sum(PROD_QTY)/quantity_other), by = BRAND]
brand_proportions <- merge(quantity_segment1_by_brand, quantity_other_by_brand)[,
affinityToBrand := targetSegment/other]
brand_proportions[order(-affinityToBrand)]
```

```
#### Deeping dive into Mainstream, young singles/couples
quantity_segment1_by_pack <- segment1[, .(targetSegment =
sum(PROD_QTY)/quantity_segment1), by = PACK_SIZE]
quantity_other_by_pack <- other[, .(other = sum(PROD_QTY)/quantity_other), by = PACK_SIZE]
pack_proportions <- merge(quantity_segment1_by_pack, quantity_other_by_pack)[,
affinityToPack := targetSegment/other]
pack_proportions[order(-affinityToPack)]

#### Preferred pack size compared to the rest of the population
quantity_segment1_by_pack <- segment1[, .(targetSegment =
sum(PROD_QTY)/quantity_segment1), by = PACK_SIZE]
quantity_other_by_pack <- other[, .(other = sum(PROD_QTY)/quantity_other), by = PACK_SIZE]
pack_proportions <- merge(quantity_segment1_by_pack, quantity_other_by_pack)[,
affinityToPack := targetSegment/other]

pack_proportions[order(-affinityToPack)]
```