```
#libraries
library(readxl)
library(tidyverse)
library(dplyr)
library(lubridate) # date format
library(stringr)
library(ggplot2)
library(tidytext)
getwd()
setwd(wd)
# Loading the data
transaction_data <-read_excel("TD.xlsx")

purchase_data <- read.csv(file.choose())

str(transaction_data)

# Checking for duplicated rows
transaction_data %>%
  filter(duplicated(transaction_data) == TRUE)

#Eliminating duplicated rows
transaction_data <- transaction_data %>%
  distinct()

# Preview 8 random rows to inspect how the table look like
transaction_data %>% sample_n(8)


# Inspecting purchasing data frame and variable types
str(purchase_data)

# Checking for duplicated rows
purchase_data %>%
  filter(duplicated(purchase_data) == TRUE)

# Preview 8 random rows to inspect how the table look like
purchase_data %>% sample_n(8)
# Copy transaction_data to tr_dt
tr_dt <- transaction_data


#DATE

# Convert to date format
tr_dt <- tr_dt %>%
  # Convert DATE to correct format
```

```
  mutate(DATE = as.Date(as.numeric(DATE), origin="1899-12-
30"),
         # Parse year
         year = year(DATE),
         # Parse month
         month = month(DATE, label = TRUE, abbr = FALSE),
         # Parse day
         day = mday(DATE),
         # Parse weekday
         weekday = wday(DATE, week_start = 1, label = TRUE,
abbr =  FALSE)) %>%
  # Relocate columns
  relocate(year, .after = DATE) %>%
  relocate(month, .after = year) %>%
  relocate(day, .after = month) %>%
  relocate(weekday, .after = day)

#Print first rows
head(tr_dt)

#Store Number

# Identify if there are NA rows
tr_dt %>%
  group_by(STORE_NBR) %>%
  count() %>%
  filter(is.na(STORE_NBR))

#Loyalty Card Number

# Checking if there are NA values
tr_dt %>%
  group_by(LYLTY_CARD_NBR) %>%
  count() %>%
  filter(is.na(LYLTY_CARD_NBR))

#  Transaction ID Number

# Identify if there are NA rows
tr_dt %>%
  group_by(TXN_ID) %>%
  count() %>%
  filter(is.na(TXN_ID))

#Product Variables

products <- tr_dt %>% select(PROD_NBR, PROD_NAME, PROD_QTY)
```

```
products
# Explore the structure of the product name, head
products %>%
  distinct(PROD_NAME) %>%
  arrange(PROD_NAME) %>%
  top_n(10)

# Extract the number that indicates the grams and create SIZE
column
tr_dt <- tr_dt %>%
  mutate(SIZE = (str_extract(PROD_NAME, "\\d+"))) %>%
  relocate(SIZE, .before = PROD_QTY)

# Check NAs
tr_dt %>%
  filter(is.na(SIZE))

# As numeric
tr_dt <- tr_dt %>%
  mutate(SIZE = as.numeric(SIZE))

products %>%
  distinct(PROD_NAME) %>%
  arrange(PROD_NAME) %>%
  head()

tr_dt <- tr_dt %>%
  mutate(
    BRAND = case_when(
      str_detect(PROD_NAME, "Burger") ~ "Burger",
      str_detect(PROD_NAME, "CCs") ~ "CCs",
      str_detect(PROD_NAME, "Cheetos") ~ "Cheetos",
      str_detect(PROD_NAME, "Cheezels") ~ "Cheezels",
      str_detect(PROD_NAME, "Cobs Popd") ~ "Cobs Popd",
      str_detect(PROD_NAME, "Doritos|Dorito") ~ "Doritos",
      str_detect(PROD_NAME, "French Fries") ~ "French Fries",
      str_detect(PROD_NAME, "Grain Waves|GrnWves") ~ "Grain
Waves",
      str_detect(PROD_NAME, "Infuzions|Infzns") ~ "Infuzions",
      str_detect(PROD_NAME, "Kettle") ~ "Kettle",
      str_detect(PROD_NAME, "Natural Chip|NCC") ~ "Natural
Chip Co",
      str_detect(PROD_NAME, "Old El Paso") ~ "Old el Paso",
      str_detect(PROD_NAME, "Pringles") ~ "Pringles",
      str_detect(PROD_NAME, "Red Rock Deli|RRD") ~ "Red Rock
Deli",
      str_detect(PROD_NAME, "Smiths|Smith") ~ "Smiths",
```

```
      str_detect(PROD_NAME, "Sunbites|Snbts") ~ "Sunbites",
      str_detect(PROD_NAME, "Thins") ~ "Thins",
      str_detect(PROD_NAME, "Tostitos") ~ "Tostitos",
      str_detect(PROD_NAME, "Twisties") ~ "Twisties",
      str_detect(PROD_NAME, "Tyrrells") ~ "Tyrrells",
      str_detect(PROD_NAME, "WW|Woolworths") ~ "Woolworths",
      TRUE ~ "Other" # If neither of the previous strings is
present
    ))


# Relocate BRAND column
tr_dt <- tr_dt %>%
  relocate(BRAND, .before = PROD_NAME)

#Inspecting random rows from the table to check things are
okay
tr_dt %>% sample_n(10)

# List of unique products that each BRAND offers
tr_dt %>%
  group_by(BRAND) %>%
  distinct(PROD_NAME) %>%
  arrange(BRAND) %>%
  head()

# Number of distinct products in each BRAND
tr_dt %>%
  group_by(BRAND) %>%
  summarise(distinct_PROD_NAME = n_distinct(PROD_NAME)) %>%
  arrange(desc(distinct_PROD_NAME)) %>%
  head()

tr_dt <- tr_dt %>%
  # Create PROD_CATEGORY column to classify in "Salsa Dip" or
"Chips"
  mutate(
    PROD_CATEGORY = case_when(
      str_detect(PROD_NAME, "Salsa|Salsa Dip") ~ "Salsa Dip",
      TRUE ~ "Chips"
    ))

# Relocate column
tr_dt <- tr_dt %>%
  relocate(PROD_CATEGORY, .after = BRAND)

# Check outliers.  Using BoxPlot to detect the presence of
```

```
outliers in the continuos variable:
boxplot(tr_dt[,c('TOT_SALES','PROD_NBR','SIZE','PROD_QTY')])

# Remove TOT_SALES outliers
tr_dt <- tr_dt %>%
  filter(TOT_SALES < 100)

#Inspecting random rows from the table to check things are
okay
tr_dt %>% sample_n(10)

# Copy purchase_data to pr_dt
pr_dt <- purchase_data

# Perform a left join
customer_purchase_data <- merge(x = tr_dt, y = pr_dt, by =
"LYLTY_CARD_NBR", all.x = T)

# Filter only Chips, as Salsa would not be relevant for the
stakeholder
chips_data <- customer_purchase_data %>%
  filter(PROD_CATEGORY == "Chips")
head(chips_data)


# Customer types distribution
customer_distribution <- chips_data %>%
  # Select distinct clients (as the same client might have
more than 1 transaction). This gives 71,287 unique rows or
clients
  mutate(LYLTY_CARD_NBR = as.character(LYLTY_CARD_NBR)) %>%
  distinct(LYLTY_CARD_NBR, .keep_all = TRUE) %>%
  # Group by loyalty plan and lifestage, count observations
  group_by(PREMIUM_CUSTOMER, LIFESTAGE) %>%
  count()
customer_distribution


ggplot(customer_distribution, aes(x = PREMIUM_CUSTOMER, y = n,
fill = PREMIUM_CUSTOMER)) +
  geom_bar(stat="identity") +
  facet_wrap( ~ LIFESTAGE)


# Summary of total sales grouped by customer type
chips_data %>%
  group_by(PREMIUM_CUSTOMER) %>%
```

```
  summarise(avg_ticket_sales = round(mean(TOT_SALES),2),
            avg_ticket_product_quantity =
round(mean(PROD_QTY),2),
            avg_ticket_size_g = round(mean(SIZE),2))



# Sales histogram per customer type
sales_customer_gg <- ggplot(data = chips_data, aes(x =
TOT_SALES)) +
  geom_histogram(binwidth =
2,aes(fill=PREMIUM_CUSTOMER),color="black") +
  facet_grid(~PREMIUM_CUSTOMER)
sales_customer_gg

# Calculate mean TOT_SALES for each PREMIUM_CUSTOMER group, to
print in the plot
mean_sales <- chips_data %>%
  group_by(PREMIUM_CUSTOMER) %>%
  summarise(mean_sales = mean(TOT_SALES, na.rm = TRUE))

# Plot the histogram with mean lines
sales_customer_gg +
  geom_vline(data = mean_sales, aes(xintercept = mean_sales),
color = "grey", linetype = "dashed", size = 1) +
  geom_text(data = mean_sales, aes(x = mean_sales, label =
sprintf("%.2f", mean_sales), y = 0.5), vjust = -0.5, color =
"white")  +
  labs(x = "TOT_SALES", y = "Frequency") +
  theme_minimal()

# Repeated customers (recurrent purchases)
repeated <- chips_data %>%
  # Group observations by customer type and number of loyalty
card
  group_by(PREMIUM_CUSTOMER, LIFESTAGE, LYLTY_CARD_NBR) %>%
  # Count rows in each group and order by descendent order
  summarise(counts = n()) %>%
  arrange(desc(counts))



# Number of visits per customer type
visits_customers <- repeated %>%
  group_by(PREMIUM_CUSTOMER, LIFESTAGE) %>%
  summarize(min_visits = min(counts),
            max_visits = max(counts),
            mean_visits = round(mean(counts),2))
```

```
# Number of visits per lifestage
ggplot(data = repeated, aes(x = counts)) +
  geom_histogram(binwidth = 1, aes(fill=LIFESTAGE),
color="black") +
  facet_wrap(~LIFESTAGE, scale="free")

# Summarize the data to get counts for each brand
brand_counts <- chips_data %>%
  group_by(PREMIUM_CUSTOMER, LIFESTAGE) %>%
  count(BRAND) %>%
  arrange(desc(n))

# Plot brand counts per customer type
ggplot(data=brand_counts, aes(y = reorder_within(BRAND, n,
PREMIUM_CUSTOMER),
                              x = n, fill = BRAND)) +
  geom_bar(stat = "identity") +
  labs(y = "Brand", x = "Count") +
  scale_y_reordered() +
  facet_wrap(~ PREMIUM_CUSTOMER, scales = "free_y")

# Plot brand counts per lifestage
ggplot(data=brand_counts, aes(y = reorder_within(BRAND, n,
LIFESTAGE), x = n, fill = BRAND)) +
  geom_bar(stat = "identity") +
  labs(y = "Brand", x = "Count") +
  scale_y_reordered() +
  facet_wrap(~LIFESTAGE,nrow=6, scales = "free_y")

# Sales trends
ggplot(data = chips_data, aes(x = DATE)) +
  geom_line(stat = "count")

# Brand sales trends
qty_brand_trend <- chips_data %>%
  group_by(DATE, BRAND) %>%
  summarize(products_quanty = sum(PROD_QTY)) %>%
  # Creating a column to group Year and Month only
  mutate(yy_mmm = format(as.Date(DATE), "%Y-%m"))

ggplot(qty_brand_trend, aes(x = DATE, y = products_quanty,
group = BRAND, color = BRAND)) +
  geom_line() +
  labs(x = "Date", y = "Total Product Quantity") +
  theme_minimal()

brand_counts_year <- chips_data %>%
```

```
  group_by(year, BRAND) %>%
  count()

# Plot brand counts per year
ggplot(brand_counts_year, aes(y = reorder_within(BRAND, n,
year), x = n, fill = BRAND)) +
  geom_bar(stat = "identity") +
  labs(y = "Brand", x = "Count") +
  scale_y_reordered() +
  facet_wrap(~ year, nrow = 2, scales = "free_y")




# Weekly products purchase and lifestage
lifestage_prod <- chips_data %>%
  group_by(LIFESTAGE, weekday, BRAND) %>%
  summarise(product_sum = sum(PROD_QTY),
            avg_sales = mean(TOT_SALES),
            avg_size = mean(SIZE))

# Weekdays : product sum
ggplot(data = lifestage_prod, aes(x = weekday,
                                  y = product_sum)) +
  geom_bar(stat = "identity") +
  labs(x = "Weekday", y = "Product Sum") +
  facet_wrap(~ LIFESTAGE)

# Weekdays: sales
ggplot(data = lifestage_prod, aes(x = weekday, y = avg_sales))
+
  geom_bar(stat = "identity") +
  labs(x = "Weekday", y = "Avg sales") +
  facet_wrap(~ LIFESTAGE)

# Weekdays: Products
ggplot(data = lifestage_prod, aes(x = weekday, y =
product_sum)) +
  geom_bar(stat = "identity") +
  labs(x = "Weekday", y = "Product Sum") +
  facet_wrap(~ LIFESTAGE)
```