```r
# --- 1. SETUP ---

# Load required libraries
# install.packages(c("data.table", "ggplot2", "ggmosaic", "readr", "dplyr"))
library(data.table)
library(ggplot2)
library(ggmosaic)
library(readr)
library(dplyr)

# Set the file path to where you have downloaded the data sets
# USER ACTION: Please update this path to your local directory.
filePath <- "D:/Quantium/Task_1/"

# --- 2. DATA LOADING & PREPARATION ---

# Load the datasets
transactionData <- fread(paste0(filePath, "QVI_transaction_data.csv"))
customerData <- fread(paste0(filePath, "QVI_purchase_behaviour.csv"))

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

# Examine PROD_NAME
# transactionData[, .N, PROD_NAME]

# --- Data Cleaning ---

# Create a word frequency table from product names to identify non-chip products
productWords <- data.table(unlist(strsplit(unique(transactionData[, PROD_NAME]), " ")))
setnames(productWords, 'words')

# Remove words with digits and special characters
productWords <- productWords[grepl("\\d", words) == FALSE, ]
productWords <- productWords[grepl("[:alpha:]", words), ]

# Show the most frequent words to identify potential data cleaning targets
# productWords[, .N, words][order(N, decreasing = TRUE)]

# Remove salsa products from the transaction data, as identified from the word frequency list
transactionData <- transactionData[!grepl("salsa", tolower(PROD_NAME)), ]

# Summarise the data to check for nulls and possible outliers
summary(transactionData)

# Identify and investigate the outlier in PROD_QTY
# transactionData[PROD_QTY == 200, ]
# transactionData[LYLTY_CARD_NBR == 226000, ]

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

# Re-examine transaction data after outlier removal
summary(transactionData)

# Create a complete sequence of dates from July 2018 to June 2019
allDates <- data.table(DATE = seq(as.Date("2018-07-01"), as.Date("2019-06-30"), by = "day"))

# Count transactions per day and merge with the complete date sequence
transactions_by_day <- merge(allDates, transactionData[, .N, by = DATE], all.x = TRUE)
# Replace NA values (days with no transactions) with 0
transactions_by_day[is.na(N), N := 0]
```

```
# --- 3. EXPLORATORY DATA ANALYSIS (EDA) ---

# Set a consistent theme for all plots
theme_set(theme_bw())
theme_update(plot.title = element_text(hjust = 0.5))

# Plot transactions over time to observe trends
ggplot(transactions_by_day, aes(x = DATE, y = N)) +
  geom_line() +
  labs(x = "Day", y = "Number of Transactions", title = "Transactions Over Time") +
  scale_x_date(date_breaks = "1 month", date_labels = "%B %Y") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# Zoom in on December to investigate the dip in sales
ggplot(transactions_by_day[month(DATE) == 12, ], aes(x = DATE, y = N)) +
  geom_line() +
  labs(x = "Day in December", y = "Number of Transactions", title = "Transactions in December
2018") +
  scale_x_date(date_breaks = "1 day", date_labels = "%d") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
# The plot shows a dip on Christmas Day, which is expected.

# --- Feature Engineering ---

# Extract pack size from product name
transactionData[, PACK_SIZE := parse_number(PROD_NAME)]

# Plot a histogram of pack sizes to see the distribution
hist(transactionData[, PACK_SIZE], main = "Distribution of Pack Sizes", xlab = "Pack Size (g)")

# Extract brand name from product name
transactionData[, BRAND := toupper(substr(PROD_NAME, 1, regexpr(pattern = ' ', PROD_NAME) - 1))]

# Clean brand names by correcting inconsistencies
transactionData[BRAND == "RED", BRAND := "RRD"]
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"]

# Check the cleaned brand names
# transactionData[, .N, by = BRAND][order(BRAND)]


# --- 4. CUSTOMER SEGMENTATION ANALYSIS ---

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

# Remove transactions with no matching customer data
data <- data[!is.na(LIFESTAGE) & !is.na(PREMIUM_CUSTOMER)]

# Calculate total sales by LIFESTAGE and PREMIUM_CUSTOMER
sales <- data[, .(SALES = sum(TOT_SALES)), .(LIFESTAGE, PREMIUM_CUSTOMER)]

# Plot proportion of sales using a mosaic plot
p_sales <- ggplot(data = sales) +
  geom_mosaic(aes(weight = SALES, x = product(PREMIUM_CUSTOMER, LIFESTAGE), fill =
PREMIUM_CUSTOMER)) +
  labs(x = "Lifestage", y = "Premium Customer", title = "Proportion of Sales by Customer Segment")
+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```

```
# Add percentage labels to the plot
p_sales + geom_text(data = ggplot_build(p_sales)$data[[1]], aes(x = (xmin + xmax) / 2, y = (ymin +
ymax) / 2, label = as.character(paste(round(.wt / sum(.wt), 3) * 100, '%'))))


# Calculate number of customers by LIFESTAGE and PREMIUM_CUSTOMER
customers <- data[, .(CUSTOMERS = uniqueN(LYLTY_CARD_NBR)), .(LIFESTAGE, PREMIUM_CUSTOMER)]
[order(-CUSTOMERS)]

# Plot proportion of customers using a mosaic plot
p_cust <- ggplot(data = customers) +
  geom_mosaic(aes(weight = CUSTOMERS, x = product(PREMIUM_CUSTOMER, LIFESTAGE), fill =
PREMIUM_CUSTOMER)) +
  labs(x = "Lifestage", y = "Premium Customer", title = "Proportion of Customers by Segment") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))

# Add percentage labels to the plot
p_cust + geom_text(data = ggplot_build(p_cust)$data[[1]], aes(x = (xmin + xmax) / 2, y = (ymin +
ymax) / 2, label = as.character(paste(round(.wt / sum(.wt), 3) * 100, '%'))))


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

# Plot average units per customer
ggplot(data = avg_units, aes(x = LIFESTAGE, y = AVG_UNITS, fill = PREMIUM_CUSTOMER)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Lifestage", y = "Avg Units per Customer", title = "Average Units per Customer by
Segment") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))


# Calculate average price per unit by LIFESTAGE and PREMIUM_CUSTOMER
avg_price <- data[, .(AVG_PRICE = sum(TOT_SALES) / sum(PROD_QTY)), .(LIFESTAGE, PREMIUM_CUSTOMER)]
[order(-AVG_PRICE)]

# Plot average price per unit
ggplot(data = avg_price, aes(x = LIFESTAGE, y = AVG_PRICE, fill = PREMIUM_CUSTOMER)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Lifestage", y = "Avg Price per Unit", title = "Average Price per Unit by Segment") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))


# --- 5. DEEP DIVE ANALYSIS ---

# Perform an independent t-test to see if the price per unit is significantly different
# between Mainstream vs Premium/Budget customers in the "YOUNG SINGLES/COUPLES" and "MIDAGE
SINGLES/COUPLES" lifestages.
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")
# The t-test results (p-value < 2.2e-16) show a statistically significant difference.

# Isolate the target segment: "Mainstream, young singles/couples"
segment1 <- data[LIFESTAGE == "YOUNG SINGLES/COUPLES" & PREMIUM_CUSTOMER == "Mainstream",]
other <- data[!(LIFESTAGE == "YOUNG SINGLES/COUPLES" & PREMIUM_CUSTOMER == "Mainstream"),]

# --- Brand Affinity Analysis ---

# Calculate brand affinity for the target segment compared to the rest of the customers
```

```r
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_sorted <- brand_proportions[order(-affinityToBrand)]

# Visualize the top 5 brands by affinity for the target segment
ggplot(head(brand_proportions_sorted, 5), aes(x = reorder(BRAND, affinityToBrand), y =
affinityToBrand)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  coord_flip() +
  labs(title = "Top 5 Brand Affinities for Mainstream Young Singles/Couples",
       x = "Brand",
       y = "Affinity Score") +
  theme(plot.title = element_text(hjust = 0.5))


# --- Pack Size Analysis ---

# Calculate pack size preference for the target segment
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_sorted <- pack_proportions[order(-affinityToPack)]

# Visualize pack size affinity
ggplot(pack_proportions_sorted, aes(x = PACK_SIZE, y = affinityToPack)) +
  geom_bar(stat = "identity", fill = "coral") +
  labs(title = "Pack Size Affinity for Mainstream Young Singles/Couples",
       x = "Pack Size (g)",
       y = "Affinity Score") +
  theme(plot.title = element_text(hjust = 0.5))

# The plot shows that this segment has a strong preference for the 270g pack size.
# Let's see what products come in 270g packs.
# data[PACK_SIZE == 270, unique(PROD_NAME)]
# Output: "Twisties | Cheese | 270g", "Twisties | Chicken | 270g"

# --- END OF ANALYSIS ---
```