# Quantium Virtual Internship - Retail Strategy and Analytics - Task 1

Giang Truong

## Task 1

### Load required libraries and datasets

Note that you will need to install these libraries if you have never used these before.

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

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:data.table':
##
##     between, first, last

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
#### Point the filePath to where you have downloaded the datasets to and
#### assign the data files to data.tables
setwd("C:/Giang/Studying/Virtual Internship- Quantium")
transactionData <- read_excel("QVI_transaction_data.xlsx")
customerData <- read.csv("QVI_purchase_behaviour.csv")
```

### Exploratory data analysis

The first step in any analysis is to first understand the data. Let's take a look at each of the datasets provided. ### Examining transaction data We can use `str()` to look at the format of each column and see a sample of the data. As we have read in the dataset as a `data.table` object, we can also run `transactionData` in the console to see a sample of the data or use `head(transactionData)` to look at the first 10 rows. Let's check if columns we would expect to be numeric are in numeric form and date columns are in date format.

```
#### Examine transaction data
str(transactionData)
```

```
## tibble [264,836 x 8] (S3: tbl_df/tbl/data.frame)
## $ DATE : num [1:264836] 43390 43599 43605 43329 43330 ...
## $ STORE_NBR : num [1:264836] 1 1 1 2 2 4 4 4 5 7 ...
## $ LYLTY_CARD_NBR: num [1:264836] 1000 1307 1343 2373 2426 ...
## $ TXN_ID : num [1:264836] 1 348 383 974 1038 ...
## $ PROD_NBR : num [1:264836] 5 66 61 69 108 57 16 24 42 52 ...
## $ PROD_NAME : chr [1:264836] "Natural Chip Compny SeaSalt175g" "CCs Nacho
Cheese 175g" "Smiths Crinkle Cut Chips Chicken 170g" "Smiths Chip Thinly
S/Cream&Onion 175g" ...
## $ PROD_QTY : num [1:264836] 2 3 2 5 3 1 1 1 1 2 ...
## $ TOT_SALES : num [1:264836] 6 6.3 2.9 15 13.8 5.1 5.7 3.6 3.9 7.2 ...
```

```
transactionData <- data.table(transactionData)
```

We can see that the date column is in an integer format. Let's change this to a date format.

```
#### Convert DATE column to a date format
#### A quick search online tells us that CSV and Excel integer dates begin on 30 Dec 1899
transactionData$DATE <- as.Date(transactionData$DATE, origin = "1899-12-30")
```

We should check that we are looking at the right products by examining PROD_NAME.

```
#### Examine PROD_NAME
head(transactionData$PROD_NAME)
```

```
## [1] "Natural Chip        Compny SeaSalt175g"
## [2] "CCs Nacho Cheese    175g"
## [3] "Smiths Crinkle Cut  Chips Chicken 170g"
## [4] "Smiths Chip Thinly  S/Cream&Onion 175g"
## [5] "Kettle Tortilla ChpsHny&Jlpno Chili 150g"
## [6] "Old El Paso Salsa   Dip Tomato Mild 300g"
```

```
str(transactionData$PROD_NAME)
```

```
##  chr [1:264836] "Natural Chip        Compny SeaSalt175g" ...
```

```
summary(transactionData$PROD_NAME)
```

```
##    Length     Class      Mode
##    264836 character character
```

Looks like we are definitely looking at potato chips but how can we check that these are all chips? We can do some basic text analysis by summarising the individual words in the product name.

```r
#### Examine the words in PROD_NAME to see if there are any incorrect entries
#### such as products that are not chips
productWords <- data.table(unlist(strsplit(unique(transactionData$PROD_NAME), " ")))
setnames(productWords, 'words')
```

As we are only interested in words that will tell us if the product is chips or not, let's remove all words with digits and special characters such as '&' from our set of product words. We can do this using `grepl()`.

```r
#### Removing digits
productWords <- productWords[grepl("\\d",words)== FALSE, ]
#### Removing special characters
productWords <- productWords[grepl("[:alpha:]",words), ]
#### Let's look at the most common words by counting the number of times a word appears and
#### sorting them by this frequency in order of highest to lowest frequency
productWords[, .N, words][order(N, decreasing = TRUE)]
```

```
##             words  N
##   1:        Chips 21
##   2:       Smiths 16
##   3:      Crinkle 14
##   4:       Kettle 13
##   5:       Cheese 12
##  ---
## 127: Chikn&Garlic  1
## 128:        Aioli  1
## 129:         Slow  1
## 130:        Belly  1
## 131:    Bolognese  1
```

There are salsa products in the dataset but we are only interested in the chips category, so let's remove these.

```r
#### Remove salsa products
transactionData[, SALSA := grepl("salsa", tolower(PROD_NAME))]
transactionData <- transactionData[SALSA == FALSE, ][, SALSA := NULL]
```

Next, we can use `summary()` to check summary statistics such as mean, min and max values for each feature to see if there are any obvious outliers in the data and if there are any nulls in any of the columns (`NA's : number of nulls` will appear in the output if there are any nulls).

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

```
##       DATE              STORE_NBR     LYLTY_CARD_NBR        TXN_ID
##  Min.   :2018-07-01   Min.   :  1.0   Min.   :   1000   Min.   :      1
##  1st Qu.:2018-09-30   1st Qu.: 70.0   1st Qu.:  70015   1st Qu.:  67569
##  Median :2018-12-30   Median :130.0   Median : 130367   Median : 135183
##  Mean   :2018-12-30   Mean   :135.1   Mean   : 135531   Mean   : 135131
##  3rd Qu.:2019-03-31   3rd Qu.:203.0   3rd Qu.: 203084   3rd Qu.: 202654
##  Max.   :2019-06-30   Max.   :272.0   Max.   :2373711   Max.   :2415841
##     PROD_NBR       PROD_NAME            PROD_QTY         TOT_SALES
```

```
##  Min.   :  1.00   Length:246742      Min.   :  1.000   Min.   :  1.700
##  1st Qu.: 26.00   Class :character   1st Qu.:  2.000   1st Qu.:  5.800
##  Median : 53.00   Mode  :character   Median :  2.000   Median :  7.400
##  Mean   : 56.35                      Mean   :  1.908   Mean   :  7.321
##  3rd Qu.: 87.00                      3rd Qu.:  2.000   3rd Qu.:  8.800
##  Max.   :114.00                      Max.   :200.000   Max.   :650.000
```

There are no nulls in the columns but product quantity appears to have an outlier which we should investigate further. Let's investigate further the case where 200 packets of chips are bought in one transaction.

```r
#### Filter the dataset to find the outlier
transactionData[PROD_QTY == 200, ]
```

```
##          DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
## 1: 2018-08-19       226         226000 226201        4
## 2: 2019-05-20       226         226000 226210        4
##                             PROD_NAME PROD_QTY TOT_SALES
## 1: Dorito Corn Chp     Supreme 380g      200       650
## 2: Dorito Corn Chp     Supreme 380g      200       650
```

There are two transactions where 200 packets of chips are bought in one transaction and both of these transactions were by the same customer.

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

```
##          DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
## 1: 2018-08-19       226         226000 226201        4
## 2: 2019-05-20       226         226000 226210        4
##                             PROD_NAME PROD_QTY TOT_SALES
## 1: Dorito Corn Chp     Supreme 380g      200       650
## 2: Dorito Corn Chp     Supreme 380g      200       650
```

It looks like this customer has only had the two transactions over the year and is not an ordinary retail customer. The customer might be buying chips for commercial purposes instead. We'll remove this loyalty card number from further analysis.

```r
#### Filter out the customer based on the loyalty card number
transactionData <- transactionData[LYLTY_CARD_NBR != 226000, ]
#### Re-examine transaction data
summary(transactionData)
```

```
##       DATE              STORE_NBR     LYLTY_CARD_NBR        TXN_ID
##  Min.   :2018-07-01   Min.   :  1.0   Min.   :   1000   Min.   :      1
##  1st Qu.:2018-09-30   1st Qu.: 70.0   1st Qu.:  70015   1st Qu.:  67569
##  Median :2018-12-30   Median :130.0   Median : 130367   Median : 135182
##  Mean   :2018-12-30   Mean   :135.1   Mean   : 135530   Mean   : 135130
##  3rd Qu.:2019-03-31   3rd Qu.:203.0   3rd Qu.: 203083   3rd Qu.: 202652
##  Max.   :2019-06-30   Max.   :272.0   Max.   :2373711   Max.   :2415841
##     PROD_NBR       PROD_NAME            PROD_QTY        TOT_SALES
##  Min.   :  1.00   Length:246740      Min.   :1.000   Min.   : 1.700
##  1st Qu.: 26.00   Class :character   1st Qu.:2.000   1st Qu.: 5.800
```

```
##  Median : 53.00   Mode  :character   Median :2.000   Median : 7.400
##  Mean   : 56.35                     Mean   :1.906   Mean   : 7.316
##  3rd Qu.: 87.00                     3rd Qu.:2.000   3rd Qu.: 8.800
##  Max.   :114.00                     Max.   :5.000   Max.   :29.500
```

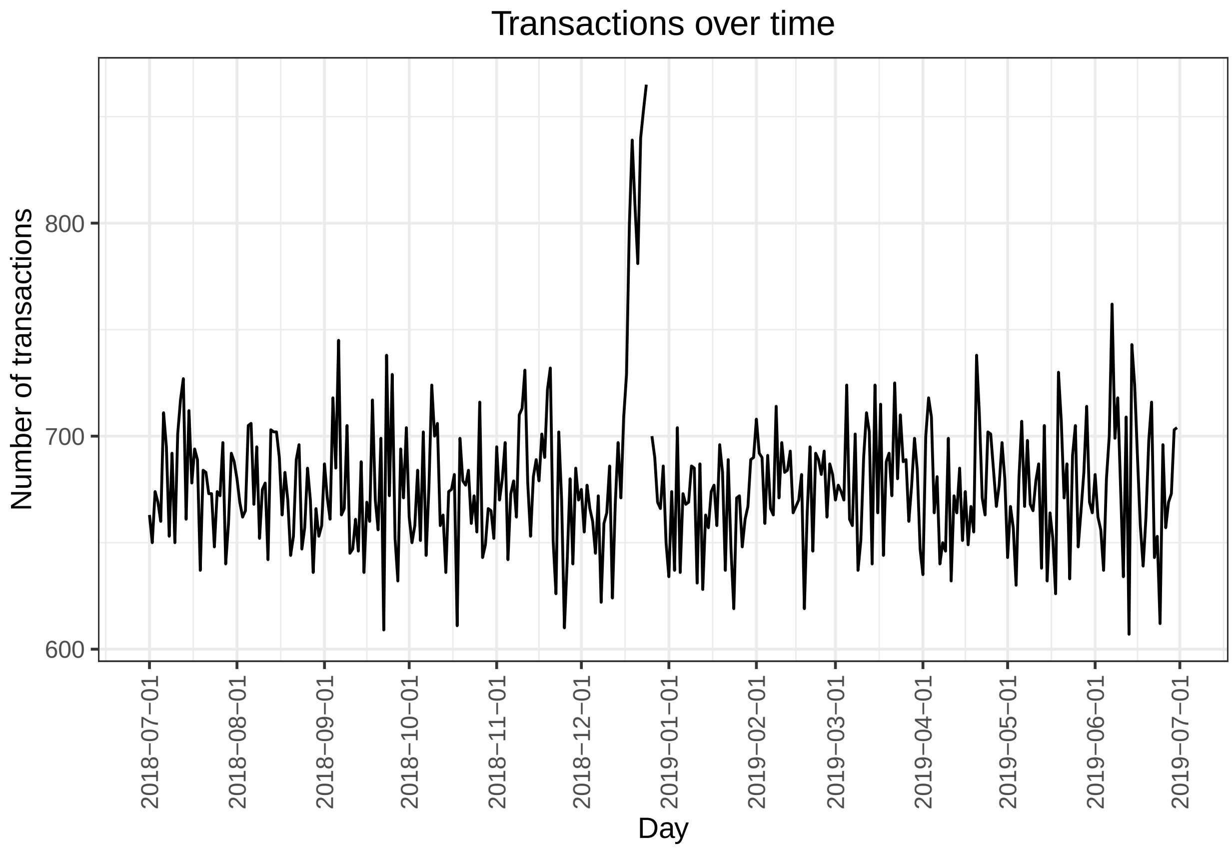
That's better. Now, let's look at the number of transaction lines over time to see if there are any obvious data issues such as missing data.

```r
#### Count the number of transactions by date
transactionData[, .(Count = .N), by = .(DATE)]
```

```
##            DATE Count
##   1: 2018-10-17   682
##   2: 2019-05-14   705
##   3: 2019-05-20   707
##   4: 2018-08-17   663
##   5: 2018-08-18   683
##  ---
## 360: 2018-12-08   622
## 361: 2019-01-30   689
## 362: 2019-02-09   671
## 363: 2018-08-31   658
## 364: 2019-02-12   684
```

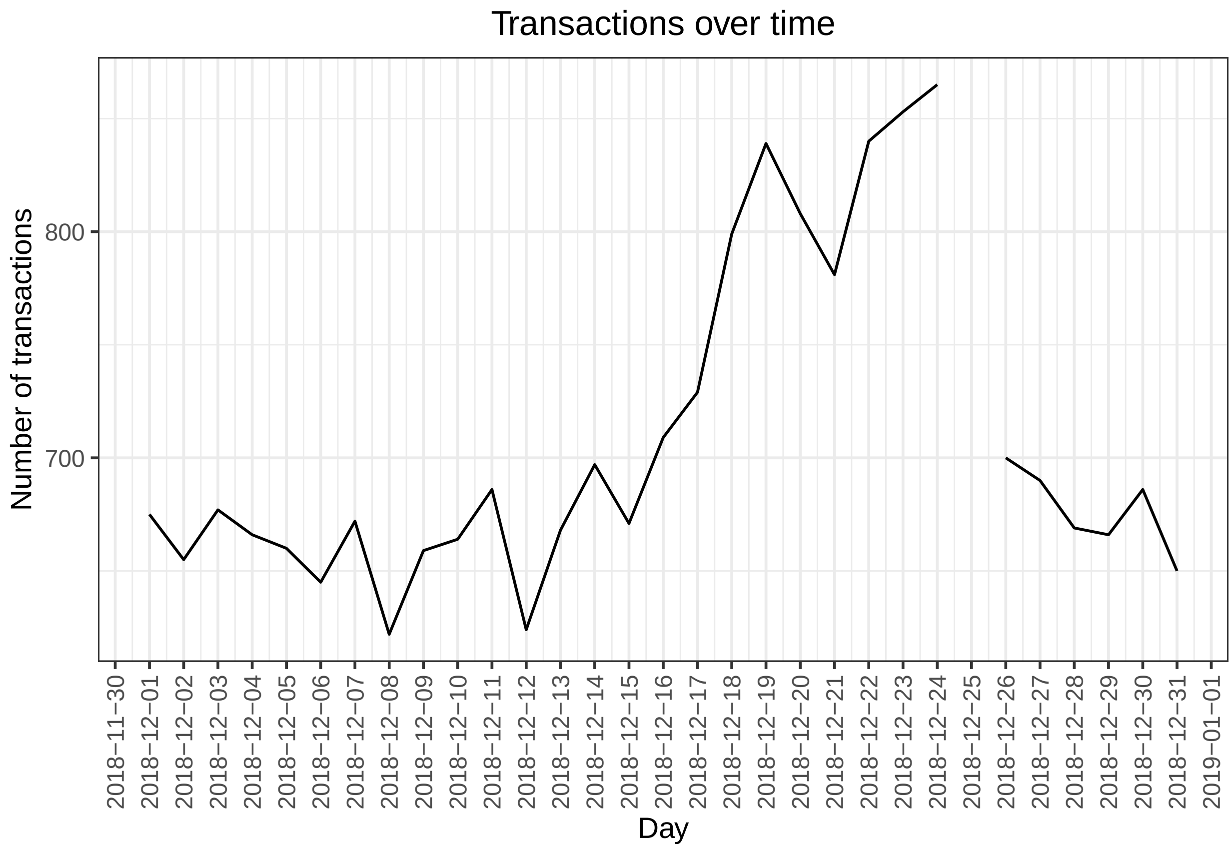
There's only 364 rows, meaning only 364 dates which indicates a missing date. Let's create a sequence of dates from 1 Jul 2018 to 30 Jun 2019 and use this to create a chart of number of transactions over time to find the missing date.

```r
#### Create a sequence of dates and join this the count of transactions by date

total_dates <- seq(as.Date("2018-07-01"), as.Date("2019-06-30"), by = "day")

# Join the date sequence with the transaction data to get the transaction counts
transactions_by_day <- transactionData[, .(Transaction_count = .N), by = .(DATE)]
transactions_by_day <- merge(data.table(DATE = total_dates), transactions_by_day, 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 = Transaction_count)) +
 geom_line() +
 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))
```




We can see that there is an increase in purchases in December and a break in late December. Let's zoom in on this.

```
#### Filter to December and look at individual days
ggplot(transactions_by_day[month(DATE)==12], aes(x = DATE, y = Transaction_count)) +
 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 see that the increase in sales occurs in the lead-up to Christmas and that there are zero sales on Christmas day itself. This is due to shops being closed on Christmas day. Now that we are satisfied that the data no longer has outliers, we can move on to creating other features such as brand of chips or pack size from PROD_NAME. We will start with pack size.

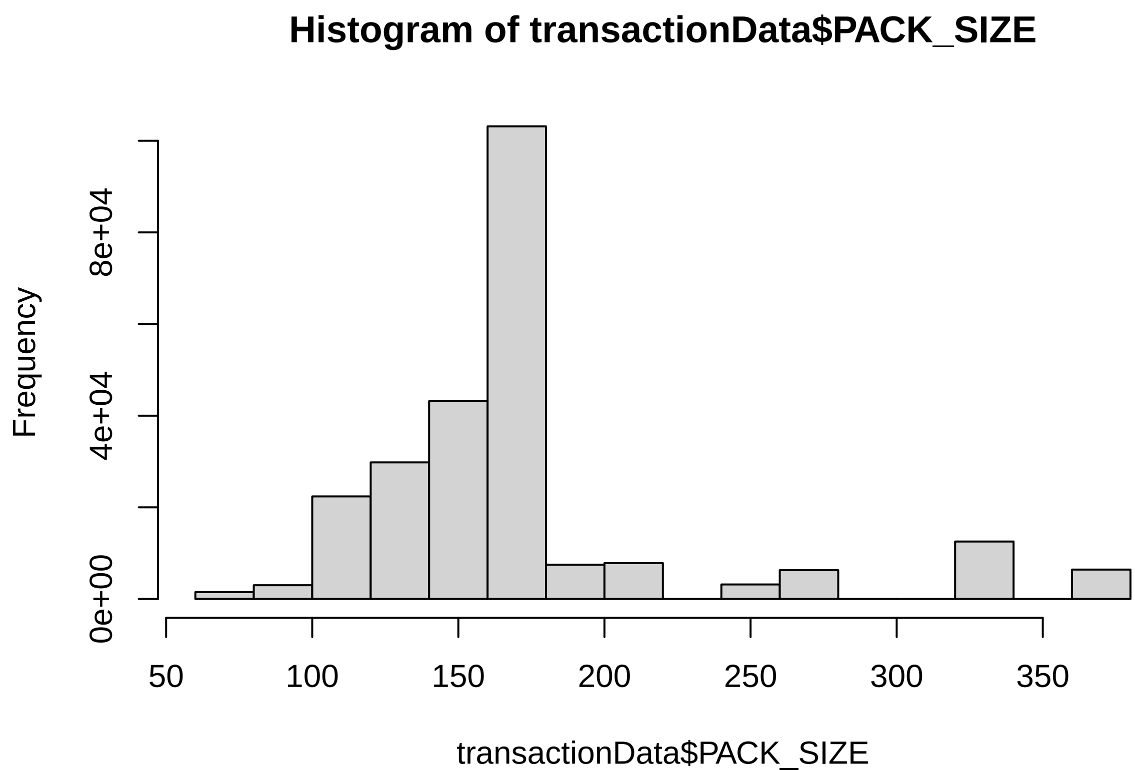
```
#### Pack size
#### We can work this out by taking the digits that are in PROD_NAME
transactionData[, PACK_SIZE := parse_number(PROD_NAME)]
#### Always check your output
#### Let's check if the pack sizes look sensible
transactionData[, .N, PACK_SIZE][order(PACK_SIZE)]
```

```
##     PACK_SIZE     N
##  1:        70  1507
##  2:        90  3008
##  3:       110 22387
##  4:       125  1454
##  5:       134 25102
##  6:       135  3257
##  7:       150 40203
##  8:       160  2970
##  9:       165 15297
## 10:       170 19983
## 11:       175 66390
## 12:       180  1468
```

```
## 13:        190  2995
## 14:        200  4473
## 15:        210  6272
## 16:        220  1564
## 17:        250  3169
## 18:        270  6285
## 19:        330 12540
## 20:        380  6416
```

The largest size is 380g and the smallest size is 70g - seems sensible!

```r
#### Let's plot a histogram of PACK_SIZE since we know that it is a categorical variable and not a cont
hist(transactionData$PACK_SIZE)
```




Pack sizes created look reasonable. Now to create brands, we can use the first word in PROD_NAME to work out the brand name…

```r
#### Brands
# Create a new column called "brand"
transactionData$BRAND <- NA

# Use regular expressions to extract the first word from the PROD_NAME column and assign it to the bran

transactionData$BRAND[grep("^\\w+", transactionData$PROD_NAME)] <-
  gsub("(^\\w+).*", "\\1", transactionData$PROD_NAME[grep("^\\w+", transactionData$PROD_NAME)])
```

```r
# Note: grep("^\\w+", transactionData$PROD_NAME) searches the PROD_NAME column for strings that begin w
# gsub("(^\\w+).*", "\\1", transactionData$PROD_NAME[grep("^\\w+", transactionData$PROD_NAME)]) extract

# Print the first 10 rows of the updated transactionData table
head(transactionData, 10)
```

```
##           DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
##  1: 2018-10-17         1           1000      1        5
##  2: 2019-05-14         1           1307    348       66
##  3: 2019-05-20         1           1343    383       61
##  4: 2018-08-17         2           2373    974       69
##  5: 2018-08-18         2           2426   1038      108
##  6: 2019-05-16         4           4149   3333       16
##  7: 2019-05-16         4           4196   3539       24
##  8: 2018-08-20         5           5026   4525       42
##  9: 2018-08-18         7           7150   6900       52
## 10: 2019-05-17         7           7215   7176       16
##                                    PROD_NAME PROD_QTY TOT_SALES PACK_SIZE
##  1:   Natural Chip        Compny SeaSalt175g        2       6.0       175
##  2:                 CCs Nacho Cheese    175g        3       6.3       175
##  3:   Smiths Crinkle Cut  Chips Chicken 170g        2       2.9       170
##  4:   Smiths Chip Thinly  S/Cream&Onion 175g        5      15.0       175
##  5: Kettle Tortilla ChpsHny&Jlpno Chili 150g        3      13.8       150
##  6: Smiths Crinkle Chips Salt & Vinegar 330g        1       5.7       330
##  7:    Grain Waves         Sweet Chilli 210g        1       3.6       210
##  8:  Doritos Corn Chip Mexican Jalapeno 150g        1       3.9       150
##  9:    Grain Waves Sour    Cream&Chives 210G        2       7.2       210
## 10: Smiths Crinkle Chips Salt & Vinegar 330g        1       5.7       330
##       BRAND
##  1: Natural
##  2:     CCs
##  3:  Smiths
##  4:  Smiths
##  5:  Kettle
##  6:  Smiths
##  7:   Grain
##  8: Doritos
##  9:   Grain
## 10:  Smiths
```

```r
#### Checking brands
transactionData[, .N, BRAND][order(-N)]
```

```
##          BRAND     N
##  1:     Kettle 41288
##  2:     Smiths 27390
##  3:   Pringles 25102
##  4:    Doritos 22041
##  5:      Thins 14075
##  6:        RRD 11894
##  7:  Infuzions 11057
##  8:         WW 10320
##  9:       Cobs  9693
```

```
## 10:   Tostitos  9471
## 11:   Twisties  9454
## 12:   Tyrrells  6442
## 13:      Grain  6272
## 14:    Natural  6050
## 15:   Cheezels  4603
## 16:        CCs  4551
## 17:        Red  4427
## 18:     Dorito  3183
## 19:     Infzns  3144
## 20:      Smith  2963
## 21:    Cheetos  2927
## 22:      Snbts  1576
## 23:     Burger  1564
## 24: Woolworths  1516
## 25:    GrnWves  1468
## 26:   Sunbites  1432
## 27:        NCC  1419
## 28:     French  1418
##           BRAND     N
```

Some of the brand names look like they are of the same brands - such as RED and RRD, which are both Red Rock Deli chips. Let's combine these together.

```
#### Clean brand names
transactionData[BRAND == "RED", BRAND := "RRD"]
transactionData[BRAND == "Dorito", BRAND := "Doritos"]
transactionData[BRAND == "WW", BRAND := "Woolworths"]
transactionData[BRAND == "NCC", BRAND := "Natural"]
transactionData[BRAND == "Infzns", BRAND := "Infuzions"]
transactionData[BRAND == "Smith", BRAND := "Smiths"]
transactionData[BRAND == "Snbts", BRAND := "Sunbites"]
transactionData[BRAND == "Grain", BRAND := "GrnWves"]
#### Check again
transactionData[, .N, BRAND][order(BRAND)]
```

```
##           BRAND     N
##  1:     Burger  1564
##  2:        CCs  4551
##  3:    Cheetos  2927
##  4:   Cheezels  4603
##  5:       Cobs  9693
##  6:    Doritos 25224
##  7:     French  1418
##  8:    GrnWves  7740
##  9:  Infuzions 14201
## 10:     Kettle 41288
## 11:    Natural  7469
## 12:   Pringles 25102
## 13:        RRD 11894
## 14:        Red  4427
## 15:     Smiths 30353
## 16:   Sunbites  3008
```

```
## 17:      Thins 14075
## 18:   Tostitos  9471
## 19:   Twisties  9454
## 20:   Tyrrells  6442
## 21: Woolworths 11836
##          BRAND     N
```

```
# Over to you! Check the results look reasonable.
```

#### Examining customer data

Now that we are happy with the transaction dataset, let's have a look at the customer dataset.

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

```
## 'data.frame': 72637 obs. of 3 variables:
## $ LYLTY_CARD_NBR : int 1000 1002 1003 1004 1005 1007 1009 1010 1011 1012 ...
## $ LIFESTAGE : chr "YOUNG SINGLES/COUPLES" "YOUNG SINGLES/COUPLES" "YOUNG
FAMILIES" "OLDER SINGLES/COUPLES" ...
## $ PREMIUM_CUSTOMER: chr "Premium" "Mainstream" "Budget" "Mainstream" ...
```

```
summary(customerData)
```

```
##  LYLTY_CARD_NBR     LIFESTAGE          PREMIUM_CUSTOMER
##  Min.   :   1000   Length:72637       Length:72637
##  1st Qu.:  66202   Class :character   Class :character
##  Median : 134040   Mode  :character   Mode  :character
##  Mean   : 136186
##  3rd Qu.: 203375
##  Max.   :2373711
```

```
head(customerData,10)
```

```
##    LYLTY_CARD_NBR              LIFESTAGE PREMIUM_CUSTOMER
## 1            1000  YOUNG SINGLES/COUPLES          Premium
## 2            1002  YOUNG SINGLES/COUPLES       Mainstream
## 3            1003         YOUNG FAMILIES           Budget
## 4            1004  OLDER SINGLES/COUPLES       Mainstream
## 5            1005 MIDAGE SINGLES/COUPLES       Mainstream
## 6            1007  YOUNG SINGLES/COUPLES           Budget
## 7            1009           NEW FAMILIES          Premium
## 8            1010  YOUNG SINGLES/COUPLES       Mainstream
## 9            1011  OLDER SINGLES/COUPLES       Mainstream
## 10           1012         OLDER FAMILIES       Mainstream
```

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

As the number of rows in `data` is the same as that of `transactionData`, we can be sure that no duplicates were created. This is because we created `data` by setting `all.x = TRUE` (in other words, a left join) which means take all the rows in `transactionData` and find rows with matching values in shared columns and then joining the details in these rows to the `x` or the first mentioned table Let's also check if some customers were not matched on by checking for nulls.

```
sum(is.na(data$LIFESTAGE))
```

```
## [1] 0
```

```
sum(is.na(data$PREMIUM_CUSTOMER))
```

```
## [1] 0
```

Great, there are no nulls! So all our customers in the transaction data has been accounted for in the customer dataset. Note that if you are continuing with Task 2, you may want to retain this dataset which you can write out as a csv

```
write.csv(data,"QVI_data.csv")
```

Data exploration is now complete!

### Data analysis on customer segments

Now that the data is ready for analysis, we can define some metrics of interest to the client:

- Who spends the most on chips (total sales), describing customers by lifestage and how premium their general purchasing behaviour is

- How many customers are in each segment - How many chips are bought per customer by segment

- What's the average chip price by customer segment

We could also ask our data team for more information. Examples are:

- The customer's total spend over the period and total spend for each transaction to understand what proportion of their grocery spend is on chips

- Proportion of customers in each customer segment overall to compare against the mix of customers who purchase chips

Let's start with calculating total sales by LIFESTAGE and PREMIUM_CUSTOMER and plotting the split by these segments to describe which customer segment contribute most to chip sales.

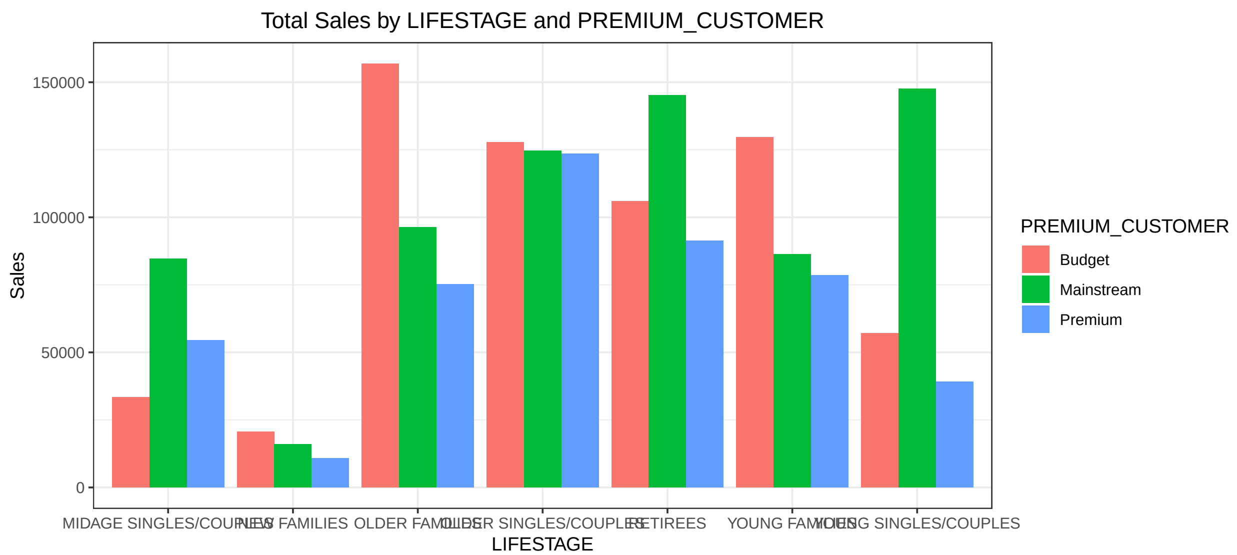
```
#### Total sales by LIFESTAGE and PREMIUM_CUSTOMER
total_sales <- data %>%
  group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarize(total_sales = sum(TOT_SALES))
```

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

```
# Print the results
print(total_sales)
```

```
## # A tibble: 21 x 3
## # Groups:   LIFESTAGE [7]
##    LIFESTAGE              PREMIUM_CUSTOMER total_sales
##    <chr>                  <chr>                  <dbl>
##  1 MIDAGE SINGLES/COUPLES Budget                33346.
##  2 MIDAGE SINGLES/COUPLES Mainstream            84734.
##  3 MIDAGE SINGLES/COUPLES Premium               54444.
##  4 NEW FAMILIES           Budget                20607.
##  5 NEW FAMILIES           Mainstream            15980.
##  6 NEW FAMILIES           Premium               10761.
##  7 OLDER FAMILIES         Budget               156864.
##  8 OLDER FAMILIES         Mainstream            96414.
##  9 OLDER FAMILIES         Premium               75243.
## 10 OLDER SINGLES/COUPLES  Budget               127834.
## # ... with 11 more rows
```

```
#Visualizing total sales:
ggplot(total_sales, aes(x = LIFESTAGE, y = total_sales, fill = PREMIUM_CUSTOMER)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Total Sales by LIFESTAGE and PREMIUM_CUSTOMER", x = "LIFESTAGE", y = "Sales", fill = "PI
```




Sales are coming mainly from Budget - older families, Mainstream - young singles/couples, and Mainstream - retirees Let's see if the higher sales are due to there being more customers who buy chips.

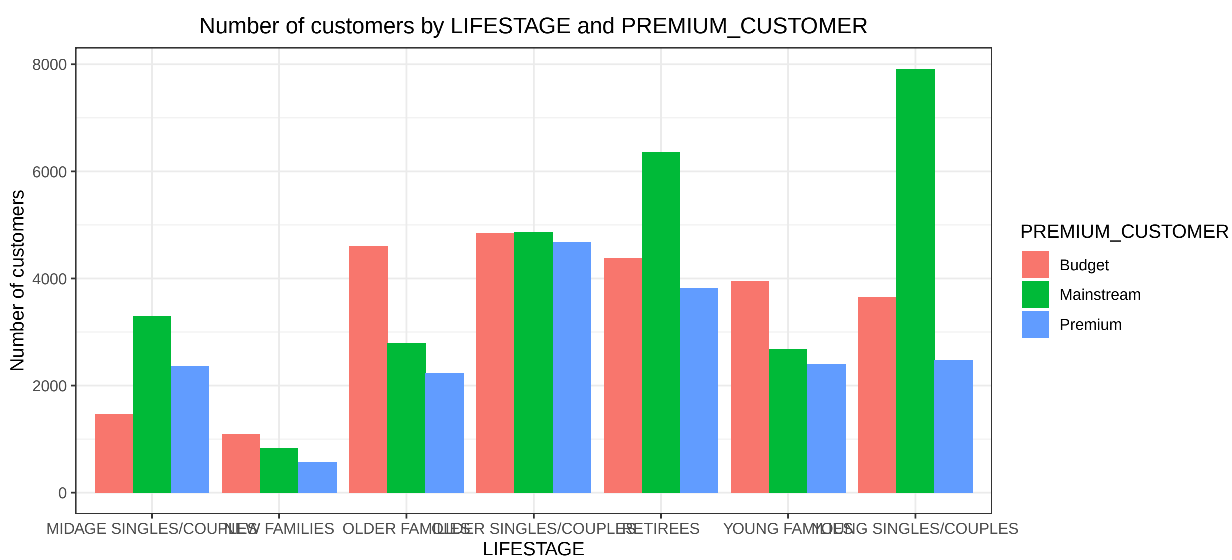
```
#### Number of customers by LIFESTAGE and PREMIUM_CUSTOMER
customers <- data %>%
  group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarize(customers = uniqueN(LYLTY_CARD_NBR))
```

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

```r
print(customers)
```

```
## # A tibble: 21 x 3
## # Groups:   LIFESTAGE [7]
##    LIFESTAGE              PREMIUM_CUSTOMER customers
##    <chr>                  <chr>                <int>
##  1 MIDAGE SINGLES/COUPLES Budget                1474
##  2 MIDAGE SINGLES/COUPLES Mainstream            3298
##  3 MIDAGE SINGLES/COUPLES Premium               2369
##  4 NEW FAMILIES           Budget                1087
##  5 NEW FAMILIES           Mainstream             830
##  6 NEW FAMILIES           Premium                575
##  7 OLDER FAMILIES         Budget                4611
##  8 OLDER FAMILIES         Mainstream            2788
##  9 OLDER FAMILIES         Premium               2231
## 10 OLDER SINGLES/COUPLES  Budget                4849
## # ... with 11 more rows
```

```r
ggplot(customers, aes(x = LIFESTAGE, y = customers, fill = PREMIUM_CUSTOMER)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Number of customers by LIFESTAGE and PREMIUM_CUSTOMER", x = "LIFESTAGE", y = "Number of
```




There are more Mainstream - young singles/couples and Mainstream - retirees who buy chips. This contributes to there being more sales to these customer segments but this is not a major driver for the Budget - Older families segment. Higher sales may also be driven by more units of chips being bought per customer. Let's have a look at this next.

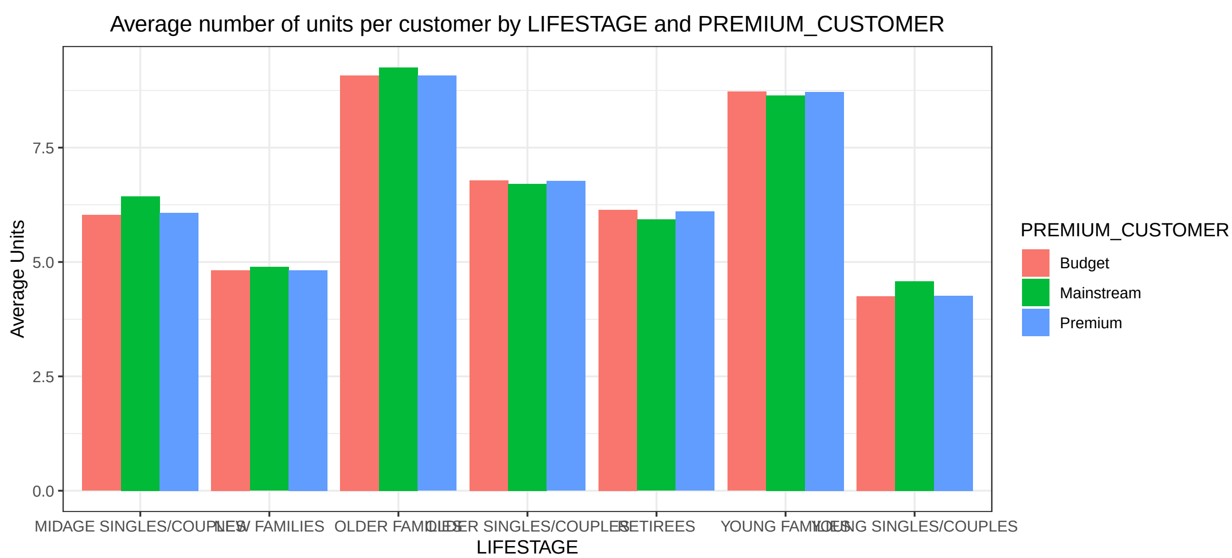
```r
#### Average number of units per customer by LIFESTAGE and PREMIUM_CUSTOMER
avg_units <- data %>%
  group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarize(avg_units = sum(PROD_QTY)/uniqueN(LYLTY_CARD_NBR))
```

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

```
print(avg_units)
```

```
## # A tibble: 21 x 3
## # Groups:   LIFESTAGE [7]
##    LIFESTAGE              PREMIUM_CUSTOMER avg_units
##    <chr>                  <chr>                <dbl>
##  1 MIDAGE SINGLES/COUPLES Budget                6.03
##  2 MIDAGE SINGLES/COUPLES Mainstream            6.43
##  3 MIDAGE SINGLES/COUPLES Premium               6.08
##  4 NEW FAMILIES           Budget                4.82
##  5 NEW FAMILIES           Mainstream            4.89
##  6 NEW FAMILIES           Premium               4.82
##  7 OLDER FAMILIES         Budget                9.08
##  8 OLDER FAMILIES         Mainstream            9.26
##  9 OLDER FAMILIES         Premium               9.07
## 10 OLDER SINGLES/COUPLES  Budget                6.78
## # ... with 11 more rows
```

```
ggplot(avg_units, aes(x = LIFESTAGE, y = avg_units, fill = PREMIUM_CUSTOMER)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Average number of units per customer by LIFESTAGE and PREMIUM_CUSTOMER", x = "LIFESTAGE
```




```
# Over to you! Calculate and plot the average number of units per customer by those two dimensions.
```

Older families and young families in general buy more chips per customer Let's also investigate the average price per unit chips bought for each customer segment as this is also a driver of total sales.

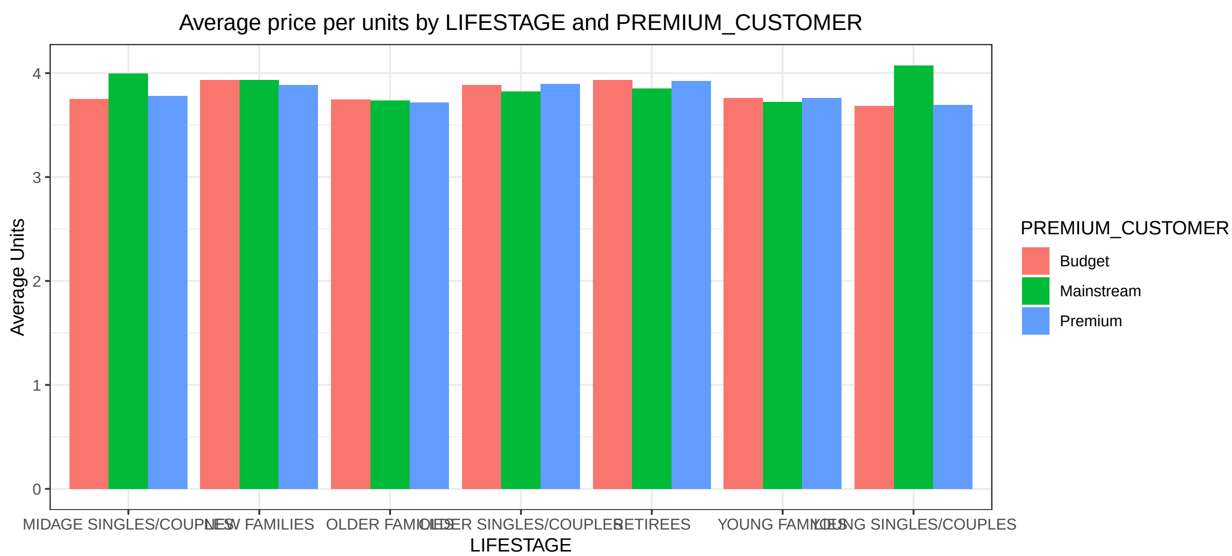
```r
#### Average price per unit by LIFESTAGE and PREMIUM_CUSTOMER
avg_price <- data %>%
  group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarize(avg_price = sum(TOT_SALES)/sum(PROD_QTY))
```

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

```r
print(avg_price)
```

```
## # A tibble: 21 x 3
## # Groups:   LIFESTAGE [7]
##    LIFESTAGE              PREMIUM_CUSTOMER avg_price
##    <chr>                  <chr>                <dbl>
##  1 MIDAGE SINGLES/COUPLES Budget                3.75
##  2 MIDAGE SINGLES/COUPLES Mainstream            3.99
##  3 MIDAGE SINGLES/COUPLES Premium               3.78
##  4 NEW FAMILIES           Budget                3.93
##  5 NEW FAMILIES           Mainstream            3.94
##  6 NEW FAMILIES           Premium               3.89
##  7 OLDER FAMILIES         Budget                3.75
##  8 OLDER FAMILIES         Mainstream            3.74
##  9 OLDER FAMILIES         Premium               3.72
## 10 OLDER SINGLES/COUPLES  Budget                3.89
## # ... with 11 more rows
```

```r
ggplot(avg_price, aes(x = LIFESTAGE, y = avg_price, fill = PREMIUM_CUSTOMER)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Average price per units by LIFESTAGE and PREMIUM_CUSTOMER", x = "LIFESTAGE", y = "Averag
```




Mainstream midage and young singles and couples are more willing to pay more per packet of chips compared to their budget and premium counterparts. This may be due to premium shoppers being more likely to buy healthy snacks and when they buy chips, this is mainly for entertainment purposes rather than their own consumption. This is also supported by there being fewer premium midage and young singles and couples buying chips compared to their mainstream counterparts.

As the difference in average price per unit isn't large, we can check if this difference is statistically different.

```
#### Perform an independent t-test between mainstream vs premium
#### and budget midage and
#### young singles and couples
data$price <- data$TOT_SALES/data$PROD_QTY

t.test(data[LIFESTAGE %in% c("YOUNG SINGLES/COUPLES","MIDAGE SINGLES/COUPLES") & PREMIUM_CUSTOMER == "Ma
```

```
##
## 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("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES") & PREMIUM_CUSTOMER !=
"Mainstream", 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
```

The t-test results in a p-value < 2.2e-16 which is < 0.05, i.e. the unit price for mainstream, young and mid-age singles and couples are significantly higher than that of budget or premium, young and midage singles and couples.

## Deep dive into specific customer segments for insights We have found quite a few interesting insights that we can dive deeper into. We might want to target customer segments that contribute the most to sales to retain them or further increase sales. Let's look at Mainstream - young singles/couples. For instance, let's find out if they tend to buy a particular brand of chips.

```
#### Deep dive into Mainstream, young singles/couples
set1 <- data[LIFESTAGE %in% c("YOUNG SINGLES/COUPLES","MIDAGE SINGLES/COUPLES") & PREMIUM_CUSTOMER == "M
set2 <- data[LIFESTAGE %in% c("YOUNG SINGLES/COUPLES","MIDAGE SINGLES/COUPLES") & PREMIUM_CUSTOMER != "M
#Brand affinity

quantity_set1 <- sum(set1$PROD_QTY)
quantity_set2 <- sum(set2$PROD_QTY)

target_segment <- set1 %>%
  group_by(BRAND) %>%
  summarise(target_segment = sum(PROD_QTY)/quantity_set1)
print(target_segment)
```

```
## # A tibble: 21 x 2
##    BRAND      target_segment
##    <chr>               <dbl>
##  1 Burger            0.00345
```

```
##   2 CCs              0.0124
##   3 Cheetos          0.00881
##   4 Cheezels         0.0187
##   5 Cobs             0.0447
##   6 Doritos          0.118
##   7 French           0.00390
##   8 GrnWves          0.0324
##   9 Infuzions        0.0636
##  10 Kettle           0.196
## # ... with 11 more rows
```

```
other_segment <- set2 %>%
  group_by(BRAND) %>%
  summarise(other_segment = sum(PROD_QTY)/quantity_set2)
print(other_segment)
```

```
## # A tibble: 21 x 2
##    BRAND     other_segment
##    <chr>             <dbl>
##  1 Burger          0.00814
##  2 CCs             0.0235
##  3 Cheetos         0.0125
##  4 Cheezels        0.0184
##  5 Cobs            0.0378
##  6 Doritos         0.0948
##  7 French          0.00675
##  8 GrnWves         0.0304
##  9 Infuzions       0.0563
## 10 Kettle          0.152
## # ... with 11 more rows
```

```
brand_proportions <- merge(target_segment,other_segment)
brand_proportions$affinityToBrand <- brand_proportions$target_segment/brand_proportions$other_segment

print(brand_proportions[order(brand_proportions$affinityToBrand),])
```

```
##         BRAND target_segment other_segment affinityToBrand
## 1      Burger    0.003447195   0.008144576       0.4232504
## 16   Sunbites    0.006302448   0.014850683       0.4243878
## 21 Woolworths    0.026776698   0.059706633       0.4484711
## 2         CCs    0.012378565   0.023522023       0.5262543
## 7      French    0.003899857   0.006746627       0.5780455
## 11    Natural    0.021466625   0.034705620       0.6185345
## 14        RRD    0.034820154   0.052737145       0.6602586
## 13        Red    0.012848637   0.019226873       0.6682645
## 3     Cheetos    0.008809499   0.012520767       0.7035910
## 15     Smiths    0.103311397   0.128773451       0.8022725
## 4    Cheezels    0.018663602   0.018416467       1.0134192
## 17      Thins    0.059194262   0.055836946       1.0601271
## 8     GrnWves    0.032434973   0.030369950       1.0679956
## 9   Infuzions    0.063599011   0.056323190       1.1291799
## 12   Pringles    0.113792263   0.098525062       1.1549575
## 5        Cobs    0.044709078   0.037764901       1.1838791
```

```
## 20   Tyrrells    0.029840872   0.024494510       1.2182678
## 6     Doritos    0.117639890   0.094837716       1.2404336
## 19   Twisties    0.045353250   0.035617326       1.2733480
## 18   Tostitos    0.044726488   0.035110823       1.2738661
## 10     Kettle    0.195985236   0.151768710       1.2913415
```

```
# Over to you! Work out of there are brands that these two customer segments prefer more than others. Y
```

We can see that : [INSIGHTS] Let's also find out if our target segment tends to buy larger packs of chips.

```
#### Preferred pack size compared to the rest of the population

packsize_target_segment <- set1 %>%
  group_by(PACK_SIZE) %>%
  summarise(packsize_target_segment = sum(PROD_QTY)/quantity_set1)
print(packsize_target_segment)
```

```
## # A tibble: 20 x 2
##    PACK_SIZE packsize_target_segment
##        <dbl>                   <dbl>
##  1        70                 0.00359
##  2        90                 0.00630
##  3       110                 0.105
##  4       125                 0.00308
##  5       134                 0.114
##  6       135                 0.0147
##  7       150                 0.159
##  8       160                 0.00738
##  9       165                 0.0562
## 10       170                 0.0803
## 11       175                 0.260
## 12       180                 0.00383
## 13       190                 0.00853
## 14       200                 0.0101
## 15       210                 0.0286
## 16       220                 0.00345
## 17       250                 0.0139
## 18       270                 0.0314
## 19       330                 0.0607
## 20       380                 0.0308
```

```
packsize_other_segment <- set2 %>%
  group_by(PACK_SIZE) %>%
  summarise(packsize_other_segment = sum(PROD_QTY)/quantity_set2)
print(packsize_other_segment)
```

```
## # A tibble: 20 x 2
##    PACK_SIZE packsize_other_segment
##        <dbl>                  <dbl>
##  1        70                0.00768
##  2        90                0.0149
##  3       110                0.0864
```

```
## 4      125             0.00707
## 5      134             0.0985
## 6      135             0.0120
## 7      150             0.161
## 8      160             0.0164
## 9      165             0.0650
## 10     170             0.0796
## 11     175             0.273
## 12     180             0.00600
## 13     190             0.0134
## 14     200             0.0222
## 15     210             0.0244
## 16     220             0.00814
## 17     250             0.0116
## 18     270             0.0240
## 19     330             0.0443
## 20     380             0.0244
```

```
pack_proportions <- merge(packsize_target_segment,packsize_other_segment)
pack_proportions$affinityToPack <- pack_proportions$packsize_target_segment/pack_proportions$packsize_o

print(pack_proportions)
```

```
##    PACK_SIZE packsize_target_segment packsize_other_segment affinityToPack
## 1         70             0.003586476            0.007678593      0.4670746
## 2         90             0.006302448            0.014850683      0.4243878
## 3        110             0.104721613            0.086409498      1.2119225
## 4        125             0.003081584            0.007070789      0.4358189
## 5        134             0.113792263            0.098525062      1.1549575
## 6        135             0.014676695            0.012034523      1.2195493
## 7        150             0.158640621            0.161351757      0.9831974
## 8        160             0.007381873            0.016410714      0.4498203
## 9        165             0.056182318            0.064954009      0.8649554
## 10       170             0.080260455            0.079602091      1.0082707
## 11       175             0.260002089            0.272681227      0.9535020
## 12       180             0.003830217            0.005997001      0.6386887
## 13       190             0.008530938            0.013351432      0.6389530
## 14       200             0.010115255            0.022225374      0.4551219
## 15       210             0.028604756            0.024372949      1.1736272
## 16       220             0.003447195            0.008144576      0.4232504
## 17       250             0.013928062            0.011609060      1.1997579
## 18       270             0.031425189            0.024008266      1.3089320
## 19       330             0.060708938            0.044308927      1.3701288
## 20       380             0.030781016            0.024413469      1.2608211
```

```
data[PACK_SIZE == 270, unique(PROD_NAME)]
```

```
## [1] "Twisties Cheese     270g" "Twisties Chicken270g"
```

```
data[PACK_SIZE == 330, unique(PROD_NAME)]
```

```
## [1] "Doritos Cheese      Supreme 330g"
```

```
## [2] "Smiths Crinkle      Original 330g"
## [3] "Smiths Crinkle Chips Salt & Vinegar 330g"
## [4] "Cheezels Cheese 330g"
```

The mainstream young single/couple are more likely to buy 270 and 330g package which account for 30% and 37% more than the other pack size of chips. Investigating the brand name of them then we found that Twisties Cheese is the favourite 270g pack. The 330g packs have several options: Doritos Cheese, Smiths Crinkle Original and Smiths Crinkle Chips Salts and Vinegar flavour.

## [INSIGHTS]

It is clear that Mainstream young singles and couples are likely to spend more money on chips. The retiree shoppers also spend on it more than the other group of people.

They prefer Tyrells, Twisties, Kettle, Tostitos more than the other brands. This result could help to focus on those branding more than the others since they are the most preferable brands of the main customer segmentation.

The pack size of 270g and 330g also need to consider to focus because young single and couple prefer them more than the other. However, the Smiths for 2 flavor: "Original" and "Salts & Vinegar" are their favourite, it maybe because of the convenient pack size for them.