Retail Strategy and Customer Analytics

```
####Install packages
##install.packages("tidyverse")
#install.packages("data.table")
#install.packages("dplyr")
#install.packages("ggplot2")
#install.packages("ggmosaic")
#install.packages("readr")
#install.packages("readxl")
#install.packages("arules")
#install.packages("stringr")
#install.packages("arulesViz")

#### Load required libraries
##library(tidyverse)
library(dplyr, warn.conflicts = FALSE)
library(data.table, warn.conflicts = FALSE)
library(ggplot2)
library(ggmosaic)
library(readr)
library(readxl)
library(stringr)
library(arules, warn.conflicts = FALSE)
```

```
## Loading required package: Matrix
```

```
library(arulesViz, warn.conflicts = FALSE)

####  Load the datasets and
#### assign the data files to data.tables

transactionData <- read_excel("C:/Users/Devi Prasad/Desktop/Quantium Data
↪  Analytics/QVI_transaction_data.xlsx")
transactionData <- data.table(transactionData)

customerData <- read.csv("C:/Users/Devi Prasad/Desktop/Quantium Data
↪  Analytics/QVI_purchase_behaviour.csv")
customerData <- data.table(customerData)
```

Exploratory data analysis

```
head(transactionData,10)
```

```
##      DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
##     <num>     <num>          <num>  <num>    <num>
##  1: 43390         1           1000      1        5
##  2: 43599         1           1307    348       66
##  3: 43605         1           1343    383       61
##  4: 43329         2           2373    974       69
##  5: 43330         2           2426   1038      108
##  6: 43604         4           4074   2982       57
##  7: 43601         4           4149   3333       16
##  8: 43601         4           4196   3539       24
##  9: 43332         5           5026   4525       42
## 10: 43330         7           7150   6900       52
##                                     PROD_NAME PROD_QTY TOT_SALES
##                                        <char>    <num>     <num>
##  1:   Natural Chip        Compny SeaSalt175g        2       6.0
##  2:                 CCs Nacho Cheese    175g        3       6.3
##  3:   Smiths Crinkle Cut  Chips Chicken 170g        2       2.9
##  4:   Smiths Chip Thinly  S/Cream&Onion 175g        5      15.0
##  5: Kettle Tortilla ChpsHny&Jlpno Chili 150g        3      13.8
##  6: Old El Paso Salsa   Dip Tomato Mild 300g        1       5.1
##  7: Smiths Crinkle Chips Salt & Vinegar 330g        1       5.7
##  8:    Grain Waves         Sweet Chilli 210g        1       3.6
##  9:  Doritos Corn Chip Mexican Jalapeno 150g        1       3.9
## 10:    Grain Waves Sour    Cream&Chives 210G        2       7.2
```

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
#### CSV and Excel integer dates begin on 30 Dec 1899
transactionData$DATE <- as.Date(transactionData$DATE, origin = "1899-12-30")
head(transactionData)
```

```
##          DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
##        <Date>     <num>          <num>  <num>    <num>
## 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: 2019-05-19         4           4074   2982       57
##                                     PROD_NAME PROD_QTY TOT_SALES
##                                        <char>    <num>     <num>
## 1:   Natural Chip        Compny SeaSalt175g        2       6.0
## 2:                 CCs Nacho Cheese    175g        3       6.3
## 3:   Smiths Crinkle Cut  Chips Chicken 170g        2       2.9
## 4:   Smiths Chip Thinly  S/Cream&Onion 175g        5      15.0
## 5: Kettle Tortilla ChpsHny&Jlpno Chili 150g        3      13.8
## 6: Old El Paso Salsa   Dip Tomato Mild 300g        1       5.1
```

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

```
transactionData[, .N, PROD_NAME]
```

Examine PROD_NAME

```
##                                     PROD_NAME     N
##                                        <char> <int>
##   1:   Natural Chip        Compny SeaSalt175g  1468
##   2:                 CCs Nacho Cheese    175g  1498
##   3:   Smiths Crinkle Cut  Chips Chicken 170g  1484
##   4:   Smiths Chip Thinly  S/Cream&Onion 175g  1473
##   5: Kettle Tortilla ChpsHny&Jlpno Chili 150g  3296
##  ---
## 110:    Red Rock Deli Chikn&Garlic Aioli 150g  1434
## 111:      RRD SR Slow Rst     Pork Belly 150g  1526
## 112:                 RRD Pc Sea Salt     165g  1431
## 113:       Smith Crinkle Cut   Bolognese 150g  1451
## 114:                 Doritos Salsa Mild  300g  1472
```

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.

```
#### 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()`.

```
# Remove digits, and special characters, and then sort the distinct
#### words by frequency of occurrence.
#### Removing digits
productWords <- productWords[!grepl("[0-9]", words)]

#### Removing special characters
productWords <- productWords[!grepl("[^[:alnum:] ]", words)]

#### Removing empty rows
productWords <- productWords[!apply(productWords == "", 1, all),]

#### 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

wordcount <- productWords %>%
  dplyr::count(words,sort = TRUE)
View(wordcount)
```

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

```
#### 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).

```
#### 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
```

```
#Check the number of missing values present in each column
colSums(is.na(transactionData))
```

```
##           DATE      STORE_NBR LYLTY_CARD_NBR         TXN_ID       PROD_NBR
##              0              0              0              0              0
##      PROD_NAME       PROD_QTY      TOT_SALES
##              0              0              0
```

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.

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

```
##          DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
##        <Date>     <num>          <num>  <num>    <num>
## 1: 2018-08-19       226         226000 226201        4
## 2: 2019-05-20       226         226000 226210        4
##                          PROD_NAME PROD_QTY TOT_SALES
##                             <char>    <num>     <num>
## 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.

```
#### Let's see if the customer has had other transactions
# Filter to see what other transactions that customer made.
transactionData[LYLTY_CARD_NBR == 226000, ]
```

```
##          DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
##        <Date>     <num>          <num>  <num>    <num>
## 1: 2018-08-19       226         226000 226201        4
## 2: 2019-05-20       226         226000 226210        4
##                          PROD_NAME PROD_QTY TOT_SALES
##                             <char>    <num>     <num>
## 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.

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

#### Re-examine transaction data
View(transactionData)
```

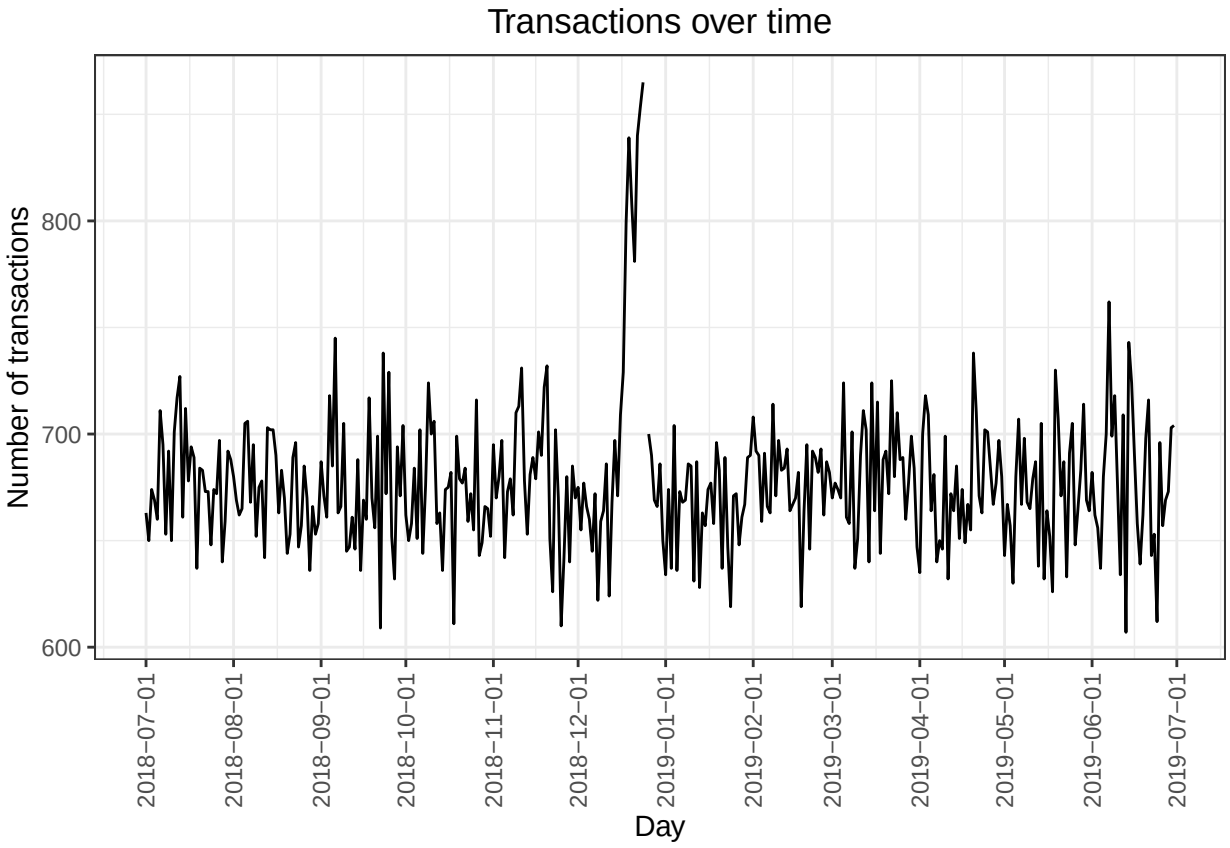
That's better. Now, let's look at the number of transaction lines over time to see if there are any obvious data issues such as missing data.

```
#### Count the number of transactions by date
#Create a summary of transaction count by date.
transactionCount <- dplyr::count(transactionData, DATE)
```

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.

```
#### Create a sequence of dates and join this the count of transactions by date
#create a column of dates that includes every day from 1 Jul 2018 to
#30 Jun 2019, and join it onto the data to fill in the missing day.
# Create a sequence of dates
date_sequence <- seq.Date(from = as.Date("2018-07-01"), to = as.Date("2019-06-30"), by =
↪  "day")

date_df <- data.frame(DATE = date_sequence)

# Join the date_df with transactionCount DATE based on the 'DATE' column
transactions_by_day <- full_join(transactionCount, date_df, by = "DATE")

#### Setting plot themes to format graphs
theme_set(theme_bw())
theme_update(plot.title = element_text(hjust = 0.5))

#### Plot transactions over time
```

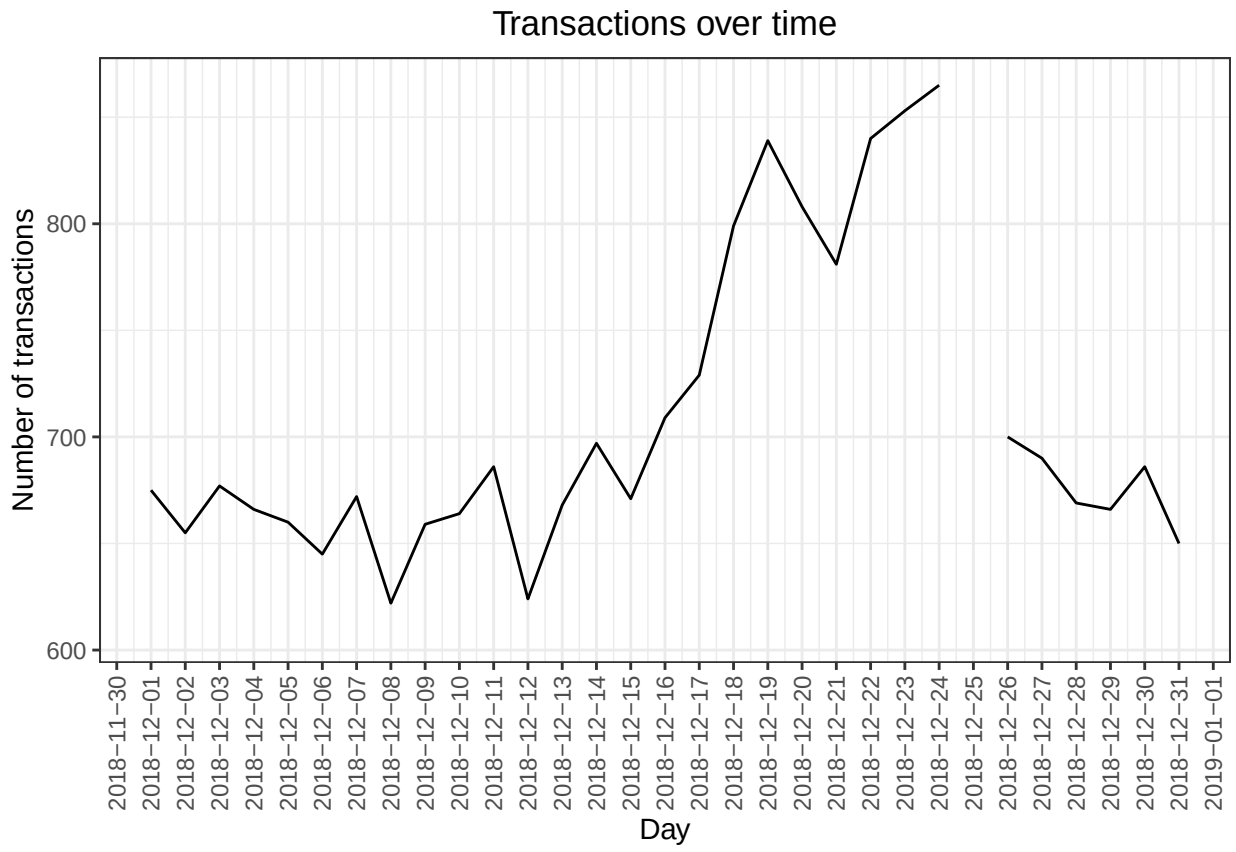
```
ggplot(transactions_by_day, aes(x = DATE, y = n)) +
  geom_line() +
  labs(x = "Day", y = "Number of transactions", title = "Transactions over time") +
  scale_x_date(breaks = "1 month") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```



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
#recreate the chart above zoomed in to the relevant dates.
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(breaks = "1 day", limits = as.Date(c("2018-12-01", "2018-12-31"))) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```

```
## Warning: Removed 334 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



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.

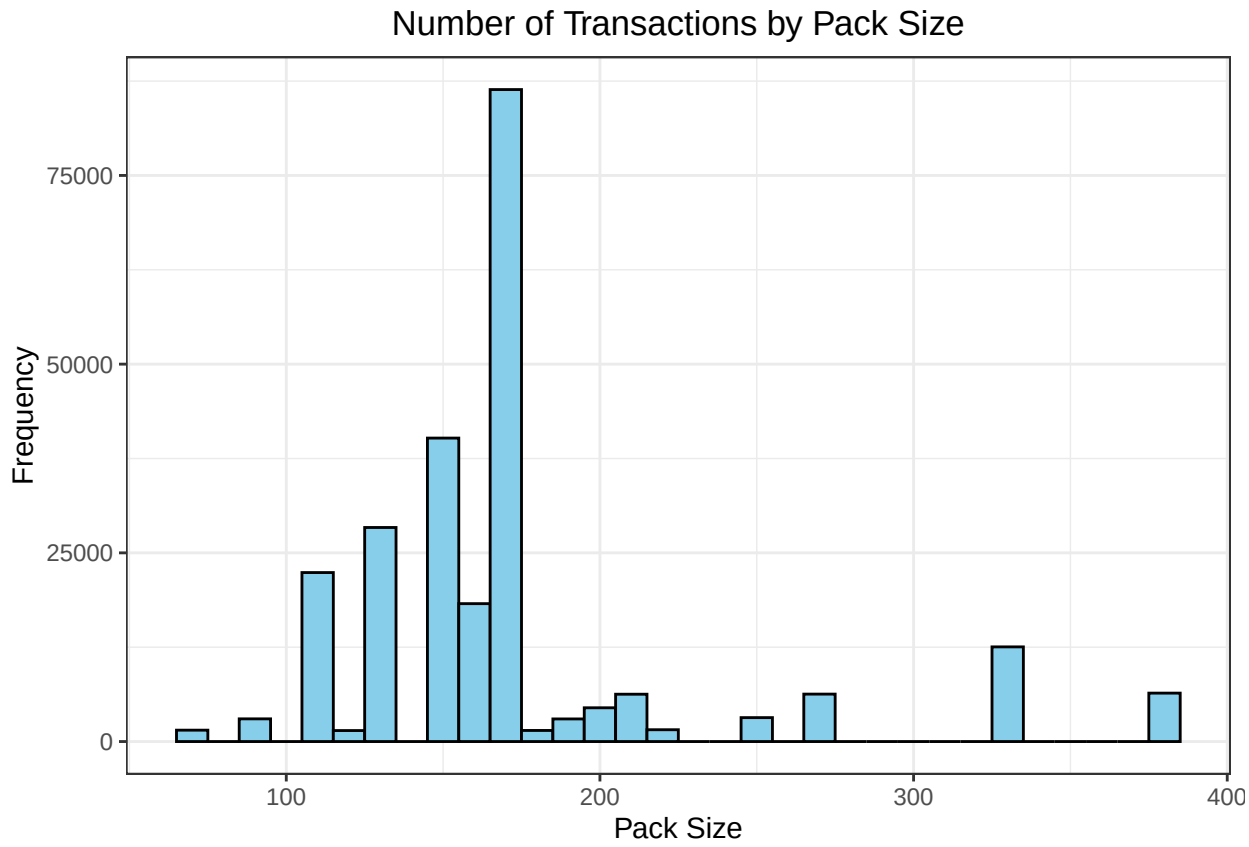
```
#### Pack size
#### We can work this out by taking the digits that are in PROD_NAME
transactionData[, PACK_SIZE := parse_number(PROD_NAME)]

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

```
##     PACK_SIZE     N
##         <num> <int>
##  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
##     PACK_SIZE     N
```

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

```
#### Let's plot a histogram of PACK_SIZE since we know that it is a categorical
#variable and not a continuous variable even though it is numeric.
#Plot a histogram showing the number of transactions by pack size.
# Create the histogram
ggplot(transactionData, aes(x = PACK_SIZE)) +
  geom_histogram(binwidth = 10, fill = "skyblue", color = "black") +
  labs(title = "Number of Transactions by Pack Size",
       x = "Pack Size",
       y = "Frequency")
```



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

```
#### Brands
#Create a column which contains the brand of the product, by extracting it from the
↪  product name.
transactionData[, BRAND := toupper(substr(PROD_NAME, 1, regexpr(pattern = ' ',
                                                                PROD_NAME) - 1))]
#### Checking brands
transactionData[, .N, by = BRAND][order(N)]
```

```
##          BRAND     N
##         <char> <int>
##  1:     FRENCH  1418
##  2:        NCC  1419
##  3:   SUNBITES  1432
##  4:    GRNWVES  1468
##  5: WOOLWORTHS  1516
##  6:     BURGER  1564
##  7:      SNBTS  1576
##  8:    CHEETOS  2927
##  9:      SMITH  2963
## 10:     INFZNS  3144
## 11:     DORITO  3183
## 12:        RED  4427
## 13:        CCS  4551
## 14:   CHEEZELS  4603
## 15:    NATURAL  6050
## 16:      GRAIN  6272
## 17:   TYRRELLS  6442
## 18:   TWISTIES  9454
## 19:   TOSTITOS  9471
## 20:       COBS  9693
## 21:         WW 10320
## 22:  INFUZIONS 11057
## 23:        RRD 11894
## 24:      THINS 14075
## 25:    DORITOS 22041
## 26:   PRINGLES 25102
## 27:     SMITHS 27390
## 28:     KETTLE 41288
##          BRAND     N
```

```
#Check the results look reasonable.

#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 == "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 again
transactionData[, .N, by = BRAND][order(BRAND)]
```

```
##          BRAND     N
##         <char> <int>
##  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 16321
## 14:     SMITHS 30353
## 15:   SUNBITES  3008
## 16:      THINS 14075
## 17:   TOSTITOS  9471
## 18:   TWISTIES  9454
## 19:   TYRRELLS  6442
## 20: WOOLWORTHS 11836
##          BRAND     N
```

```
### Examining customer data
#Now that we are happy with the transaction dataset, let's have a look at the customer
↪  dataset.

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

```
##    LYLTY_CARD_NBR               LIFESTAGE PREMIUM_CUSTOMER
##             <int>                  <char>           <char>
## 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
```

```
#Do some basic summaries of the dataset, including distributions of any key columns.
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
```

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

# Print the result
print(data)
```

```
## Key: <LYLTY_CARD_NBR>
##         LYLTY_CARD_NBR       DATE STORE_NBR TXN_ID PROD_NBR
##                  <int>     <Date>     <num>  <num>    <num>
##      1:           1000 2018-10-17         1      1        5
##      2:           1002 2018-09-16         1      2       58
##      3:           1003 2019-03-07         1      3       52
##      4:           1003 2019-03-08         1      4      106
##      5:           1004 2018-11-02         1      5       96
##     ---
## 246736:        2370651 2018-08-03        88 240350        4
## 246737:        2370701 2018-12-08        88 240378       24
## 246738:        2370751 2018-10-01        88 240394       60
## 246739:        2370961 2018-10-24        88 240480       70
## 246740:        2373711 2018-12-14        88 241815       16
##                                     PROD_NAME PROD_QTY TOT_SALES PACK_SIZE
##                                        <char>    <num>     <num>     <num>
##      1:   Natural Chip        Compny SeaSalt175g        2       6.0       175
##      2:    Red Rock Deli Chikn&Garlic Aioli 150g        1       2.7       150
##      3:    Grain Waves Sour    Cream&Chives 210G        1       3.6       210
##      4:   Natural ChipCo      Hony Soy Chckn175g        1       3.0       175
##      5:           WW Original Stacked Chips 160g        1       1.9       160
##     ---
## 246736:         Dorito Corn Chp     Supreme 380g        2      13.0       380
## 246737:    Grain Waves         Sweet Chilli 210g        2       7.2       210
## 246738:     Kettle Tortilla ChpsFeta&Garlic 150g        2       9.2       150
## 246739:  Tyrrells Crisps     Lightly Salted 165g        2       8.4       165
## 246740: Smiths Crinkle Chips Salt & Vinegar 330g        2      11.4       330
##              BRAND              LIFESTAGE PREMIUM_CUSTOMER
##             <char>                 <char>           <char>
##      1:    NATURAL  YOUNG SINGLES/COUPLES          Premium
##      2:        RRD  YOUNG SINGLES/COUPLES       Mainstream
##      3:    GRNWVES         YOUNG FAMILIES           Budget
##      4:    NATURAL         YOUNG FAMILIES           Budget
##      5: WOOLWORTHS  OLDER SINGLES/COUPLES       Mainstream
##     ---
## 246736:    DORITOS MIDAGE SINGLES/COUPLES       Mainstream
## 246737:    GRNWVES         YOUNG FAMILIES       Mainstream
## 246738:     KETTLE         YOUNG FAMILIES          Premium
## 246739:   TYRRELLS         OLDER FAMILIES           Budget
## 246740:     SMITHS  YOUNG SINGLES/COUPLES       Mainstream
```

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.

```
#Check for missing customer details
#See if any transactions did not have a matched customer.
colSums(is.na(data))
```

```
##   LYLTY_CARD_NBR             DATE        STORE_NBR           TXN_ID
##                0                0                0                0
##         PROD_NBR        PROD_NAME         PROD_QTY        TOT_SALES
##                0                0                0                0
##        PACK_SIZE            BRAND        LIFESTAGE PREMIUM_CUSTOMER
##                0                0                0                0
```

Great, there are no nulls! So all our customers in the transaction data has been #accounted for in the customer dataset.

```
#Save this dataset as a csv
filePath = "C:/Users/Devi Prasad/Desktop/Quantium Data Analytics/"
fwrite(data, paste0(filePath,"QVI_data.csv"))
```

Data analysis on customer segments

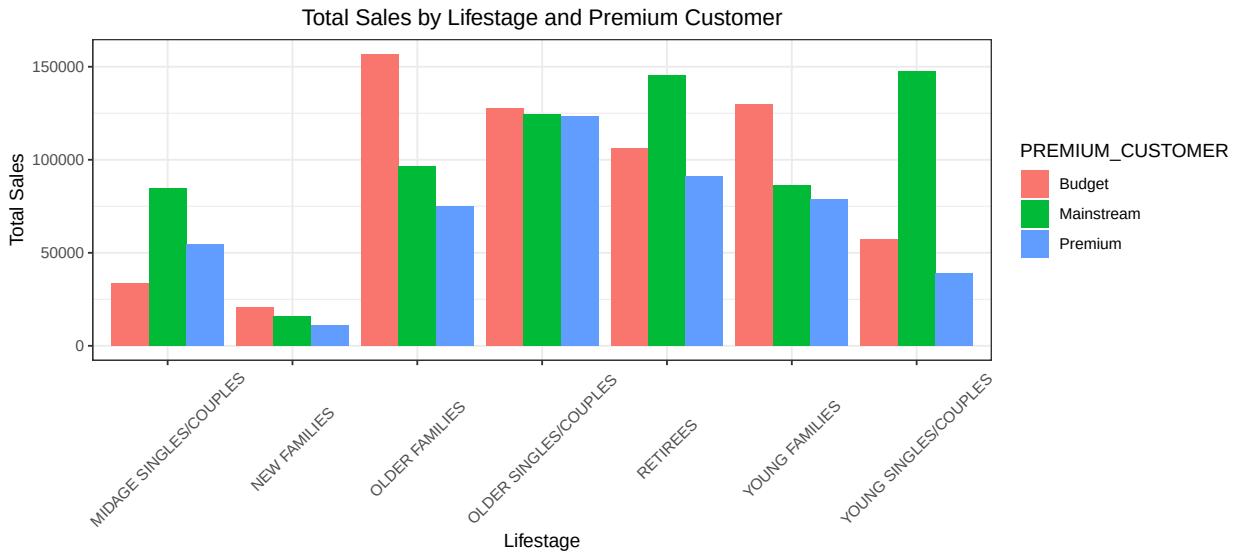
Now that the data is ready for analysis, we can define some metrics of interest to the client: - Who spends the most on chips (total sales), describing customers by lifestage and - how premium their generalpurchasing behaviour is - How many customers are in each segment - How many chips are bought per 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
#Calculate the summary of sales by those dimensions and create a plot.
# Group and summarize data
summary_sales <- data %>%
  group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarise(total_sales = sum(TOT_SALES))
```

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

```
View(summary_sales)

# Create a bar chart
ggplot(data = summary_sales, aes(x = LIFESTAGE, y = total_sales, fill =
↪  PREMIUM_CUSTOMER)) +
  geom_bar(stat = "identity", position = "dodge") +
```

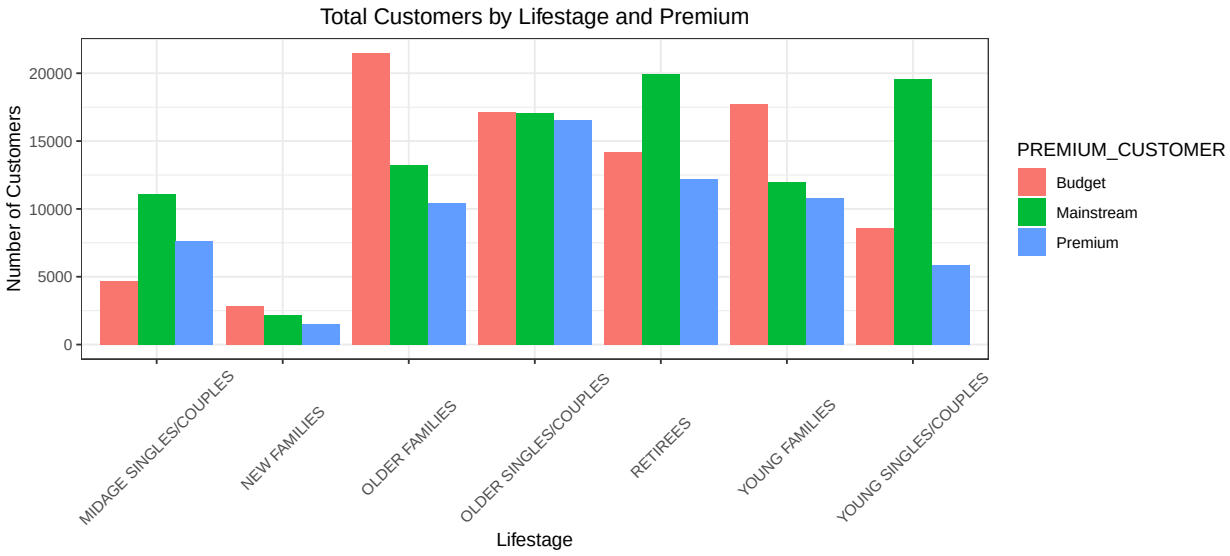
```
  labs(x = "Lifestage", y = "Total Sales", title = "Total Sales by Lifestage and Premium
  ↪  Customer") +
  theme(axis.text.x = element_text(angle = 45, vjust = 0.5))
```



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
#Calculate the summary of number of customers by those dimensions and create a plot.
summary_customers <- data %>%
  count(LIFESTAGE, PREMIUM_CUSTOMER, sort = TRUE)
View(summary_customers)

#Create a bar chart
ggplot(data = summary_customers, aes(x = LIFESTAGE, y = n, fill = PREMIUM_CUSTOMER)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Lifestage", y = "Number of Customers", title = "Total Customers by Lifestage
  ↪  and Premium") +
  theme(axis.text.x = element_text(angle = 45, vjust = 0.5))
```



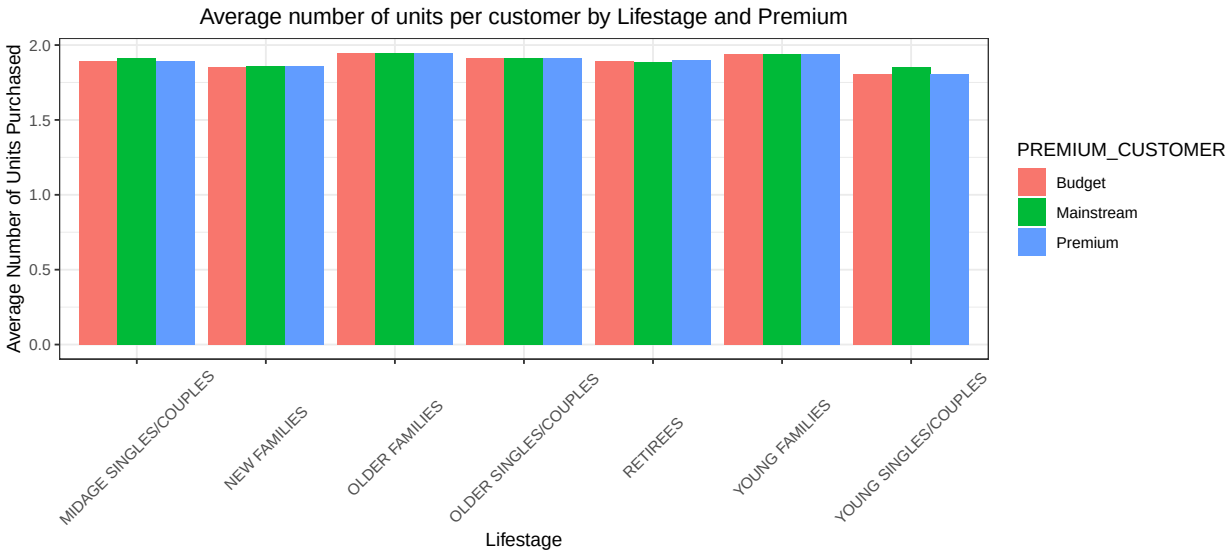
There are more Mainstream - young singles/couples and Mainstream - retirees who buy chips. This contributes to there being more sales to these customer segments but this is not a major driver for the Budget - Older families segment.

Higher sales may also be driven by more units of chips being bought per customer. Let's have a look at this next.

```
#### Average number of units per customer by LIFESTAGE and PREMIUM_CUSTOMER
#Calculate and plot the average number of units per customer by those two dimensions.
average_units <- data %>%
  group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
summarise(mean_qty=mean(PROD_QTY))
```

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

```
View(average_units)

#Create a bar chart
ggplot(data = average_units, aes(x = LIFESTAGE, y = mean_qty, fill = PREMIUM_CUSTOMER)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Lifestage", y = "Average Number of Units Purchased", title = "Average number
  ↪  of units per customer by Lifestage and Premium") +
  theme(axis.text.x = element_text(angle = 45, vjust = 0.5))
```



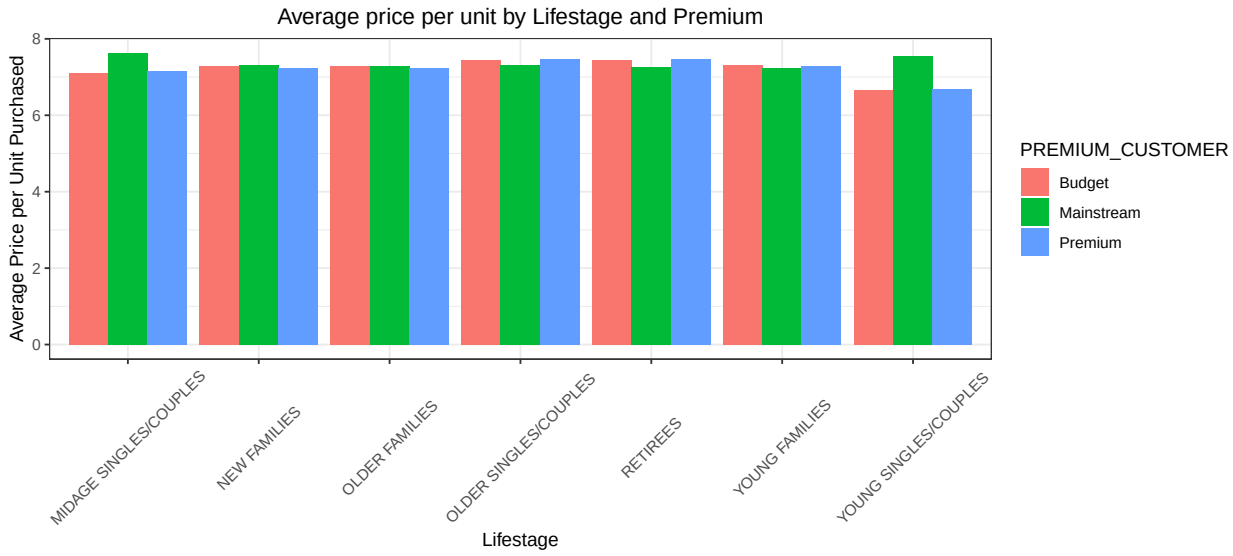
Older families and young families in general buy more chips per customer.

Let's also investigate the average price per unit chips bought for each customer segment as this is also a driver of total sales.

```
#### Average price per unit by LIFESTAGE and PREMIUM_CUSTOMER
#Calculate and plot the average price per unit sold (average sale
#price) by those two customer dimensions.
average_price <- data %>%
  group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarise(mean_price=mean(TOT_SALES))
```

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

```
View(average_price)

#Create a bar chart
ggplot(data = average_price, aes(x = LIFESTAGE, y = mean_price, fill = PREMIUM_CUSTOMER))
↪  +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Lifestage", y = "Average Price per Unit Purchased", title = "Average price
  ↪  per unit by Lifestage and Premium") +
  theme(axis.text.x = element_text(angle = 45, vjust = 0.5))
```



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

pricePerUnit <- data[, price := TOT_SALES/PROD_QTY]
# Perform a t-test to see if the difference is significant.
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")
```

```
##
## 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
```

#Insights The t-test results in a p-value $<$ 2.2e-16, 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
# Let's check if there are brands that these two customer segments prefer more than
↪  others.

segment1 <- data[LIFESTAGE == "YOUNG SINGLES/COUPLES" & PREMIUM_CUSTOMER ==
                   "Mainstream",]
other <- data[!(LIFESTAGE == "YOUNG SINGLES/COUPLES" & PREMIUM_CUSTOMER ==
                  "Mainstream"),]

#### Brand affinity compared to the rest of the population
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)]
```

```
##          BRAND targetSegment       other affinityToBrand
##         <char>         <num>       <num>           <num>
##  1:   TYRRELLS   0.031552795 0.025692464       1.2280953
##  2:   TWISTIES   0.046183575 0.037876520       1.2193194
##  3:    DORITOS   0.122760524 0.101074684       1.2145526
##  4:     KETTLE   0.197984817 0.165553442       1.1958967
##  5:   TOSTITOS   0.045410628 0.037977861       1.1957131
##  6:   PRINGLES   0.119420290 0.100634769       1.1866703
##  7:       COBS   0.044637681 0.039048861       1.1431238
##  8:  INFUZIONS   0.064679089 0.057064679       1.1334347
##  9:      THINS   0.060372671 0.056986370       1.0594230
## 10:    GRNWVES   0.032712215 0.031187957       1.0488733
## 11:   CHEEZELS   0.017971014 0.018646902       0.9637534
## 12:     SMITHS   0.096369910 0.124583692       0.7735355
## 13:     FRENCH   0.003947550 0.005758060       0.6855694
## 14:    CHEETOS   0.008033126 0.012066591       0.6657329
```

```
## 15:         RRD   0.043809524 0.067493678       0.6490908
## 16:     NATURAL   0.019599724 0.030853989       0.6352412
## 17:         CCS   0.011180124 0.018895650       0.5916771
## 18:    SUNBITES   0.006349206 0.012580210       0.5046980
## 19: WOOLWORTHS   0.024099379 0.049427188       0.4875733
## 20:      BURGER   0.002926156 0.006596434       0.4435967
##           BRAND targetSegment       other affinityToBrand
```

#We can see that : Mainstream young singles/couples are 22% more likely to purchase Tyrrells chips compared to the rest of the population. Mainstream young singles/couples are 56% less likely to purchase Burger Rings compared to the rest of the population.

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
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)]
```

```
##     PACK_SIZE targetSegment       other affinityToPack
##         <num>         <num>       <num>          <num>
##  1:       270   0.031828847 0.025095929      1.2682873
##  2:       380   0.032160110 0.025584213      1.2570295
##  3:       330   0.061283644 0.050161917      1.2217166
##  4:       134   0.119420290 0.100634769      1.1866703
##  5:       110   0.106280193 0.089791190      1.1836372
##  6:       210   0.029123533 0.025121265      1.1593180
##  7:       135   0.014768806 0.013075403      1.1295106
##  8:       250   0.014354727 0.012780590      1.1231662
##  9:       170   0.080772947 0.080985964      0.9973697
## 10:       150   0.157598344 0.163420656      0.9643722
## 11:       175   0.254989648 0.270006956      0.9443818
## 12:       165   0.055652174 0.062267662      0.8937572
## 13:       190   0.007481021 0.012442016      0.6012708
## 14:       180   0.003588682 0.006066692      0.5915385
## 15:       160   0.006404417 0.012372920      0.5176157
## 16:        90   0.006349206 0.012580210      0.5046980
## 17:       125   0.003008972 0.006036750      0.4984423
## 18:       200   0.008971705 0.018656115      0.4808989
## 19:        70   0.003036577 0.006322350      0.4802924
## 20:       220   0.002926156 0.006596434      0.4435967
##     PACK_SIZE targetSegment       other affinityToPack
```

It looks like Mainstream young singles/couples are 26% more likely to purchase a 270g pack of chips #compared to the rest of the population but let's dive into what brands sell this pack size.

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

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

Twisties are the only brand offering 270g packs and so this may instead be reflecting a higher likelihood of purchasing Twisties.

Conclusion:

Sales have mainly been due to Budget - older families, Mainstream - young singles/couples, and Mainstream-retirees shoppers.

We found that the high spend in chips for mainstream young singles/couples and retirees is due to there being more of them than other buyers.

Mainstream, midage and young singles and couples are also more likely to pay more per packet of chips.This is indicative of impulse buying behaviour.

We've also found that Mainstream young singles and couples are 22% more likely to purchase Tyrrells chips compared to the rest of the population. The Category Manager may want to increase the category's performance by off-locating some Tyrrells and smaller packs of chips in space near segments where young singles and couples frequent more often to increase visibilty and impulse behaviour.