Quantium Virtual Internship - Retail Strategy and Analytics - Task 1

Emmanuel Aina

2025-06-02

This file is a solution template for the Task 1 of the Quantium Virtual Internship.

Load required libraries and datasets

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

```r
library(data.table)
library(ggplot2)
library(ggmosaic)
library(readr)
purchase_behaviour <- read_csv("QVI_purchase_behaviour.csv")
```

```
## Rows: 72637 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): LIFESTAGE, PREMIUM_CUSTOMER
## dbl (1): LYLTY_CARD_NBR
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
transaction_data <- read_csv("QVI_transaction_dataa.csv")
```

```
## Rows: 264836 Columns: 8
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): DATE, PROD_NAME
## dbl (6): STORE_NBR, LYLTY_CARD_NBR, TXN_ID, PROD_NBR, PROD_QTY, TOT_SALES
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Exploratory data analysis

The first step in any analysis is to first understand the data. Let's take a look at each of the data sets provided.

Examining transaction_data

```r
#### Load necessary libraries
library(data.table)

#### Check the structure of the data set
str(transaction_data)
```

```
## spc_tbl_ [264,836 x 8] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
## $ DATE : chr [1:264836] "17/10/18" "14/05/19" "20/05/19" "17/08/18" ...
## $ 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 ...
## - attr(*, "spec")=
## .. cols(
## ..   DATE = col_character(),
## ..   STORE_NBR = col_double(),
```

```
## ..   LYLTY_CARD_NBR = col_double(),
## ..   TXN_ID = col_double(),
## ..   PROD_NBR = col_double(),
## ..   PROD_NAME = col_character(),
## ..   PROD_QTY = col_double(),
## ..   TOT_SALES = col_double()
## .. )
## - attr(*, "problems")=<externalptr>
```

```r
#### Display the first 10 rows
head(transaction_data, 10)
```

```
## # A tibble: 10 x 8
## DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR PROD_NAME PROD_QTY TOT_SALES
## <chr> <dbl> <dbl> <dbl> <dbl> <chr> <dbl> <dbl>
## 1 17/10/~ 1 1000 1 5 Natural ~ 2 6
## 2 14/05/~ 1 1307 348 66 CCs Nach~ 3 6.3
## 3 20/05/~ 1 1343 383 61 Smiths C~ 2 2.9
## 4 17/08/~ 2 2373 974 69 Smiths C~ 5 15
## 5 18/08/~ 2 2426 1038 108 Kettle T~ 3 13.8
## 6 19/05/~ 4 4074 2982 57 Old El P~ 1 5.1
## 7 16/05/~ 4 4149 3333 16 Smiths C~ 1 5.7
## 8 16/05/~ 4 4196 3539 24 Grain Wa~ 1 3.6
## 9 20/08/~ 5 5026 4525 42 Doritos ~ 1 3.9
## 10 18/08/~ 7 7150 6900 52 Grain Wa~ 2 7.2
```

```r
#### Check if numeric columns are indeed numeric
sapply(transaction_data, class)
```

```
##           DATE      STORE_NBR LYLTY_CARD_NBR         TXN_ID       PROD_NBR
##    "character"      "numeric"      "numeric"      "numeric"      "numeric"
##      PROD_NAME       PROD_QTY      TOT_SALES
##    "character"      "numeric"      "numeric"
```

```r
#### Display summary statistics
summary(transaction_data)
```

```
##      DATE              STORE_NBR     LYLTY_CARD_NBR        TXN_ID
##  Length:264836      Min.   :  1.0   Min.    :    1000   Min.    :       1
##  Class :character   1st Qu.: 70.0   1st Qu.:   70021   1st Qu.:   67602
##  Mode  :character   Median :130.0   Median : 130358   Median : 135138
##                     Mean   :135.1   Mean   : 135549   Mean   : 135158
##                     3rd Qu.:203.0   3rd Qu.: 203094   3rd Qu.: 202701
##                     Max.   :272.0   Max.   :2373711   Max.   :2415841
##     PROD_NBR       PROD_NAME            PROD_QTY          TOT_SALES
##  Min.   :  1.00   Length:264836      Min.   :  1.000   Min.   :  1.500
##  1st Qu.: 28.00   Class :character   1st Qu.:  2.000   1st Qu.:  5.400
##  Median : 56.00   Mode  :character   Median :  2.000   Median :  7.400
##  Mean   : 56.58                      Mean   :  1.907   Mean   :  7.304
##  3rd Qu.: 85.00                      3rd Qu.:  2.000   3rd Qu.:  9.200
##  Max.   :114.00                      Max.   :200.000   Max.   :650.000
```

```r
#### Convert DATE column to a date format
transaction_data$DATE <- as.Date(transaction_data$DATE, format = "%d/%m/%y")
```

Let's change this to a date format.

```r
#### Check the structure of the column
str(transaction_data$PROD_NAME)
```

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

```
##  chr [1:264836] "Natural Chip Compny SeaSalt175g" "CCs Nacho Cheese 175g" ...
```

```r
#### View the first few product names
head(transaction_data$PROD_NAME, 10)
```

```
##  [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"
##  [7] "Smiths Crinkle Chips Salt & Vinegar 330g"
##  [8] "Grain Waves Sweet Chilli 210g"
##  [9] "Doritos Corn Chip Mexican Jalapeno 150g"
## [10] "Grain Waves Sour Cream&Chives 210G"
```

```r
#### Count unique product names
num_unique_products <- length(unique(transaction_data$PROD_NAME))
num_unique_products
```

```
## [1] 114
```

```r
#### Display summary statistics
summary(transaction_data$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 summarizing the individual words in the product name.

```r
#### Ensure correct data set reference (adjust as needed)
productWords <- data.table(unlist(strsplit(unique(transaction_data$PROD_NAME), " ")))

#### Rename the column to 'words'
setnames(productWords, "words")

#### Count frequency of words
wordCounts <- productWords[, .N, by = words][order(-N)]

#### View the most common words
head(wordCounts, 20)
```

Further examine on PROD_NAME

```
##        words     N
##       <char> <int>
##  1:     175g    26
##  2:    Chips    21
##  3:     150g    19
##  4:        &    17
##  5:   Smiths    16
##  6:  Crinkle    14
##  7:      Cut    14
##  8:   Kettle    13
##  9:   Cheese    12
## 10:     Salt    12
## 11: Original    10
## 12:     Chip     9
## 13:    Salsa     9
## 14:  Doritos     9
## 15:     170g     8
## 16:     Corn     8
## 17: Pringles     8
## 18:     134g     8
## 19:     165g     8
## 20:      RRD     8
##        words     N
```

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.

```r
# Extract words from product names
productWords <- data.table(unlist(strsplit(unique(transaction_data$PROD_NAME), " ")))
setnames(productWords, "words")

# Remove words containing digits (0-9) or special characters ('&', '@', etc.)
cleanWords <- productWords[!grepl("[0-9&@#$%^*()_+=\\-]", words), ]

# Count frequency of cleaned words
```

```r
wordCounts <- cleanWords[, .N, by = words][order(-N)]

# Display the most common words after cleaning
head(wordCounts, 20)
```

Remove digits, and special characters, and then sort the distinct words by frequency of occurrence.

```
##        words     N
##       <char> <int>
##   1:     Chips    21
##   2:    Smiths    16
##   3:   Crinkle    14
##   4:       Cut    14
##   5:    Kettle    13
##   6:    Cheese    12
##   7:      Salt    12
##   8: Original    10
##   9:      Chip     9
##  10:     Salsa     9
##  11:   Doritos     9
##  12:      Corn     8
##  13: Pringles     8
##  14:       RRD     8
##  15:   Chicken     7
##  16:        WW     7
##  17:      Sour     6
##  18:       Sea     6
##  19:    Thinly     5
##  20:   Vinegar     5
##        words     N
```

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

```r
# Create a logical vector to identify products containing "salsa"
transaction_data$SALSA <- grepl("salsa", tolower(transaction_data$PROD_NAME))

# Remove rows where SALSA is TRUE (i.e., products containing "salsa")
transaction_data <- transaction_data[transaction_data$SALSA == FALSE, ]

# Drop the SALSA column since it's no longer needed
transaction_data$SALSA <- NULL

# Display the first few rows of the cleaned data set
head(transaction_data)
```

Remove salsa products

```
## # A tibble: 6 x 8
## DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR PROD_NAME PROD_QTY
## <date> <dbl> <dbl> <dbl> <dbl> <chr> <dbl>
## 1 2018-10-17 1 1000 1 5 Natural Chip Com~ 2
## 2 2019-05-14 1 1307 348 66 CCs Nacho Cheese~ 3
## 3 2019-05-20 1 1343 383 61 Smiths Crinkle C~ 2
## 4 2018-08-17 2 2373 974 69 Smiths Chip Thin~ 5
## 5 2018-08-18 2 2426 1038 108 Kettle Tortilla ~ 3
## 6 2019-05-16 4 4149 3333 16 Smiths Crinkle C~ 1
## # i 1 more variable: TOT_SALES <dbl>
```

There are no nulls in the columns, so lets check for possible outliers

```r
# Generate summary statistics for all columns
summary(transaction_data)
```

Summaries the data to check for possible outliers

```
##       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
```

```r
# Check for null values explicitly in each column
colSums(is.na(transaction_data))
```

```
##           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.

```r
# Identify the customer who purchased 200 packets of chips
customer_id <- transaction_data$LYLTY_CARD_NBR[transaction_data$PROD_QTY == 200]

# Filter transactions made by this customer
customer_transactions <- transaction_data[transaction_data$LYLTY_CARD_NBR == customer_id, ]

# View a summary of their transactions
summary(customer_transactions)
```

```
##       DATE              STORE_NBR    LYLTY_CARD_NBR       TXN_ID
##  Min.   :2018-08-19   Min.   :226   Min.   :226000   Min.   :226201
##  1st Qu.:2018-10-26   1st Qu.:226   1st Qu.:226000   1st Qu.:226203
##  Median :2019-01-03   Median :226   Median :226000   Median :226206
```

```
##  Mean   :2019-01-03   Mean   :226   Mean   :226000   Mean   :226206
##  3rd Qu.:2019-03-12   3rd Qu.:226   3rd Qu.:226000   3rd Qu.:226208
##  Max.   :2019-05-20   Max.   :226   Max.   :226000   Max.   :226210
##     PROD_NBR  PROD_NAME             PROD_QTY      TOT_SALES
##  Min.   :4   Length:2           Min.   :200   Min.   :650
##  1st Qu.:4   Class :character   1st Qu.:200   1st Qu.:650
##  Median :4   Mode  :character   Median :200   Median :650
##  Mean   :4                      Mean   :200   Mean   :650
##  3rd Qu.:4                      3rd Qu.:200   3rd Qu.:650
##  Max.   :4                      Max.   :200   Max.   :650
```

```r
# Display sample transactions
head(customer_transactions)
```

```
## # A tibble: 2 x 8
## DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR PROD_NAME PROD_QTY
## <date> <dbl> <dbl> <dbl> <dbl> <chr> <dbl>
## 1 2018-08-19 226 226000 226201 4 Dorito Corn Chp ~ 200
## 2 2019-05-20 226 226000 226210 4 Dorito Corn Chp ~ 200
## # i 1 more variable: TOT_SALES <dbl>
```

There are two transactions where 200 packets of chips are bought in one transaction and both of these transactions were by the same customer. 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
# Identify the customer with bulk purchases
customer_id <- transaction_data$LYLTY_CARD_NBR[transaction_data$PROD_QTY == 200]

# Remove all transactions from this customer
filtered_transaction_data <- transaction_data[transaction_data$LYLTY_CARD_NBR != customer_id, ]

# Re-examine the dataset after filtering
summary(filtered_transaction_data)
```

Filter out the customer based on the loyalty card number

```
##       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
```

```r
# Display the first few rows
head(filtered_transaction_data)
```

```
## # A tibble: 6 x 8
## DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR PROD_NAME PROD_QTY
## <date> <dbl> <dbl> <dbl> <dbl> <chr> <dbl>
## 1 2018-10-17 1 1000 1 5 Natural Chip Com~ 2
## 2 2019-05-14 1 1307 348 66 CCs Nacho Cheese~ 3
## 3 2019-05-20 1 1343 383 61 Smiths Crinkle C~ 2
## 4 2018-08-17 2 2373 974 69 Smiths Chip Thin~ 5
## 5 2018-08-18 2 2426 1038 108 Kettle Tortilla ~ 3
## 6 2019-05-16 4 4149 3333 16 Smiths Crinkle C~ 1
## # i 1 more variable: TOT_SALES <dbl>
```

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 transactions for each unique date
transaction_summary <- aggregate(TXN_ID ~ DATE, data = transaction_data, FUN = length)
```

```r
# Rename columns for clarity
colnames(transaction_summary) <- c("Date", "Transaction_Count")

# View summary of transactions per date
summary(transaction_summary)
```

Count Transaction by date

```
##       Date            Transaction_Count
##  Min.   :2018-07-01   Min.   :607.0
##  1st Qu.:2018-09-29   1st Qu.:658.0
##  Median :2018-12-30   Median :674.0
##  Mean   :2018-12-30   Mean   :677.9
##  3rd Qu.:2019-03-31   3rd Qu.:694.2
##  Max.   :2019-06-30   Max.   :865.0
```

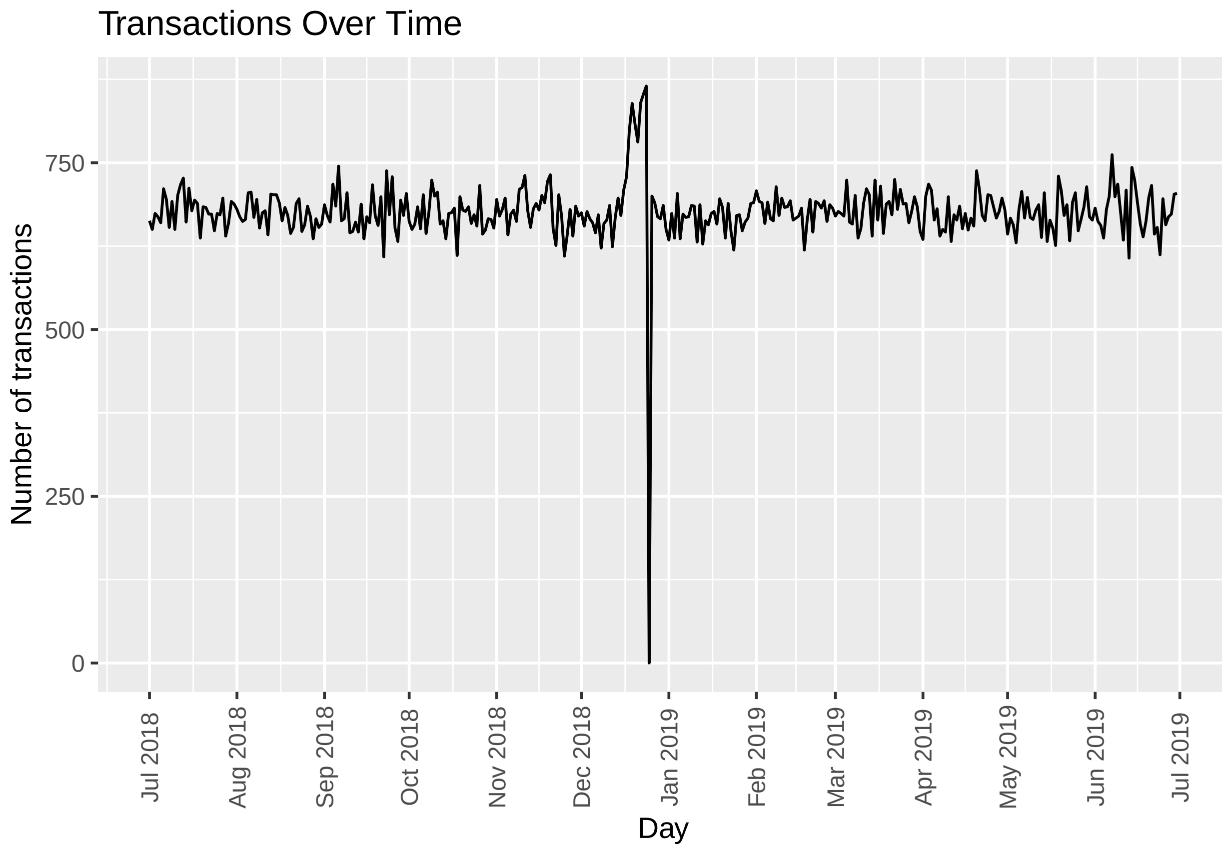
```r
# Display the first few rows
head(transaction_summary)
```

```
##         Date Transaction_Count
## 1 2018-07-01               663
## 2 2018-07-02               650
## 3 2018-07-03               674
## 4 2018-07-04               669
## 5 2018-07-05               660
## 6 2018-07-06               711
```

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.

Created a sequence of dates and join this the count of transactions by date

```
## [1] "2018-12-25"
```

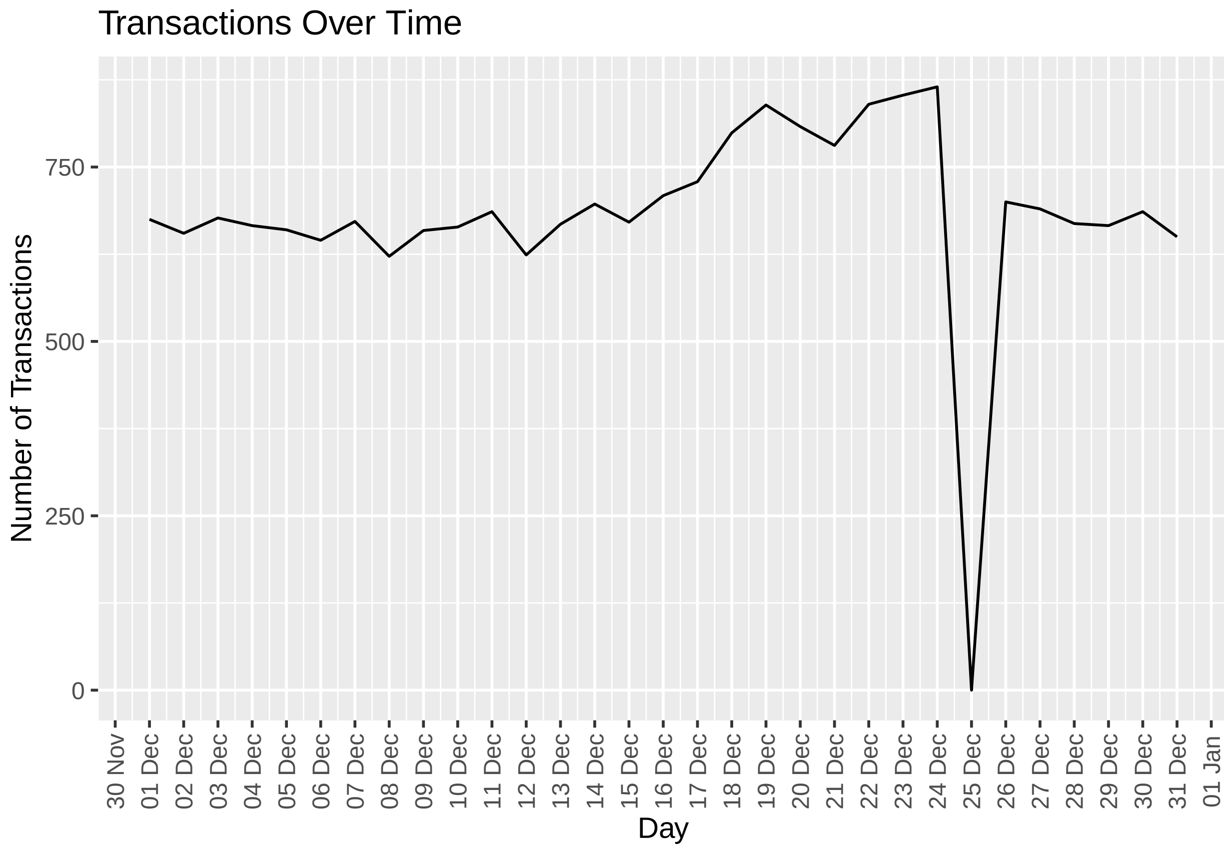



```
##         DATE Transaction_Count
## 1 2018-07-01               663
## 2 2018-07-02               650
## 3 2018-07-03               674
## 4 2018-07-04               669
## 5 2018-07-05               660
## 6 2018-07-06               711
```

```
##       DATE            Transaction_Count
##  Min.   :2018-07-01   Min.   :607.0
##  1st Qu.:2018-09-29   1st Qu.:658.0
##  Median :2018-12-30   Median :674.0
##  Mean   :2018-12-30   Mean   :677.9
##  3rd Qu.:2019-03-31   3rd Qu.:694.2
##  Max.   :2019-06-30   Max.   :865.0
```

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

Filtered to December and look at individual days



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.

```r
transaction_data$PACK_SIZE <- parse_number(transaction_data$PROD_NAME)

# Create a frequency table of PACK_SIZE
pack_summary <- as.data.frame(table(transaction_data$PACK_SIZE))

# Rename the columns for readability
names(pack_summary) <- c("PACK_SIZE", "Frequency")

# Convert PACK_SIZE from factor to numeric, if necessary
pack_summary$PACK_SIZE <- as.numeric(as.character(pack_summary$PACK_SIZE))
```

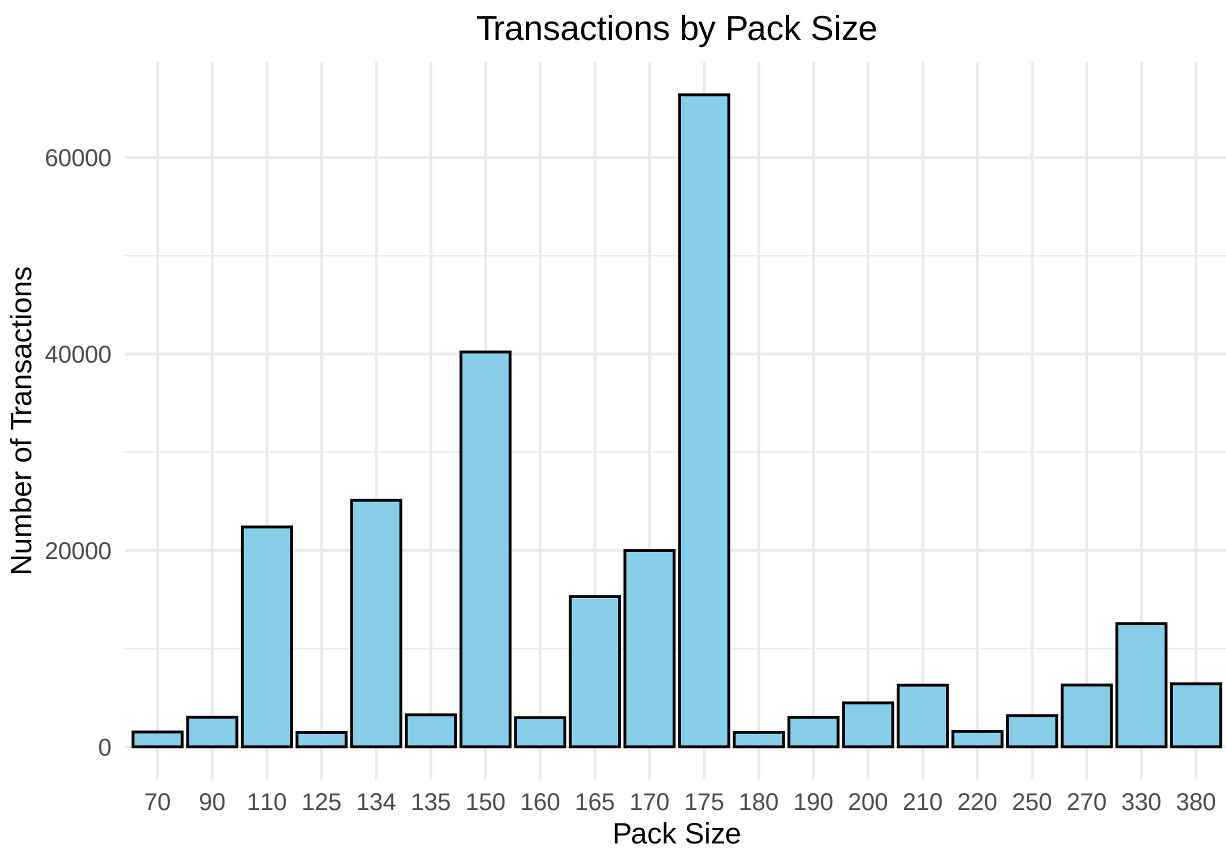
```r
# Order the summary by pack size
pack_summary <- pack_summary[order(pack_summary$PACK_SIZE), ]

# Print the summary to check if the pack sizes look sensible
print(pack_summary)
```

created a new column 'PACK_SIZE' by extracting the first numeric value from PROD_NAME

```
##    PACK_SIZE Frequency
## 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      6418
```

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

Created 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.



Pack sizes created look reasonable.

```r
# Create the BRAND column by extracting the first word from PROD_NAME
transaction_data$BRAND <- sub(" .*", "", transaction_data$PROD_NAME)

# Check the results by creating a frequency table of the brands
brand_summary <- as.data.frame(table(transaction_data$BRAND))
names(brand_summary) <- c("BRAND", "Frequency")

# Order by frequency (descending) to see the most common brands first
brand_summary <- brand_summary[order(-brand_summary$Frequency), ]

# Print the brand summary
print(brand_summary)
```

Created brands, we can use the first word in PROD_NAME to work out the brand name

```
##          BRAND Frequency
## 13      Kettle     41288
## 20      Smiths     27390
## 16    Pringles     25102
## 7      Doritos     22041
## 23       Thins     14075
## 18         RRD     11894
## 11   Infuzions     11057
## 28          WW     10320
## 5         Cobs      9693
## 24    Tostitos      9471
## 25    Twisties      9454
## 26    Tyrrells      6442
## 9        Grain      6272
## 14     Natural      6050
## 4     Cheezels      4603
## 2          CCs      4551
## 17         Red      4427
## 6       Dorito      3185
## 12      Infzns      3144
## 19       Smith      2963
## 3      Cheetos      2927
## 21       Snbts      1576
## 1       Burger      1564
## 27  Woolworths      1516
## 10     GrnWves      1468
## 22    Sunbites      1432
## 15         NCC      1419
## 8       French      1418
```

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: for example, change "RED" to "RRD"
transaction_data$BRAND[transaction_data$BRAND == "Red"] <- "RRD"
```

```r
transaction_data$BRAND[transaction_data$BRAND =="SNBTS"] <-"SUNBITES"
transaction_data$BRAND[transaction_data$BRAND =="INFZNS"]<- "INFUZIONS"
transaction_data$BRAND[transaction_data$BRAND =="WW"]<-"WOOLWORTHS"
transaction_data$BRAND[transaction_data$BRAND =="SMITH"]<- "SMITHS"
transaction_data$BRAND[transaction_data$BRAND =="NCC"]<- "NATURAL"
transaction_data$BRAND[transaction_data$BRAND =="DORITO"]<- "DORITOS"
transaction_data$BRAND[transaction_data$BRAND =="GRAIN"]<- "GRNWVES"

# Check the cleaned results by creating a frequency table of the brands
brand_summary <- as.data.frame(table(transaction_data$BRAND))
names(brand_summary) <- c("BRAND", "Frequency")

# Order the summary by frequency (optional: descending order)
brand_summary <- brand_summary[order(-brand_summary$Frequency), ]

# Print the summary to see if the brand names now look reasonable
print(brand_summary)
```

Cleaned and examined BRAND names

```
##          BRAND Frequency
## 13      Kettle     41288
## 19      Smiths     27390
## 16    Pringles     25102
## 7      Doritos     22041
## 17         RRD     16321
## 22       Thins     14075
## 11   Infuzions     11057
## 27 WOOLWORTHS     10320
## 5         Cobs      9693
## 23    Tostitos      9471
## 24    Twisties      9454
## 25    Tyrrells      6442
## 9        Grain      6272
## 14     Natural      6050
## 4     Cheezels      4603
## 2          CCs      4551
```

```
## 6      Dorito      3185
## 12     Infzns      3144
## 18      Smith      2963
## 3     Cheetos      2927
## 20      Snbts      1576
## 1      Burger      1564
## 26 Woolworths      1516
## 10    GrnWves      1468
## 21   Sunbites      1432
## 15    NATURAL      1419
## 8      French      1418
```

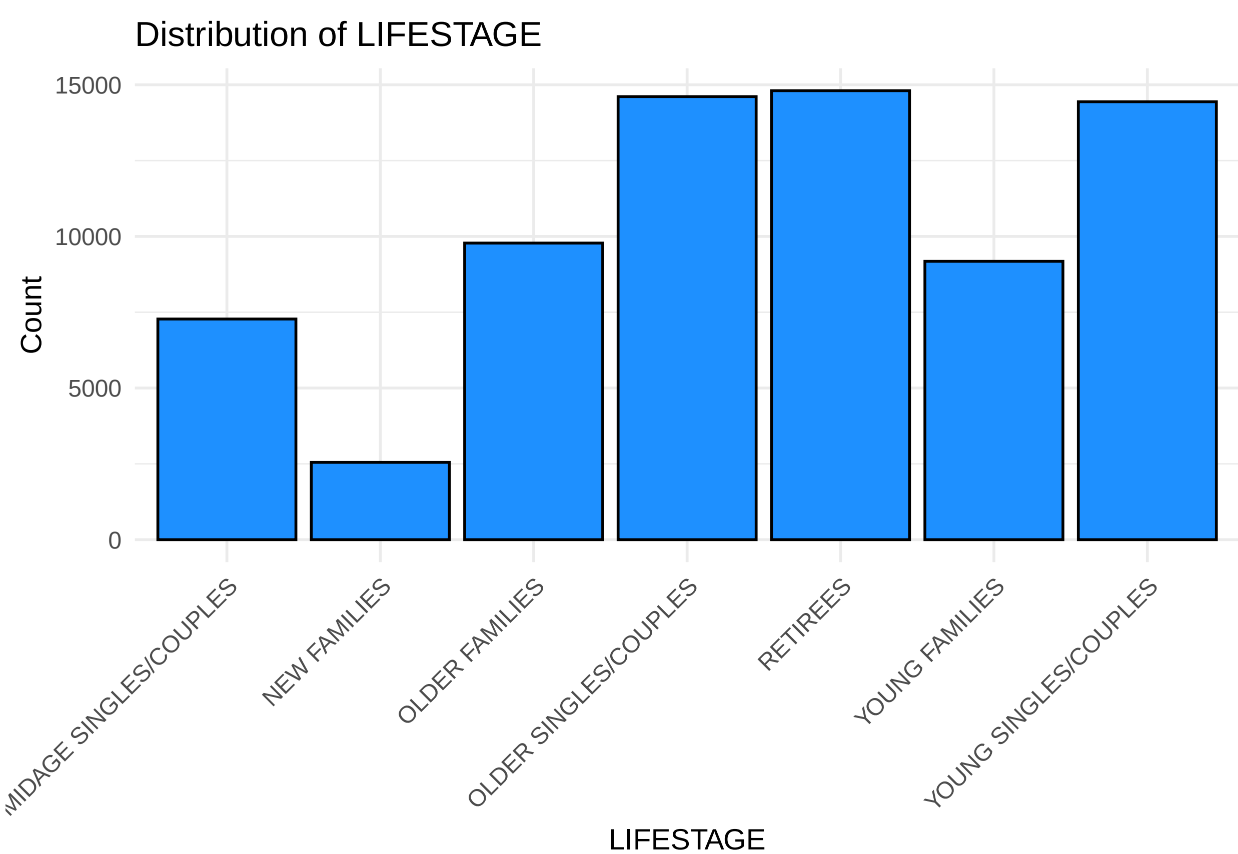
Now that i am satisfied with the transaction data set, let's have a look at the customer data set.

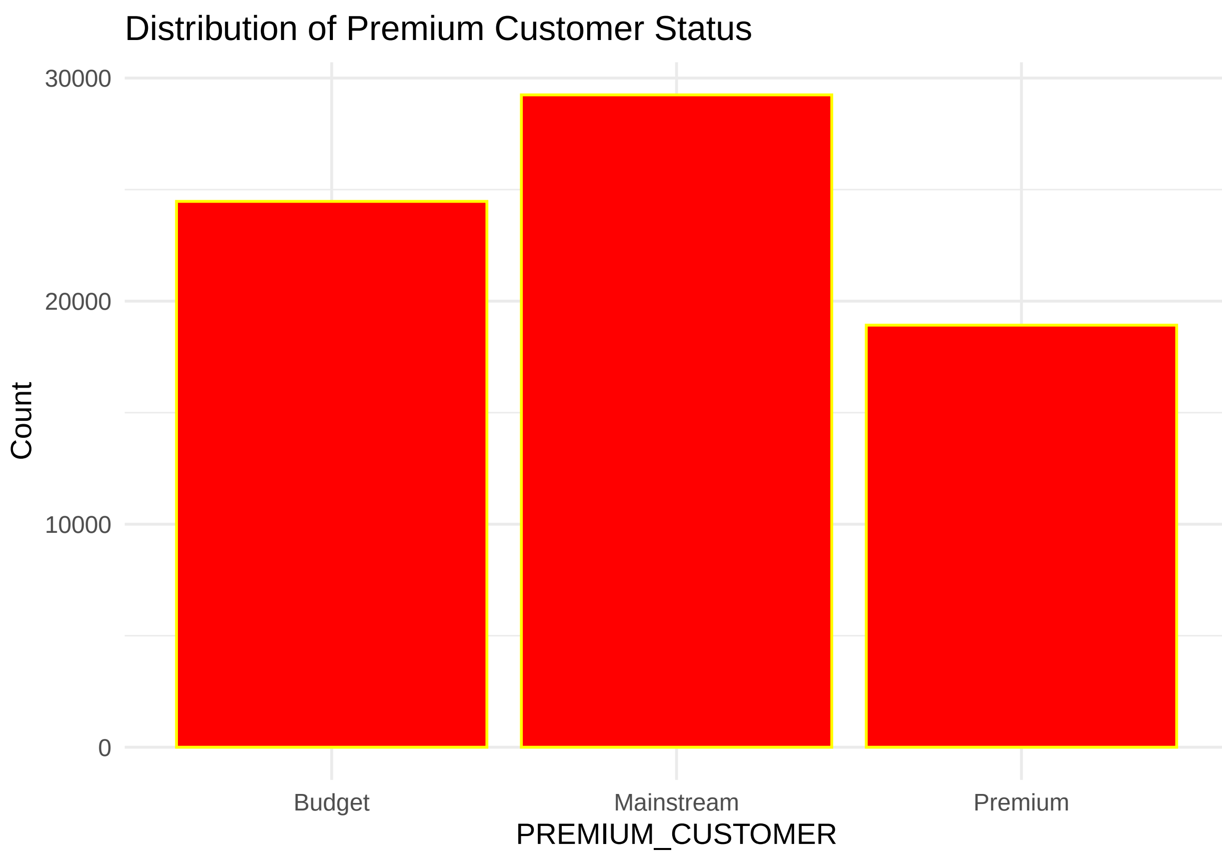
Examining Purchase_behaviour data

```
## spc_tbl_ [72,637 x 3] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
## $ LYLTY_CARD_NBR : num [1:72637] 1000 1002 1003 1004 1005 ...
## $ LIFESTAGE : chr [1:72637] "YOUNG SINGLES/COUPLES" "YOUNG SINGLES/COUPLES"
"YOUNG FAMILIES" "OLDER SINGLES/COUPLES" ...
## $ PREMIUM_CUSTOMER: chr [1:72637] "Premium" "Mainstream" "Budget"
"Mainstream" ...
## - attr(*, "spec")=
## .. cols(
## ..  LYLTY_CARD_NBR = col_double(),
## ..  LIFESTAGE = col_character(),
## ..  PREMIUM_CUSTOMER = col_character()
## .. )
## - attr(*, "problems")=<externalptr>
```

```
##  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
```

```
## Frequency distribution of LIFESTAGE:
##
## MIDAGE SINGLES/COUPLES           NEW FAMILIES         OLDER FAMILIES
##                   7275                   2549                   9780
##  OLDER SINGLES/COUPLES               RETIREES         YOUNG FAMILIES
##                  14609                  14805                   9178
##  YOUNG SINGLES/COUPLES
##                  14441

##
## Frequency distribution of PREMIUM_CUSTOMER:
##
##     Budget Mainstream    Premium
##      24470      29245      18922

##
## Number of unique loyalty card numbers:
## [1] 72637
```





Satisfied with the purchase_behaviour data set

Merge transaction_data to purchase_behaviour

```
#### Merge transaction data to customer data
data <- merge(transaction_data, purchase_behaviour, by = "LYLTY_CARD_NBR", all.x = TRUE)
```

```
# Check for missing customer details in merged dataset
missing_customers <- data[is.na(data$LIFESTAGE) | is.na(data$PREMIUM_CUSTOMER), ]

# Print summary of missing records
cat("Number of transactions without a matched customer:", nrow(missing_customers), "\n")
```

```
## Number of transactions without a matched customer: 0
```

```r
# Optionally, view the first few rows of missing records
if(nrow(missing_customers) > 0) {
  print(head(missing_customers))
}
```

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

Saved file for further analysis

```r
fwrite(data, "C:/Users/user/Desktop/QVI_data.csv")

file.exists("C:/Users/user/Desktop/QVI_data.csv")
```

```
## [1] TRUE
```

Data exploration is now complete!

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

```
##
## 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
```

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

```
## [1] 27
```

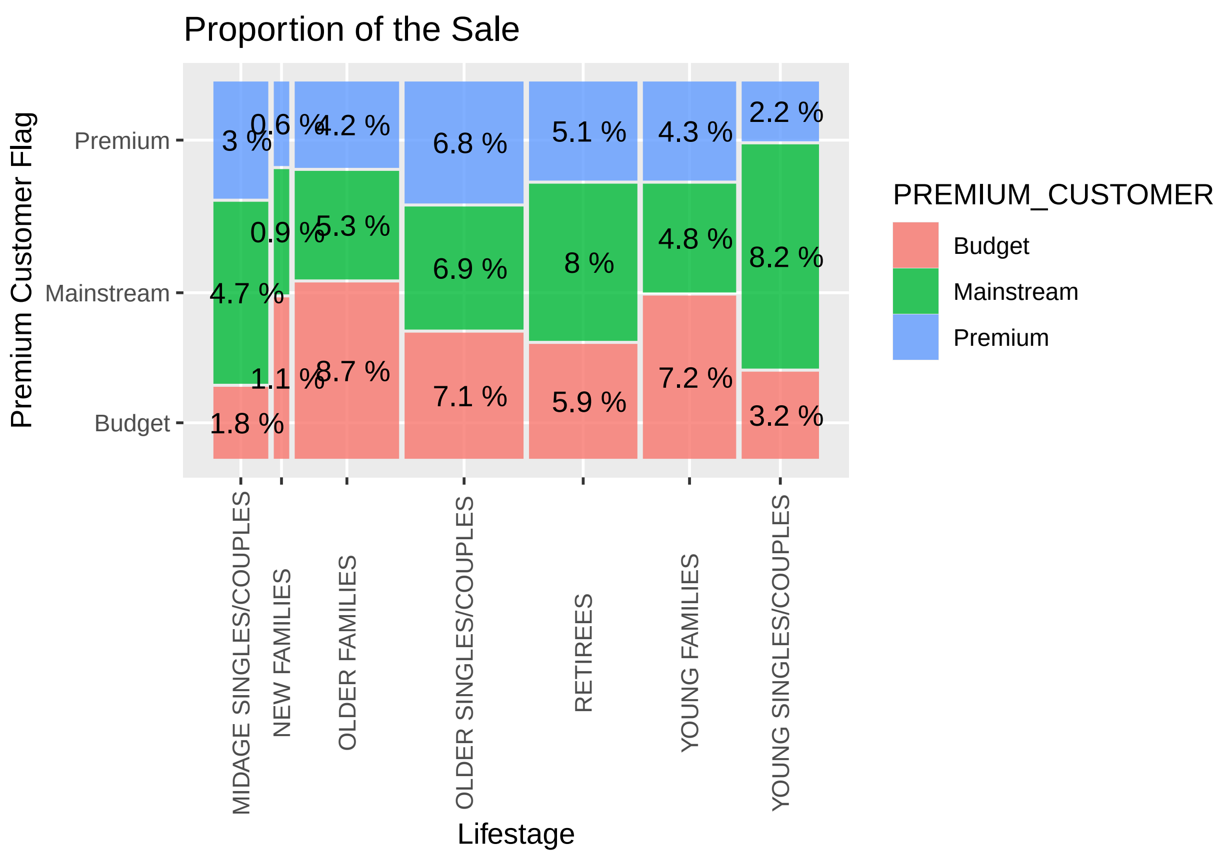
```
## # A tibble: 21 x 3
## # Groups:   LIFESTAGE [7]
##    LIFESTAGE              PREMIUM_CUSTOMER Total_Sales
##    <chr>                  <chr>                  <dbl>
##  1 OLDER FAMILIES         Budget               156864.
##  2 YOUNG SINGLES/COUPLES  Mainstream           147582.
##  3 RETIREES               Mainstream           145169.
##  4 YOUNG FAMILIES         Budget               129718.
##  5 OLDER SINGLES/COUPLES  Budget               127834.
##  6 OLDER SINGLES/COUPLES  Mainstream           124648.
##  7 OLDER SINGLES/COUPLES  Premium              123538.
##  8 RETIREES               Budget               105916.
##  9 OLDER FAMILIES         Mainstream            96414.
## 10 RETIREES               Premium               91297.
## # i 11 more rows
```

```
## Warning: The `scale_name` argument of `continuous_scale()` is deprecated as of ggplot2
## 3.5.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Warning: The `trans` argument of `continuous_scale()` is deprecated as of ggplot2 3.5.0.
## i Please use the `transform` argument instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Warning: `unite_()` was deprecated in tidyr 1.2.0.
## i Please use `unite()` instead.
## i The deprecated feature was likely used in the ggmosaic package.
##   Please report the issue at <https://github.com/haleyjeppson/ggmosaic>.
```

```
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

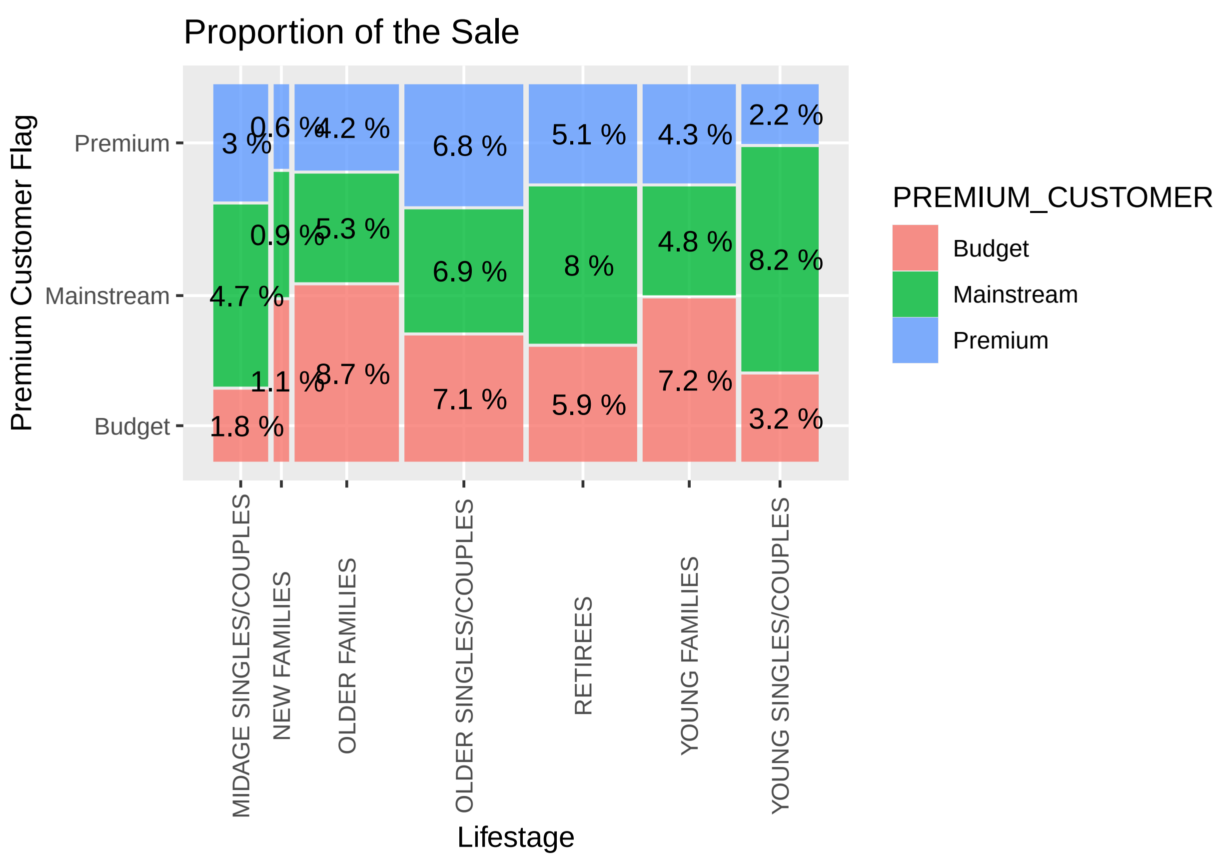


Sales are coming mainly from Budget - older families, Mainstream - young singles/couples, and Mainstream - retirees

Lets's see if the higher sales are due to there being more customers who buy chips.

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

## # A tibble: 21 x 3
## # Groups:   LIFESTAGE [7]
##    LIFESTAGE              PREMIUM_CUSTOMER Number_of_Customers
##    <chr>                  <chr>                          <int>
```

```
##    1 YOUNG SINGLES/COUPLES Mainstream                          7917
##    2 RETIREES              Mainstream                          6358
##    3 OLDER SINGLES/COUPLES Mainstream                          4858
##    4 OLDER SINGLES/COUPLES Budget                              4849
##    5 OLDER SINGLES/COUPLES Premium                             4682
##    6 OLDER FAMILIES        Budget                              4611
##    7 RETIREES              Budget                              4385
##    8 YOUNG FAMILIES        Budget                              3953
##    9 RETIREES              Premium                             3812
##   10 YOUNG SINGLES/COUPLES Budget                              3647
## # i 11 more rows
```



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.

Average number of units per customer by LIFESTAGE and PREMUIM_CUSTOMER

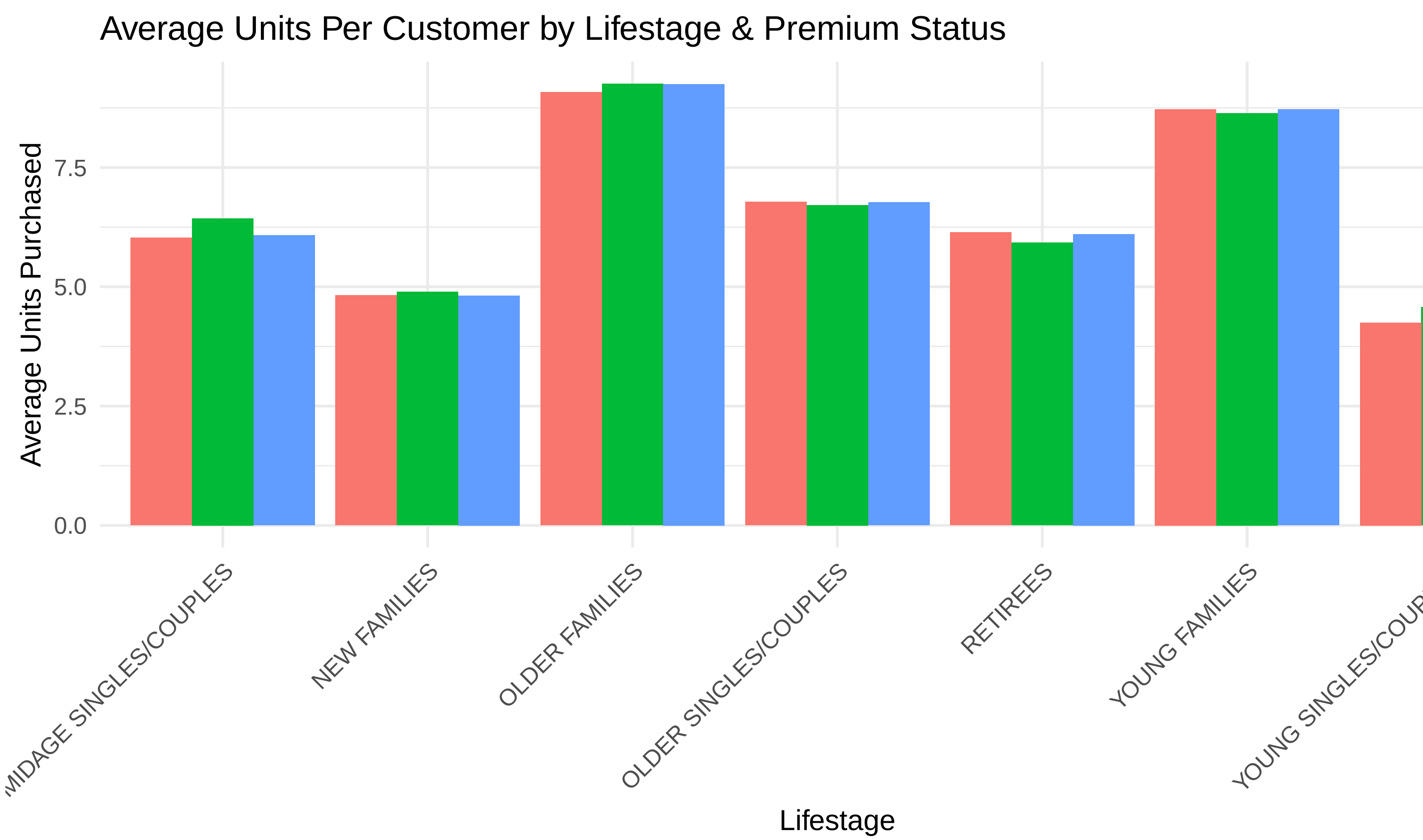
```r
# Your ggplot code here}
library(dplyr)
library(ggplot2)



# Calculate the average number of units per customer by LIFESTAGE & PREMIUM_CUSTOMER
avg_units_per_customer <- data %>%
  group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarise(Average_Units = sum(PROD_QTY) / n_distinct(LYLTY_CARD_NBR)) %>%
  arrange(desc(Average_Units))
```

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

```r
# Print summary table
print(avg_units_per_customer)
```

```
## # A tibble: 21 x 3
## # Groups:   LIFESTAGE [7]
##    LIFESTAGE              PREMIUM_CUSTOMER Average_Units
##    <chr>                  <chr>                    <dbl>
##  1 OLDER FAMILIES         Mainstream                9.26
##  2 OLDER FAMILIES         Premium                   9.25
##  3 OLDER FAMILIES         Budget                    9.08
##  4 YOUNG FAMILIES         Budget                    8.72
##  5 YOUNG FAMILIES         Premium                   8.72
##  6 YOUNG FAMILIES         Mainstream                8.64
##  7 OLDER SINGLES/COUPLES  Budget                    6.78
##  8 OLDER SINGLES/COUPLES  Premium                   6.77
##  9 OLDER SINGLES/COUPLES  Mainstream                6.71
## 10 MIDAGE SINGLES/COUPLES Mainstream                6.43
## # i 11 more rows
```

```
# Create a bar plot to visualize average units per customer by segment
ggplot(data = avg_units_per_customer, aes(x = LIFESTAGE, y = Average_Units, fill = PREMIUM_CUSTOMER)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  labs(title = "Average Units Per Customer by Lifestage & Premium Status",
       x = "Lifestage",
       y = "Average Units Purchased",
       fill = "Premium Customer") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



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

Average price per unit sold by LIFESTAGE and PREMIUM_CUSTOMER

```r
# Calculate the average price per unit sold by LIFESTAGE & PREMIUM_CUSTOMER
avg_price_per_unit <- data %>%
  group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarise(Average_Price = sum(TOT_SALES) / sum(PROD_QTY)) %>%
  arrange(desc(Average_Price))
```

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

```r
# Print summary table
print(avg_price_per_unit)
```

```
## # A tibble: 21 x 3
## # Groups:   LIFESTAGE [7]
##    LIFESTAGE              PREMIUM_CUSTOMER Average_Price
##    <chr>                  <chr>                    <dbl>
##  1 YOUNG SINGLES/COUPLES  Mainstream                4.07
##  2 MIDAGE SINGLES/COUPLES Mainstream                3.99
##  3 NEW FAMILIES           Mainstream                3.94
##  4 RETIREES               Budget                    3.93
##  5 NEW FAMILIES           Budget                    3.93
##  6 RETIREES               Premium                   3.92
##  7 OLDER SINGLES/COUPLES  Premium                   3.90
##  8 OLDER SINGLES/COUPLES  Budget                    3.89
##  9 NEW FAMILIES           Premium                   3.89
## 10 RETIREES               Mainstream                3.85
## # i 11 more rows
```

```r
# Create a bar plot to visualize the average price per unit by customer segment
ggplot(avg_price_per_unit, aes(x = LIFESTAGE, y = Average_Price, fill = PREMIUM_CUSTOMER)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  labs(title = "Average unit Per price by Lifestage & Premium Status",
       x = "Lifestage",
       y = "Average Price Per Unit",
       fill = "Premium Customer") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

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

```
library(dplyr)

#### Perform an independent t-test between mainstream vs premium and budgetmidage andyoung singles and

# Calculate price per unit
library(dplyr)
```

```r
library(data.table)
# Ensure QVI_data is a data.table
setDT(data)

# Calculate price per unit
price_per_unit <- data[, price := TOT_SALES / PROD_QTY]

# Perform t-test comparing Mainstream vs Premium/Budget in specified life stages
t.test(
  data[LIFESTAGE %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES") & PREMIUM_CUSTOMER == "Mains
  data[LIFESTAGE %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES") & PREMIUM_CUSTOMER != "Mains
  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
```

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.

```
##Step 1: Prepare Data for Analysis, we need to filter data to Mainstream - Young Singles/Couples and e

library(dplyr)

 #### Deep dive into Mainstream, young singles/couples
library(data.table)

# Segment Mainstream Young Singles/Couples
segment1 <- data[LIFESTAGE == "YOUNG SINGLES/COUPLES" & PREMIUM_CUSTOMER == "Mainstream"]

# Segment Other Customers
other <- data[!(LIFESTAGE == "YOUNG SINGLES/COUPLES" & PREMIUM_CUSTOMER == "Mainstream")]

# Calculate total quantity per segment
quantity_segment1 <- sum(segment1$PROD_QTY)
quantity_other <- sum(other$PROD_QTY)

# Brand affinity calculations
quantity_segment1_by_brand <- segment1[, .(targetSegment = sum(PROD_QTY) / quantity_segment1), by = BRA
quantity_other_by_brand <- other[, .(other = sum(PROD_QTY) / quantity_other), by = BRAND]

# Merge data to compare brand affinity
brand_proportions <- merge(quantity_segment1_by_brand, quantity_other_by_brand, by = "BRAND")

# Calculate affinity ratio
brand_proportions[, affinityToBrand := targetSegment / other]

# Sort brands by affinity
brand_proportions[order(-affinityToBrand)]
```

```
##          BRAND targetSegment       other affinityToBrand
```

```
##          <char>        <num>       <num>          <num>
##  1:   Tyrrells   0.031552795 0.025668816      1.2292267
##  2:   Twisties   0.046183575 0.037841656      1.2204427
##  3:    Doritos   0.107053140 0.088233534      1.2132931
##  4:     Kettle   0.197984817 0.165401059      1.1969985
##  5:   Tostitos   0.045410628 0.037942905      1.1968147
##  6:     Infzns   0.014934438 0.012561727      1.1888841
##  7:   Pringles   0.119420290 0.100542140      1.1877636
##  8:      Grain   0.029123533 0.025098142      1.1603860
##  9:     Dorito   0.015707384 0.013668558      1.1491618
## 10:       Cobs   0.044637681 0.039012918      1.1441769
## 11:  Infuzions   0.049744651 0.044450427      1.1191040
## 12:      Thins   0.060372671 0.056933917      1.0603990
## 13:   Cheezels   0.017971014 0.018629739      0.9646413
## 14:     Smiths   0.089772257 0.112112091      0.8007366
## 15:     French   0.003947550 0.005752760      0.6862010
## 16:    Cheetos   0.008033126 0.012055484      0.6663462
## 17:        RRD   0.043809524 0.067431554      0.6496888
## 18:    Natural   0.015955832 0.024957775      0.6393131
## 19:    NATURAL   0.003643892 0.005867815      0.6209964
## 20:        CCs   0.011180124 0.018878258      0.5922222
## 21:    GrnWves   0.003588682 0.006061108      0.5920835
## 22:      Smith   0.006597654 0.012356929      0.5339234
## 23:      Snbts   0.003478261 0.006581158      0.5285181
## 24: WOOLWORTHS   0.021256039 0.043009936      0.4942123
## 25:   Sunbites   0.002870945 0.005987473      0.4794920
## 26: Woolworths   0.002843340 0.006371757      0.4462411
## 27:     Burger   0.002926156 0.006590362      0.4440053
##           BRAND targetSegment       other affinityToBrand
```

We can see that : • Mainstream young singles/couples are 23% 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.

```r
#### Preferred pack size compared to the rest of the population
library(data.table)

# Ensure 'segment1' and 'other' are data.tables
setDT(segment1)
setDT(other)

# Calculate pack proportions
quantity_segment1_by_pack <- segment1[, .(targetSegment = sum(PROD_QTY) / quantity_segment1), by = PACK_
quantity_other_by_pack <- other[, .(other = sum(PROD_QTY) / quantity_other), by = PACK_SIZE]

# Merge data and compute affinity score
pack_proportions <- merge(quantity_segment1_by_pack, quantity_other_by_pack, by = "PACK_SIZE")

# Calculate affinity to pack size
pack_proportions[, affinityToPack := targetSegment / other]

# Sort results by affinity descending
pack_proportions[order(-affinityToPack)]
```

```
##     PACK_SIZE targetSegment       other affinityToPack
##        <fctr>         <num>       <num>          <num>
##  1:       270   0.031828847 0.025072830      1.2694557
##  2:       330   0.061283644 0.050115746      1.2228421
##  3:       380   0.032160110 0.026481106      1.2144550
##  4:       134   0.119420290 0.100542140      1.1877636
##  5:       110   0.106280193 0.089708542      1.1847277
##  6:       210   0.029123533 0.025098142      1.1603860
##  7:       135   0.014768806 0.013063368      1.1305512
##  8:       250   0.014354727 0.012768826      1.1242010
##  9:       170   0.080772947 0.080911421      0.9982886
## 10:       150   0.157598344 0.163270237      0.9652607
## 11:       175   0.254989648 0.269758430      0.9452518
## 12:       165   0.055652174 0.062210349      0.8945806
## 13:       190   0.007481021 0.012430564      0.6018248
## 14:       180   0.003588682 0.006061108      0.5920835
```

```
## 15:       160   0.006404417 0.012361531      0.5180925
## 16:        90   0.006349206 0.012568630      0.5051629
## 17:       125   0.003008972 0.006031194      0.4989015
## 18:       200   0.008971705 0.018638943      0.4813419
## 19:        70   0.003036577 0.006316531      0.4807349
## 20:       220   0.002926156 0.006590362      0.4440053
##     PACK_SIZE targetSegment       other affinityToPack
```

It looks like Mainstream young singles/couples are 27% 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

Let's recap what we've found!

Sales have mainly been due to Budget-older families,Mainstream-youngsingles/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 youngsingles and couples are 23%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 discretionary space near segments where young singles and couples frequent more often to increase visibility and impulse behaviour.

Quantium can help the Category Manager with recommendations of where these segments are and further help them with measuring the impact of the changed placement. We'll work on measuring the impact of trials in the next task and putting all these together in the third task