
#Title: "Chips Retails Strategy"

#Author: "Gbadebo Taiwo Samuel"

#Date: "2024-09-06"

#output: html_document

---

installing and loading the required packages

```
# This R environment comes with many helpful analytics packages installed
# It is defined by the kaggle/rstats Docker image: https://github.com/kaggle/docker-rstats
# For example, here's a helpful package to load

install.packages('data.table')
install.packages('ggmosaic')
install.packages('tidyverse')
library('data.table')
library('ggplot2')
library('ggmosaic')
library('readr')
library(tidyverse)
# metapackage of all tidyverse packages

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

list.files(path = "/kaggle/input/retail-strategy-data/QVI_purchase_behaviour1.csv")
list.files(path = "/kaggle/input/retail-strategy-data/QVI_transaction_data1.csv")

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save &
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

also installing the dependencies ‘plyr’, ‘lazyeval’, ‘crosstalk’, ‘productplots’, ‘plotly’, ‘ggrepel’


Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
✓ dplyr     1.1.4     ✓ stringr   1.5.1
✓ forcats   1.0.0     ✓ tibble    3.2.1
✓ lubridate 1.9.3     ✓ tidyr     1.3.1
✓ purrr     1.0.2
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::between()     masks data.table::between()
✖ dplyr::filter()      masks stats::filter()
✖ dplyr::first()       masks data.table::first()
✖ lubridate::hour()    masks data.table::hour()
✖ lubridate::isoweek() masks data.table::isoweek()
✖ dplyr::lag()         masks stats::lag()
✖ dplyr::last()        masks data.table::last()
✖ lubridate::mday()    masks data.table::mday()
✖ lubridate::minute()  masks data.table::minute()
✖ lubridate::month()   masks data.table::month()
✖ lubridate::quarter() masks data.table::quarter()
✖ lubridate::second()  masks data.table::second()
✖ purrr::transpose()   masks data.table::transpose()
✖ lubridate::wday()    masks data.table::wday()
✖ lubridate::week()    masks data.table::week()
✖ lubridate::yday()    masks data.table::yday()
✖ lubridate::year()    masks data.table::year()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

loading the dataset for cleaning and exploration

```
transactiondata <- read.csv("/content/QVI_transaction_data.csv")
customerdata <- read.csv("/content/QVI_purchase_behaviour.csv")
```

Dataset summary

loading the dataset for inspection of the data to be worked on.

```
#transactiondata summary
str(transactiondata)
```

```
'data.frame':   264836 obs. of  8 variables:
 $ DATE          : int  43390 43599 43605 43329 43330 43604 43601 43601 43332 43330 ...
 $ STORE_NBR     : int  1 1 1 2 2 4 4 4 5 7 ...
 $ LYLTY_CARD_NBR: int  1000 1307 1343 2373 2426 4074 4149 4196 5026 7150 ...
 $ TXN_ID        : int  1 348 383 974 1038 2982 3333 3539 4525 6900 ...
 $ PROD_NBR      : int  5 66 61 69 108 57 16 24 42 52 ...
 $ PROD_NAME     : chr  "Natural Chip        Compny SeaSalt175g" "CCs Nacho Cheese    175g" "Smiths Crinkle Cut  Chips Chicken 170g" "Sm
 $ PROD_QTY      : int  2 3 2 5 3 1 1 1 1 2 ...
 $ TOT_SALES     : num  6 6.3 2.9 15 13.8 5.1 5.7 3.6 3.9 7.2 ...
```

```
#customerdata summary
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" ...
```

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

```
 if (is.list(transactiondata$DATE)) {transactiondata$DATE <- unlist(transactiondata$DATE)}
transactiondata$DATE <- as.numeric(transactiondata$DATE)
transactiondata$DATE <- as.Date(transactiondata$DATE, origin = "1899-12-30")
str(transactiondata)
```

```
'data.frame':   264836 obs. of  8 variables:
 $ DATE          : Date, format: "2018-10-17" "2019-05-14" ...
 $ STORE_NBR     : int  1 1 1 2 2 4 4 4 5 7 ...
 $ LYLTY_CARD_NBR: int  1000 1307 1343 2373 2426 4074 4149 4196 5026 7150 ...
 $ TXN_ID        : int  1 348 383 974 1038 2982 3333 3539 4525 6900 ...
 $ PROD_NBR      : int  5 66 61 69 108 57 16 24 42 52 ...
 $ PROD_NAME     : chr  "Natural Chip        Compny SeaSalt175g" "CCs Nacho Cheese    175g" "Smiths Crinkle Cut  Chips Chicken 170g" "Sm
 $ PROD_QTY      : int  2 3 2 5 3 1 1 1 1 2 ...
 $ TOT_SALES     : num  6 6.3 2.9 15 13.8 5.1 5.7 3.6 3.9 7.2 ...
```

```
#inspecting the data set for anomalies
summary(transactiondata$PROD_NAME)
```

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

```
head(transactiondata$PROD_NAME)
```

```
'Natural Chip        Compny SeaSalt175g' · 'CCs Nacho Cheese    175g' · 'Smiths Crinkle Cut  Chips Chicken 170g' · 'Smiths Chip Thinly  S/Cream&Onion 175g' ·
'Kettle Tortilla ChpsHny&Jlpno Chili 150g' · 'Old El Paso Salsa   Dip Tomato Mild 300g'
```

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()`.

```
productWords <- strsplit(transactiondata$PROD_NAME, " ")
productWords <- unlist(productWords)
productWordsdata <- data.table(words = productWords)
wordSummary <- productWordsdata[, .N, by = words][order(-N)]
print(wordSummary)
```

```
          words      N
         <char>  <int>
      1:           504838
      2:      175g  60561
      3:     Chips  49770
      4:      150g  41633
      5:    Kettle  41288
     ---
    217: Frch/Onin   1432
    218:        Pc   1431
    219:       NCC   1419
    220:    Garden   1419
    221:     Fries   1418
```

Removing digits and Removing special characters,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.

```
productWordsdata <- productWordsdata[grepl("^[a-zA-Z0-9]+$", words)]
wordSummary <- productWordsdata[, .N, by = words][order(-N)]
print(wordSummary)
```

```
        words     N
       <char> <int>
      1:    175g 60561
      2:   Chips 49770
      3:    150g 41633
      4:  Kettle 41288
      5:  Smiths 28860
     ---
    191: Whlegrn  1432
    192:      Pc  1431
    193:     NCC  1419
    194:  Garden  1419
    195:   Fries  1418
```

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

```
transactiondata <- transactiondata[PROD_NAME != "SALSA"]
print(transactiondata)
```

```
Error in eval(expr, envir, enclos): object 'PROD_NAME' not found
Traceback:

1. transactiondata[PROD_NAME != "SALSA"]
2. `[.data.frame`(transactiondata, PROD_NAME != "SALSA")
```

Next steps: Explain error

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.:  70021   1st Qu.:  67602
 Median :2018-12-30   Median :130.0   Median : 130358   Median : 135138
 Mean   :2018-12-30   Mean   :135.1   Mean   : 135550   Mean   : 135158
 3rd Qu.:2019-03-31   3rd Qu.:203.0   3rd Qu.: 203094   3rd Qu.: 202701
 Max.   :2019-06-30   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
```

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

```
library(data.table)
transactiondata <- as.data.table(transactiondata)
transactiondata <- transactiondata[grepl("chip|chips", tolower(PROD_NAME)), ]
print(transactiondata)
```

```
             DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
           <Date>     <int>          <int>  <int>    <int>
        1: 2018-10-17         1           1000      1        5
        2: 2019-05-20         1           1343    383       61
        3: 2018-08-17         2           2373    974       69
        4: 2019-05-16         4           4149   3333       16
        5: 2018-08-20         5           5026   4525       42
       ---
    74566: 2018-11-04       271         271193 269365       33
    74567: 2019-05-29       271         271193 269366       75
    74568: 2019-03-25       272         272194 269908       75
    74569: 2018-11-12       272         272319 270087       44
    74570: 2018-12-27       272         272379 270188       42
                                      PROD_NAME PROD_QTY TOT_SALES
                                         <char>    <int>     <num>
       1:   Natural Chip        Compny SeaSalt175g        2       6.0
       2:   Smiths Crinkle Cut  Chips Chicken 170g        2       2.9
       3:   Smiths Chip Thinly  S/Cream&Onion 175g        5      15.0
       4: Smiths Crinkle Chips Salt & Vinegar 330g        1       5.7
       5:  Doritos Corn Chip Mexican Jalapeno 150g        1       3.9
      ---
    74566: Cobs Popd Swt/Chlli &Sr/Cream Chips 110g        2       7.6
    74567:           Cobs Popd Sea Salt  Chips 110g        2       7.6
    74568:           Cobs Popd Sea Salt  Chips 110g        2       7.6
    74569:           Thins Chips Light&  Tangy 175g        2       6.6
    74570:  Doritos Corn Chip Mexican Jalapeno 150g        2       7.8
```

```
highPRODQTY <- transactiondata[PROD_QTY > 199]
print(highPRODQTY)
```

```
Empty data.table (0 rows and 8 cols): DATE,STORE_NBR,LYLTY_CARD_NBR,TXN_ID,PROD_NBR,PROD_NAME...
```

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

```
unique_customer <- transactiondata[LYLTY_CARD_NBR == "226000"]
print(unique_customer)
```

```
Empty data.table (0 rows and 8 cols): DATE,STORE_NBR,LYLTY_CARD_NBR,TXN_ID,PROD_NBR,PROD_NAME...
```

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[LYLTY_CARD_NBR != "226000"]
print(transactiondata)
```

```
             DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
           <Date>     <int>          <int>  <int>    <int>
        1: 2018-10-17         1           1000      1        5
        2: 2019-05-20         1           1343    383       61
        3: 2018-08-17         2           2373    974       69
        4: 2019-05-16         4           4149   3333       16
        5: 2018-08-20         5           5026   4525       42
       ---
    74566: 2018-11-04       271         271193 269365       33
    74567: 2019-05-29       271         271193 269366       75
    74568: 2019-03-25       272         272194 269908       75
    74569: 2018-11-12       272         272319 270087       44
    74570: 2018-12-27       272         272379 270188       42
                                      PROD_NAME PROD_QTY TOT_SALES
                                         <char>    <int>     <num>
        1:   Natural Chip        Compny SeaSalt175g        2       6.0
        2:   Smiths Crinkle Cut  Chips Chicken 170g        2       2.9
        3:   Smiths Chip Thinly  S/Cream&Onion 175g        5      15.0
        4: Smiths Crinkle Chips Salt & Vinegar 330g        1       5.7
        5:  Doritos Corn Chip Mexican Jalapeno 150g        1       3.9
       ---
    74566: Cobs Popd Swt/Chlli &Sr/Cream Chips 110g        2       7.6
    74567:           Cobs Popd Sea Salt  Chips 110g        2       7.6
    74568:           Cobs Popd Sea Salt  Chips 110g        2       7.6
    74569:           Thins Chips Light&  Tangy 175g        2       6.6
    74570:  Doritos Corn Chip Mexican Jalapeno 150g        2       7.8
```

```
print(unique_customer)
```

```
Empty data.table (0 rows and 8 cols): DATE,STORE_NBR,LYLTY_CARD_NBR,TXN_ID,PROD_NBR,PROD_NAME...
```

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

Over to you! Create a summary of transaction count by date.

```
transactiondata <- as.data.table(transactiondata)
transaction_counts <- transactiondata[, .(transaction_count = .N), by = DATE]
print(transaction_counts)
```

```
            DATE transaction_count
          <Date>             <int>
       1: 2018-10-17               213
       2: 2019-05-20               210
       3: 2018-08-17               166
       4: 2019-05-16               194
       5: 2018-08-20               192
      ---
     360: 2019-02-13               195
     361: 2018-08-29               197
     362: 2018-12-30               215
     363: 2018-11-06               195
     364: 2018-12-20               232
```

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

```
row_counts_by_date <- transactiondata %>%
  group_by(DATE) %>%
  summarise(row_count = n())
print(row_counts_by_date)
```

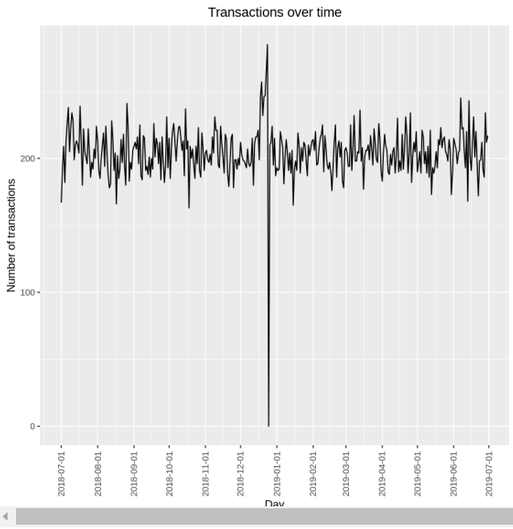
```
# A tibble: 364 × 2
   DATE       row_count
   <date>         <int>
 1 2018-07-01       167
 2 2018-07-02       194
 3 2018-07-03       209
 4 2018-07-04       182
 5 2018-07-05       213
 6 2018-07-06       226
 7 2018-07-07       238
 8 2018-07-08       205
 9 2018-07-09       222
10 2018-07-10       234
# ℹ 354 more rows
```

```
library(dplyr)
date_sequence <- seq(as.Date("2018-07-01"), as.Date("2019-06-30"), by = "day")
date_df <- data.frame(DATE = date_sequence)
transaction_count <- transactiondata %>%
  group_by(DATE) %>%
  summarise(transaction_count = n())
complete_data <- left_join(date_df, transaction_count, by = 'DATE')
complete_data$transaction_count[is.na(complete_data$transaction_count)] <- 0
head(complete_data)
```

A data.frame: 6 × 2

	DATE	transaction_count
	<date>	<dbl>
1	2018-07-01	167
2	2018-07-02	194
3	2018-07-03	209
4	2018-07-04	182
5	2018-07-05	213
6	2018-07-06	226

```
theme_update(plot.title = element_text(hjust = 0.5))
ggplot(complete_data, 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

```
december_data <- complete_data %>%
filter(format(DATE, "%m") == "12")
ggplot(december_data, aes(x = DATE, y = transaction_count)) +   geom_line() +
labs(x = "Day", y = "Number of transactions", title = "Transactions in december") +   scale_x_date(breaks = "1 day", date_labels = "%d") +
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

```
library(readr)
transactiondata[, PACK_SIZE := parse_number(PROD_NAME)]
head(transactiondata$PACK_SIZE)
```

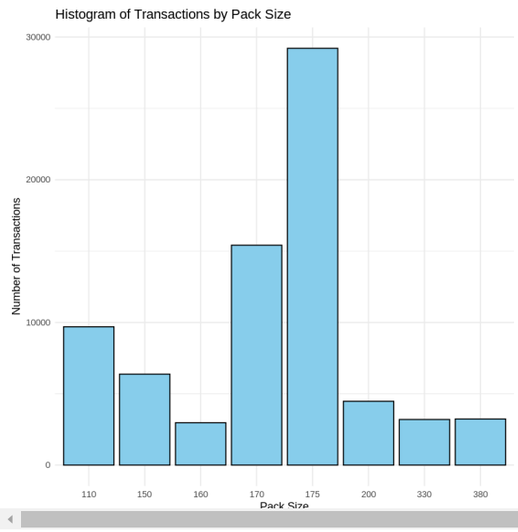
175 · 170 · 175 · 330 · 150 · 330

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

A data.table: 8 × 2

PACK_SIZE <dbl>	N <int>
110	9693
150	6376
160	2970
170	15413
175	29215
200	4473
330	3197
380	3233

```
 ggplot(transactiondata, aes(x = as.factor(PACK_SIZE))) +
geom_bar(fill = "skyblue", color = "black") +
labs(x = "Pack Size", y = "Number of Transactions", title = "Histogram of Transactions by Pack Size") +
theme_minimal()
```



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

```
transactiondata[, Brand_na := sapply(strsplit(PROD_NAME, " "), `[`, 1)]
summary(transactiondata$Brand_na)
```

```
   Length     Class      Mode
    74570 character character
```

```
brand_frequency <- transactiondata[, .N, by = Brand_na][order(-N)]
print(brand_frequency)
```

```
    Brand_na     N
      <char> <int>
1:   Smiths 16872
2:  Doritos 15874
3:    Thins 14075
4:     Cobs  9693
5:       WW  7443
6:  Natural  6050
7: Tostitos  3145
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 and check for similar cases.

```
transactiondata[Brand_na == "RED", Brand_na := "RRD"]
transactiondata[Brand_na == "Doritos", Brand_na := "Dorito"]
transactiondata[Brand_na == "Smiths", Brand_na := "Smith"]
transactiondata[Brand_na == "Infzns", Brand_na := "Infuzions"]
brand_frequency <- transactiondata[, .N, by = Brand_na][order(-N)]
print(brand_frequency)
```

```
    Brand_na     N
      <char> <int>
1:    Smith 16872
2:   Dorito 15874
3:    Thins 14075
4:     Cobs  9693
5:       WW  7443
6:  Natural  6050
7: Tostitos  3145
8:   French  1418
```

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

Customer data

```
view(customerdata)
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" ...
```

the Data looks alright, so lets marge the data.

```
data <- merge(transactiondata, customerdata, all.x = TRUE)
print(data)
```

```
Key: <LYLTY_CARD_NBR>
       LYLTY_CARD_NBR       DATE STORE_NBR TXN_ID PROD_NBR
                <int>     <Date>     <int>  <int>    <int>
    1:           1000 2018-10-17         1      1        5
    2:           1003 2019-03-08         1      4      106
    3:           1004 2018-11-02         1      5       96
    4:           1011 2018-12-19         1     15        1
    5:           1013 2019-03-04         1     18       93
   ---
74566:        2330251 2018-11-29        77 236747       14
74567:        2330291 2019-06-18        77 236754       83
74568:        2330311 2018-11-09        77 236755       90
74569:        2370581 2018-12-17        88 240317       93
74570:        2373711 2018-12-14        88 241815       16
                                    PROD_NAME PROD_QTY TOT_SALES PACK_SIZE
                                      <char>    <int>     <num>     <num>
    1:   Natural Chip        Compny SeaSalt175g        2       6.0       175
    2:   Natural ChipCo      Hony Soy Chckn175g        1       3.0       175
    3:           WW Original Stacked Chips 160g        1       1.9       160
    4:  Smiths Crinkle Cut  Chips Barbecue 170g        1       2.9       170
    5:  Doritos Corn Chip Southern Chicken 150g        1       3.9       150
   ---
74566:   Smiths Crnkle Chip  Orgnl Big Bag 380g        2      11.8       380
74567:       WW D/Style Chip     Sea Salt 200g        1       1.9       200
74568:       Tostitos Smoked     Chipotle 175g        1       4.4       175
74569:  Doritos Corn Chip Southern Chicken 150g        2       7.8       150
74570: Smiths Crinkle Chips Salt & Vinegar 330g        2      11.4       330
       Brand_na             LIFESTAGE PREMIUM_CUSTOMER
         <char>                <char>           <char>
    1:  Natural YOUNG SINGLES/COUPLES          Premium
    2:  Natural        YOUNG FAMILIES           Budget
    3:       WW OLDER SINGLES/COUPLES       Mainstream
    4:    Smith OLDER SINGLES/COUPLES       Mainstream
    5:   Dorito              RETIREES           Budget
   ---
74566:    Smith              RETIREES           Budget
74567:       WW OLDER SINGLES/COUPLES       Mainstream
74568: Tostitos YOUNG SINGLES/COUPLES           Budget
74569:   Dorito OLDER SINGLES/COUPLES           Budget
74570:    Smith 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 check if some customers were not matched on by checking for nulls.

```
summary(data)
```

```
 LYLTY_CARD_NBR         DATE              STORE_NBR         TXN_ID
 Min.   :   1000   Min.   :2018-07-01   Min.   :  1.0   Min.   :     1
 1st Qu.:  70130   1st Qu.:2018-09-30   1st Qu.: 70.0   1st Qu.: 68237
 Median : 131198   Median :2018-12-29   Median :131.0   Median :135504
 Mean   : 135910   Mean   :2018-12-30   Mean   :135.5   Mean   :135534
 3rd Qu.: 203240   3rd Qu.:2019-03-31   3rd Qu.:203.0   3rd Qu.:203284
 Max.   :2373711   Max.   :2019-06-30   Max.   :272.0   Max.   :270209
    PROD_NBR       PROD_NAME            PROD_QTY       TOT_SALES
 Min.   :  1.00   Length:74570       Min.   :1.000   Min.   : 1.900
 1st Qu.: 27.00   Class :character   1st Qu.:2.000   1st Qu.: 5.800
 Median : 42.00   Mode  :character   Median :2.000   Median : 6.600
 Mean   : 50.65                      Mean   :1.904   Mean   : 6.861
 3rd Qu.: 78.00                      3rd Qu.:2.000   3rd Qu.: 7.800
 Max.   :111.00                      Max.   :5.000   Max.   :29.500
   PACK_SIZE       Brand_na          LIFESTAGE         PREMIUM_CUSTOMER
 Min.   :110.0   Length:74570       Length:74570       Length:74570
 1st Qu.:160.0   Class :character   Class :character   Class :character
 Median :175.0   Mode  :character   Mode  :character   Mode  :character
 Mean   :179.8
 3rd Qu.:175.0
 Max.   :380.0
```

Data exploration is now complete!

Data analysis on customer segments

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.

Calculate the summary of sales by those dimensions and create a plot.

```
 total_sales <- data[, .(Total_Sales = sum(TOT_SALES)), by = .(LIFESTAGE, PREMIUM_CUSTOMER)]
print(total_sales)
```

```
                LIFESTAGE PREMIUM_CUSTOMER Total_Sales
                   <char>           <char>       <num>
 1:  YOUNG SINGLES/COUPLES          Premium     11599.4
 2:          YOUNG FAMILIES           Budget     37064.1
 3:  OLDER SINGLES/COUPLES       Mainstream     35443.2
 4:                RETIREES           Budget     30051.8
 5:          OLDER FAMILIES          Premium     21256.1
 6:                RETIREES          Premium     24804.4
 7:          YOUNG FAMILIES       Mainstream     25319.5
 8:          OLDER FAMILIES       Mainstream     28298.5
 9:  YOUNG SINGLES/COUPLES           Budget     16777.1
10: MIDAGE SINGLES/COUPLES           Budget      9838.3
11: MIDAGE SINGLES/COUPLES       Mainstream     23950.4
12:  YOUNG SINGLES/COUPLES       Mainstream     40069.9
13:  OLDER SINGLES/COUPLES           Budget     35943.0
14:                RETIREES       Mainstream     40592.1
15:  OLDER SINGLES/COUPLES          Premium     34545.0
16:            NEW FAMILIES           Budget      5651.4
17:          OLDER FAMILIES           Budget     44859.2
18:            NEW FAMILIES       Mainstream      4307.1
19: MIDAGE SINGLES/COUPLES          Premium     15349.4
20:            NEW FAMILIES          Premium      3087.3
21:          YOUNG FAMILIES          Premium     22781.1
                LIFESTAGE PREMIUM_CUSTOMER Total_Sales
```

```
 ggplot(total_sales, aes(x = LIFESTAGE, y = Total_Sales, fill = PREMIUM_CUSTOMER)) +
 geom_bar(stat = "identity", position = "dodge") +
labs(title = "Total Chip Sales by Lifestage and Premium Customer Segment", x = "Lifestage", y = "Total Sales") +
theme_minimal() +
theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

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.

Calculate the summary of number of customers by those dimensions and create a plot.

```
 customer_summary <- data[, .(number_of_customers = uniqueN(LYLTY_CARD_NBR)), by = .(LIFESTAGE, PREMIUM_CUSTOMER)]
print(customer_summary)
```

```
                LIFESTAGE PREMIUM_CUSTOMER number_of_customers
                   <char>           <char>               <int>
 1:  YOUNG SINGLES/COUPLES          Premium                1299
 2:          YOUNG FAMILIES           Budget                2687
 3:  OLDER SINGLES/COUPLES       Mainstream                3062
 4:                RETIREES           Budget                2722
 5:          OLDER FAMILIES          Premium                1532
 6:                RETIREES          Premium                2287
 7:          YOUNG FAMILIES       Mainstream                1864
 8:          OLDER FAMILIES       Mainstream                1964
 9:  YOUNG SINGLES/COUPLES           Budget                1919
10: MIDAGE SINGLES/COUPLES           Budget                 894
11: MIDAGE SINGLES/COUPLES       Mainstream                1996
12:  YOUNG SINGLES/COUPLES       Mainstream                3952
13:  OLDER SINGLES/COUPLES           Budget                3114
14:                RETIREES       Mainstream                3798
15:  OLDER SINGLES/COUPLES          Premium                2979
16:            NEW FAMILIES           Budget                 581
17:          OLDER FAMILIES           Budget                3177
18:            NEW FAMILIES       Mainstream                 444
19: MIDAGE SINGLES/COUPLES          Premium                1399
20:            NEW FAMILIES          Premium                 316
21:          YOUNG FAMILIES          Premium                1639
                LIFESTAGE PREMIUM_CUSTOMER number_of_customers
```

```
ggplot(customer_summary, aes(x = LIFESTAGE, y = number_of_customers, fill = PREMIUM_CUSTOMER)) +
geom_bar(stat = "identity", position = "dodge") +
labs(x = "Lifestage", y = "Number of Customers", title = "Number of Customers by Lifestage and Premium Customer") +
theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

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

let's Calculate and plot the average number of units per customer by those two dimensions.

```
 average_units_per_customer <- data[, .(total_units = sum(PROD_QTY),
number_of_customers = uniqueN(LYLTY_CARD_NBR)), by = .(LIFESTAGE, PREMIUM_CUSTOMER)]
average_units_per_customer[, avg_units_per_customer := total_units / number_of_customers]
print(average_units_per_customer)
```

```
                LIFESTAGE PREMIUM_CUSTOMER total_units number_of_customers
                   <char>           <char>       <int>               <int>
 1:  YOUNG SINGLES/COUPLES          Premium        3353                1299
 2:         YOUNG FAMILIES           Budget       10462                2687
 3:  OLDER SINGLES/COUPLES       Mainstream        9868                3062
 4:               RETIREES           Budget        8153                2722
 5:         OLDER FAMILIES          Premium        6105                1532
 6:               RETIREES          Premium        6755                2287
 7:         YOUNG FAMILIES       Mainstream        7179                1864
 8:         OLDER FAMILIES       Mainstream        8015                1964
 9:  YOUNG SINGLES/COUPLES           Budget        4857                1919
10: MIDAGE SINGLES/COUPLES           Budget        2794                 894
11: MIDAGE SINGLES/COUPLES       Mainstream        6383                1996
12:  YOUNG SINGLES/COUPLES       Mainstream       10459                3952
13:  OLDER SINGLES/COUPLES           Budget        9867                3114
14:               RETIREES       Mainstream       11212                3798
15:  OLDER SINGLES/COUPLES          Premium        9517                2979
16:           NEW FAMILIES           Budget        1550                 581
17:         OLDER FAMILIES           Budget       12703                3177
18:           NEW FAMILIES       Mainstream        1171                 444
19: MIDAGE SINGLES/COUPLES          Premium        4338                1399
20:           NEW FAMILIES          Premium         836                 316
21:         YOUNG FAMILIES          Premium        6424                1639
                LIFESTAGE PREMIUM_CUSTOMER total_units number_of_customers
    avg_units_per_customer
                     <num>
 1:                2.581216
 2:                3.893562
 3:                3.222730
 4:                2.995224
 5:                3.984987
 6:                2.953651
 7:                3.851395
 8:                4.080957
 9:                2.531006
10:                3.125280
11:                3.197896
12:                2.646508
13:                3.168593
14:                2.952080
15:                3.194696
16:                2.667814
17:                3.998426
18:                2.637387
19:                3.100786
20:                2.645570
21:                3.919463
    avg_units_per_customer
```

```
ggplot(average_units_per_customer, aes(x = LIFESTAGE, y = avg_units_per_customer, fill = PREMIUM_CUSTOMER)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Lifestage", y = "Average Units per Customer", title = "Average Units per Customer by Lifestage and Premium Customer") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

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.

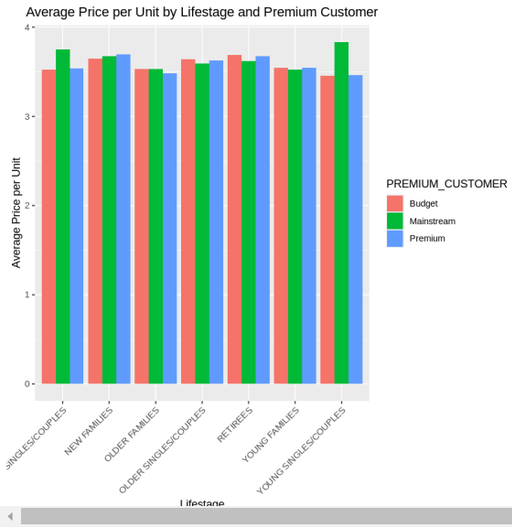
Average price per unit by LIFESTAGE and PREMIUM_CUSTOMER

Over to you! Calculate and plot the average price per unit sold (average sale price) by those two customer dimensions.

```
 data[, total_sales := TOT_SALES]
avg_price_per_unit <- data[, .(total_revenue = sum(total_sales),total_units_sold = sum(PROD_QTY)), by = .(LIFESTAGE, PREMIUM_CUSTOMER)]
avg_price_per_unit[, avg_price := total_revenue / total_units_sold]
print(avg_price_per_unit)
```

```
                LIFESTAGE PREMIUM_CUSTOMER total_revenue total_units_sold
                   <char>           <char>         <num>            <int>
 1:  YOUNG SINGLES/COUPLES          Premium       11599.4             3353
 2:         YOUNG FAMILIES           Budget       37064.1            10462
 3:  OLDER SINGLES/COUPLES       Mainstream       35443.2             9868
 4:               RETIREES           Budget       30051.8             8153
 5:         OLDER FAMILIES          Premium       21256.1             6105
 6:               RETIREES          Premium       24804.4             6755
 7:         YOUNG FAMILIES       Mainstream       25319.5             7179
 8:         OLDER FAMILIES       Mainstream       28298.5             8015
 9:  YOUNG SINGLES/COUPLES           Budget       16777.1             4857
10: MIDAGE SINGLES/COUPLES           Budget        9838.3             2794
11: MIDAGE SINGLES/COUPLES       Mainstream       23950.4             6383
12:  YOUNG SINGLES/COUPLES       Mainstream       40069.9            10459
13:  OLDER SINGLES/COUPLES           Budget       35943.0             9867
14:               RETIREES       Mainstream       40592.1            11212
15:  OLDER SINGLES/COUPLES          Premium       34545.0             9517
16:           NEW FAMILIES           Budget        5651.4             1550
17:         OLDER FAMILIES           Budget       44859.2            12703
18:           NEW FAMILIES       Mainstream        4307.1             1171
19: MIDAGE SINGLES/COUPLES          Premium       15349.4             4338
20:           NEW FAMILIES          Premium        3087.3              836
21:         YOUNG FAMILIES          Premium       22781.1             6424
                LIFESTAGE PREMIUM_CUSTOMER total_revenue total_units_sold
    avg_price
        <num>
 1:  3.459409
 2:  3.542736
 3:  3.591731
 4:  3.685981
 5:  3.481753
 6:  3.672006
 7:  3.526884
 8:  3.530692
 9:  3.454210
10:  3.521224
11:  3.752217
12:  3.831141
13:  3.642749
14:  3.620416
15:  3.629820
16:  3.646065
17:  3.531386
18:  3.678138
19:  3.538359
20:  3.692943
21:  3.546248
    avg_price
```

```
ggplot(avg_price_per_unit, aes(x = LIFESTAGE, y = avg_price, fill = PREMIUM_CUSTOMER)) +
geom_bar(stat = "identity", position = "dodge") +
labs(x = "Lifestage", y = "Average Price per Unit", title = "Average Price per Unit by Lifestage and Premium Customer") +
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

```
mainstream_customers <- data[LIFESTAGE == "YOUNG SINGLES/COUPLES" & PREMIUM_CUSTOMER == "Mainstream"]
premium_budget_customers <- data[LIFESTAGE == "MIDAGE SINGLES/COUPLES" & PREMIUM_CUSTOMER %in% c("Premium", "Budget")]
t_test_result <- t.test(mainstream_customers$total_sales, premium_budget_customers$total_sales)
print(t_test_result)
```

```
        Welch Two Sample t-test

data:  mainstream_customers$total_sales and premium_budget_customers$total_sales
t = 8.7439, df = 8097.6, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.3270052 0.5159935
sample estimates:
mean of x mean of y
 7.100815  6.679316
```

The t-test results in a p-value of XXXXXXX, i.e. the unit price for mainstream, young and mid-age singles and couples [ARE / ARE NOT] 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 Work out if there are brands that these two customer segments prefer more than others.

```
library(arules)
library(arulesViz)
```

```
Error in library(arules): there is no package called ‘arules’
Traceback:

1. library(arules)
```

Next steps: Explain error

```
mainstream_customers <- data[LIFESTAGE == "YOUNG SINGLES/COUPLES" & PREMIUM_CUSTOMER == "Mainstream"]
premium_budget_customers <- data[LIFESTAGE == "MIDAGE SINGLES/COUPLES" & PREMIUM_CUSTOMER %in% c("Premium", "Budget")]
mainstream_brand_counts <- mainstream_customers[, .N, by = Brand_na]
premium_budget_brand_counts <- premium_budget_customers[, .N, by = Brand_na]
```

```
mainstream_total <- sum(mainstream_brand_counts$N)
premium_budget_total <- sum(premium_budget_brand_counts$N)
mainstream_brand_counts[, proportion := N / mainstream_total]
premium_budget_brand_counts[, proportion := N / premium_budget_total]
```

```
brand_comparison <- merge(mainstream_brand_counts, premium_budget_brand_counts, by = "Brand_na", all = TRUE)
setnames(brand_comparison, c("N.x", "proportion.x", "N.y", "proportion.y"), c("mainstream_count", "mainstream_proportion", "premium_budget_c
brand_comparison[is.na(brand_comparison)] <- 0
```

```
brand_comparison_melted <- melt(brand_comparison, id.vars = "Brand_na", measure.vars = c("mainstream_proportion", "premium_budget_proportic
ggplot(brand_comparison_melted, aes(x = reorder(Brand_na, -value), y = value, fill = variable)) +
  geom_bar(stat = "identity", position = "dodge") +
```

```
  coord_flip() +
  labs(title = "Brand Preference Comparison between Customer Segments", x = "Brand", y = "Proportion of Transactions") +
```

we can see that the customers purchase more of Doritos, followed by smith, so 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

let's Do the same for pack size.

```
# segment the DATA
target_segment_avg_pack_size <- target_segment[, .(avg_pack_size = mean(PACK_SIZE, na.rm = TRUE))]
rest_of_population_avg_pack_size <- rest_of_population[, .(avg_pack_size = mean(PACK_SIZE, na.rm = TRUE))]


print(target_segment_avg_pack_size)
print(rest_of_population_avg_pack_size)
```

```
    avg_pack_size
            <num>
1:      182.2931
   avg_pack_size
           <num>
1:      179.6123
```

```
pack_size_comparison <- data.table(
  Segment = c("Target Segment", "Rest of Population"),
  Avg_Pack_Size = c(target_segment_avg_pack_size$avg_pack_size, rest_of_population_avg_pack_size$avg_pack_size)
)


ggplot(pack_size_comparison, aes(x = Segment, y = Avg_Pack_Size, fill = Segment)) +
  geom_bar(stat = "identity", width = 0.6) +
  labs(title = "Average Pack Size Comparison: Target Segment vs Rest of Population",
       x = "Segment",
       y = "Average Pack Size") +
  theme_minimal() +
  theme(legend.position = "none",
        plot.title = element_text(hjust = 0.5))  # Center the title
```