QVI_customer_analytics

Bena

2023-05-24

Setting up

Load required packages and libraries

```
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 4.2.3
```

```
library(ggplot2)
library(ggmosaic)
```

```
## Warning: package 'ggmosaic' was built under R version 4.2.3
```

```
library(readr)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
```

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

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2
## ──
```

```
## ✓ tibble  3.1.8     ✓ dplyr   1.1.0
## ✓ tidyr   1.3.0     ✓ stringr 1.5.0
## ✓ purrr   1.0.1     ✓ forcats 1.0.0
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✕ lubridate::as.difftime() masks base::as.difftime()
## ✕ dplyr::between()         masks data.table::between()
## ✕ lubridate::date()        masks base::date()
## ✕ dplyr::filter()          masks stats::filter()
## ✕ dplyr::first()           masks data.table::first()
## ✕ lubridate::hour()        masks data.table::hour()
## ✕ lubridate::intersect()   masks base::intersect()
## ✕ 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()
## ✕ lubridate::setdiff()     masks base::setdiff()
## ✕ purrr::transpose()       masks data.table::transpose()
## ✕ lubridate::union()       masks base::union()
## ✕ lubridate::wday()        masks data.table::wday()
## ✕ lubridate::week()        masks data.table::week()
## ✕ lubridate::yday()        masks data.table::yday()
## ✕ lubridate::year()        masks data.table::year()
```

Importing data

```
transactions <- read_csv("QVI_transaction_data.csv")
```

```
## Rows: 264839 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.
```

```
transactionData <- fread(paste0("QVI_transaction_data.csv"))
```

```
customers <- 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.
```

```
customerData <- fread(paste0("QVI_purchase_behaviour.csv"))
```

Understanding the data - Exploratory Data Analysis

Examining transaction data

```
str(transactionData)
```

```
## Classes 'data.table' and 'data.frame':   264839 obs. of  8 variables:
##  $ DATE          : chr  "10/17/2018" "5/14/2019" "5/20/2019" "8/17/2018" ...
##  $ 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 C
rinkle Cut  Chips Chicken 170g" "Smiths Chip Thinly  S/Cream&Onion 175g" ...
##  $ 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 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
str(transactions)
```

```
## spc_tbl_ [264,839 × 8] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ DATE          : chr [1:264839] "10/17/2018" "5/14/2019" "5/20/2019" "8/17/2018" ...
##  $ STORE_NBR     : num [1:264839] 1 1 1 2 2 4 4 4 5 7 ...
##  $ LYLTY_CARD_NBR: num [1:264839] 1000 1307 1343 2373 2426 ...
##  $ TXN_ID        : num [1:264839] 1 348 383 974 1038 ...
##  $ PROD_NBR      : num [1:264839] 5 66 61 69 108 57 16 24 42 52 ...
##  $ PROD_NAME     : chr [1:264839] "Natural Chip        Compny SeaSalt175g" "CCs Nacho Cheese    175g"
"Smiths Crinkle Cut  Chips Chicken 170g" "Smiths Chip Thinly  S/Cream&Onion 175g" ...
##  $ PROD_QTY      : num [1:264839] 2 3 2 5 3 1 1 1 1 2 ...
##  $ TOT_SALES     : num [1:264839] 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>
```

```
skimr::skim_without_charts(transactions)
```

Data summary

Name	transactions
Number of rows	264839
Number of columns	8
_______________________	
Column type frequency:	
character	2
numeric	6
________________________	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
DATE	1	1	8	10	0	364	0
PROD_NAME	3	1	17	40	0	114	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
STORE_NBR	3	1	135.08	76.78	1.0	70.0	130.0	203.0	272

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
LYLTY_CARD_NBR	3	1	135549.48	80579.98	1000.0	70021.0	130357.5	203094.2	2373711
TXN_ID	3	1	135158.31	78133.03	1.0	67601.5	135137.5	202701.2	2415841
PROD_NBR	3	1	56.58	32.83	1.0	28.0	56.0	85.0	114
PROD_QTY	3	1	1.91	0.64	1.0	2.0	2.0	2.0	200
TOT_SALES	3	1	7.30	3.08	1.5	5.4	7.4	9.2	650

```
skimr::skim_without_charts(transactionData)
```

Data summary

Name	transactionData
Number of rows	264839
Number of columns	8
Key	NULL
_______________________	
Column type frequency:	
character	2
numeric	6
________________________	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
DATE	0	1	0	10	1	365	0
PROD_NAME	0	1	0	40	3	115	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
STORE_NBR	3	1	135.08	76.78	1.0	70.0	130.0	203.0	272
LYLTY_CARD_NBR	3	1	135549.48	80579.98	1000.0	70021.0	130357.5	203094.2	2373711
TXN_ID	3	1	135158.31	78133.03	1.0	67601.5	135137.5	202701.2	2415841
PROD_NBR	3	1	56.58	32.83	1.0	28.0	56.0	85.0	114
PROD_QTY	3	1	1.91	0.64	1.0	2.0	2.0	2.0	200
TOT_SALES	3	1	7.30	3.08	1.5	5.4	7.4	9.2	650

```
colnames(transactions)
```

```
## [1] "DATE"           "STORE_NBR"      "LYLTY_CARD_NBR" "TXN_ID"
## [5] "PROD_NBR"       "PROD_NAME"      "PROD_QTY"       "TOT_SALES"
```

```
colnames(transactionData)
```

```
## [1] "DATE"           "STORE_NBR"      "LYLTY_CARD_NBR" "TXN_ID"
## [5] "PROD_NBR"       "PROD_NAME"      "PROD_QTY"       "TOT_SALES"
```

The skim_without_charts() function revealed that there are no missing values or whitespaces. Thus, we have 264839 & 8 variables

```
blanks <- transactionData %>%                       #confirmed there are no nulls
  filter(
    DATE == "NA",
    PROD_NAME == "NA"
    )
```

1. Amend data type for DATE column from chr

```
transactionData$DATE <- parse_date_time(transactionData$DATE, "mdy")  #converts to POSIXct
transactionData$DATE <- as.Date(transactionData$DATE, formats = "%y/%m/%d")  #converts the POSIXct to Da
te
```

2. Examine the PROD_NAME column

```
product_summary <- transactionData %>%
  pull(PROD_NAME) %>%
  unique   #lists unique products

print(product_summary)
```

```
##   [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"
##  [11] "Kettle Sensations   Siracha Lime 150g"
##  [12] "Twisties Cheese     270g"
##  [13] "WW Crinkle Cut      Chicken 175g"
##  [14] "Thins Chips Light&  Tangy 175g"
##  [15] "CCs Original 175g"
##  [16] "Burger Rings 220g"
##  [17] "NCC Sour Cream &    Garden Chives 175g"
##  [18] "Doritos Corn Chip Southern Chicken 150g"
##  [19] "Cheezels Cheese Box 125g"
##  [20] "Smiths Crinkle      Original 330g"
##  [21] "Infzns Crn Crnchers Tangy Gcamole 110g"
##  [22] "Kettle Sea Salt     And Vinegar 175g"
##  [23] "Smiths Chip Thinly  Cut Original 175g"
##  [24] "Kettle Original 175g"
##  [25] "Red Rock Deli Thai  Chilli&Lime 150g"
##  [26] "Pringles Sthrn FriedChicken 134g"
##  [27] "Pringles Sweet&Spcy BBQ 134g"
##  [28] "Red Rock Deli SR    Salsa & Mzzrlla 150g"
##  [29] "Thins Chips         Originl saltd 175g"
##  [30] "Red Rock Deli Sp    Salt & Truffle 150G"
##  [31] "Smiths Thinly       Swt Chli&S/Cream175G"
##  [32] "Kettle Chilli 175g"
##  [33] "Doritos Mexicana    170g"
##  [34] "Smiths Crinkle Cut  French OnionDip 150g"
##  [35] "Natural ChipCo      Hony Soy Chckn175g"
##  [36] "Dorito Corn Chp     Supreme 380g"
##  [37] "Twisties Chicken270g"
##  [38] "Smiths Thinly Cut   Roast Chicken 175g"
##  [39] "Smiths Crinkle Cut  Tomato Salsa 150g"
##  [40] "Kettle Mozzarella   Basil & Pesto 175g"
##  [41] "Infuzions Thai SweetChili PotatoMix 110g"
##  [42] "Kettle Sensations   Camembert & Fig 150g"
##  [43] "Smith Crinkle Cut   Mac N Cheese 150g"
##  [44] "Kettle Honey Soy    Chicken 175g"
##  [45] "Thins Chips Seasonedchicken 175g"
##  [46] "Smiths Crinkle Cut  Salt & Vinegar 170g"
##  [47] "Infuzions BBQ Rib   Prawn Crackers 110g"
##  [48] "GrnWves Plus Btroot & Chilli Jam 180g"
##  [49] "Tyrrells Crisps     Lightly Salted 165g"
##  [50] "Kettle Sweet Chilli And Sour Cream 175g"
##  [51] "Doritos Salsa       Medium 300g"
##  [52] "Kettle 135g Swt Pot Sea Salt"
##  [53] "Pringles SourCream  Onion 134g"
##  [54] "Doritos Corn Chips  Original 170g"
##  [55] "Twisties Cheese     Burger 250g"
##  [56] "Old El Paso Salsa   Dip Chnky Tom Ht300g"
```

```
##  [57] "Cobs Popd Swt/Chlli &Sr/Cream Chips 110g"
##  [58] "Woolworths Mild     Salsa 300g"
##  [59] "Natural Chip Co     Tmato Hrb&Spce 175g"
##  [60] "Smiths Crinkle Cut  Chips Original 170g"
##  [61] "Cobs Popd Sea Salt  Chips 110g"
##  [62] "Smiths Crinkle Cut  Chips Chs&Onion170g"
##  [63] "French Fries Potato Chips 175g"
##  [64] "Old El Paso Salsa   Dip Tomato Med 300g"
##  [65] "Doritos Corn Chips  Cheese Supreme 170g"
##  [66] "Pringles Original   Crisps 134g"
##  [67] "RRD Chilli&         Coconut 150g"
##  [68] "WW Original Corn    Chips 200g"
##  [69] "Thins Potato Chips  Hot & Spicy 175g"
##  [70] "Cobs Popd Sour Crm  &Chives Chips 110g"
##  [71] "Smiths Crnkle Chip  Orgnl Big Bag 380g"
##  [72] "Doritos Corn Chips  Nacho Cheese 170g"
##  [73] "Kettle Sensations   BBQ&Maple 150g"
##  [74] "WW D/Style Chip     Sea Salt 200g"
##  [75] "Pringles Chicken    Salt Crips 134g"
##  [76] "WW Original Stacked Chips 160g"
##  [77] "Smiths Chip Thinly  CutSalt/Vinegr175g"
##  [78] "Cheezels Cheese 330g"
##  [79] "Tostitos Lightly    Salted 175g"
##  [80] "Thins Chips Salt &  Vinegar 175g"
##  [81] "Smiths Crinkle Cut  Chips Barbecue 170g"
##  [82] "Cheetos Puffs 165g"
##  [83] "RRD Sweet Chilli &  Sour Cream 165g"
##  [84] "WW Crinkle Cut      Original 175g"
##  [85] "Tostitos Splash Of  Lime 175g"
##  [86] "Woolworths Medium   Salsa 300g"
##  [87] "Kettle Tortilla ChpsBtroot&Ricotta 150g"
##  [88] "CCs Tasty Cheese    175g"
##  [89] "Woolworths Cheese   Rings 190g"
##  [90] "Tostitos Smoked     Chipotle 175g"
##  [91] "Pringles Barbeque   134g"
##  [92] "WW Supreme Cheese   Corn Chips 200g"
##  [93] "Pringles Mystery    Flavour 134g"
##  [94] "Tyrrells Crisps     Ched & Chives 165g"
##  [95] "Snbts Whlgrn Crisps Cheddr&Mstrd 90g"
##  [96] "Cheetos Chs & Bacon Balls 190g"
##  [97] "Pringles Slt Vingar 134g"
##  [98] "Infuzions SourCream&Herbs Veg Strws 110g"
##  [99] "Kettle Tortilla ChpsFeta&Garlic 150g"
## [100] "Infuzions Mango     Chutny Papadums 70g"
## [101] "RRD Steak &         Chimuchurri 150g"
## [102] "RRD Honey Soy       Chicken 165g"
## [103] "Sunbites Whlegrn    Crisps Frch/Onin 90g"
## [104] "RRD Salt & Vinegar  165g"
## [105] "Doritos Cheese      Supreme 330g"
## [106] "Smiths Crinkle Cut  Snag&Sauce 150g"
## [107] "WW Sour Cream &OnionStacked Chips 160g"
## [108] "RRD Lime & Pepper   165g"
## [109] "Natural ChipCo Sea  Salt & Vinegr 175g"
## [110] "Red Rock Deli Chikn&Garlic Aioli 150g"
## [111] "RRD SR Slow Rst     Pork Belly 150g"
## [112] "RRD Pc Sea Salt     165g"
## [113] "Smith Crinkle Cut   Bolognese 150g"
```

```
## [114] "Doritos Salsa Mild  300g"
## [115] ""
```

The data includes other products which aren't chips

3. Summarize the data to spot outliers & nulls

```
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
##  NA's   :1            NA's   :3       NA's   :3         NA's   :3
##     PROD_NBR       PROD_NAME            PROD_QTY         TOT_SALES
##  Min.   :  1.00   Length:264839      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
##  NA's   :3                           NA's   :3         NA's   :3
```

```
productWords <- data.table(unlist(strsplit(unique(transactionData[, PROD_NAME]), " ")))
setnames(productWords, 'words')             #ignore, the product names are truncated & mixed up**

View(productWords)
```

Filter to remain with Chips products only

```
pattern <- "[Cc]hip([^otle])|(?i)chp(\\s)|(?i)chps(\\w+)"
transactionData3 <- transactionData %>%
  mutate(CHIPS = str_detect(PROD_NAME,pattern)) %>%
  filter(CHIPS == "TRUE")

View(transactionData3)
```

The filtering where the words with any of the regexp pattern was found left us with 84188 observations which is 31.8% of the original transactions data Export data to excel to confirm it's chips data only

```
write_csv(transactionData3,"transactions_nochp3.csv")
```

4. Create size column from product name

```
transactionData3 <- transactionData3 %>%
  mutate(SIZE = str_sub(PROD_NAME, -4, -1)) %>%
  mutate(SIZE = str_replace_all(SIZE, "g", ""))     # Remove "g" from size

transactionData3 <- transactionData3 %>%
  mutate(SIZE = as.numeric(SIZE))                   #converts `SIZE`from chr to numeric data type
```

Alternatively

```
transactionData4 <- transactionData %>%
  mutate(size = parse_number(PROD_NAME))

View(transactionData4)
```

Remove the ‘g’ from size

```
transactionData3 <- transactionData3 %>%
  mutate(PROD_NAME = str_replace_all(PROD_NAME, "([\\d+]g)", "")) %>%
  mutate(PROD_NAME = str_replace_all(PROD_NAME, "([\\d+])", ""))
```

```
size_summary <- transactionData3 %>%
  group_by(SIZE) %>%
  summarise(min_size = min(SIZE),
            max_size = max(SIZE))

View(size_summary)
```

The package sizes look reasonable, thus ok to proceed

5. Check out products in `PROD_NAME

```
unique_chips <- transactionData3 %>%
  pull(PROD_NAME) %>%
  unique

print(unique_chips)
```

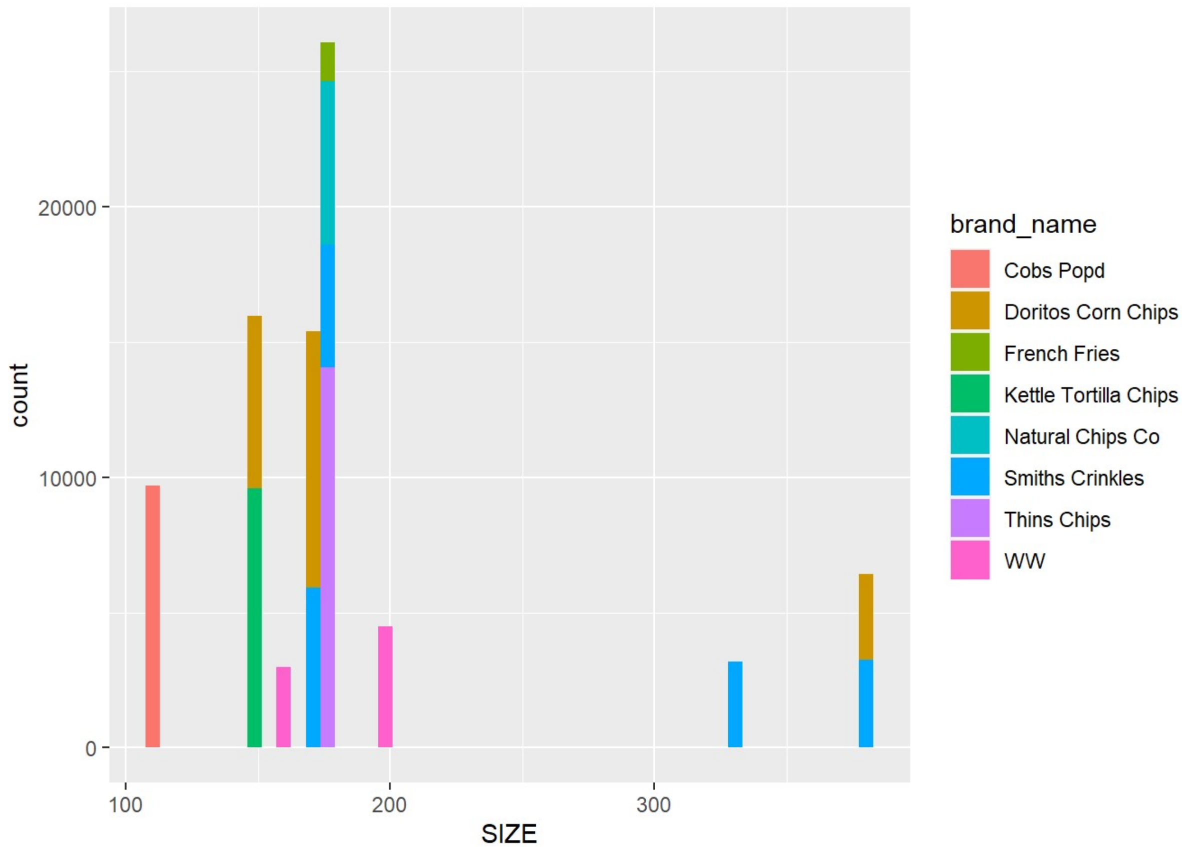
```
##  [1] "Natural Chip        Compny SeaSalt"
##  [2] "Smiths Crinkle Cut  Chips Chicken "
##  [3] "Smiths Chip Thinly  S/Cream&Onion "
##  [4] "Kettle Tortilla ChpsHny&Jlpno Chili "
##  [5] "Smiths Crinkle Chips Salt & Vinegar "
##  [6] "Doritos Corn Chip Mexican Jalapeno "
##  [7] "Thins Chips Light&  Tangy "
##  [8] "Doritos Corn Chip Southern Chicken "
##  [9] "Smiths Chip Thinly  Cut Original "
## [10] "Thins Chips         Originl saltd "
## [11] "Natural ChipCo      Hony Soy Chckn"
## [12] "Dorito Corn Chp     Supreme "
## [13] "Thins Chips Seasonedchicken "
## [14] "Doritos Corn Chips  Original "
## [15] "Cobs Popd Swt/Chlli &Sr/Cream Chips "
## [16] "Natural Chip Co     Tmato Hrb&Spce "
## [17] "Smiths Crinkle Cut  Chips Original "
## [18] "Cobs Popd Sea Salt  Chips "
## [19] "Smiths Crinkle Cut  Chips Chs&Onion"
## [20] "French Fries Potato Chips "
## [21] "Doritos Corn Chips  Cheese Supreme "
## [22] "WW Original Corn    Chips "
## [23] "Thins Potato Chips  Hot & Spicy "
## [24] "Cobs Popd Sour Crm  &Chives Chips "
## [25] "Smiths Crnkle Chip  Orgnl Big Bag "
## [26] "Doritos Corn Chips  Nacho Cheese "
## [27] "WW D/Style Chip     Sea Salt "
## [28] "WW Original Stacked Chips "
## [29] "Smiths Chip Thinly  CutSalt/Vinegr"
## [30] "Thins Chips Salt &  Vinegar "
## [31] "Smiths Crinkle Cut  Chips Barbecue "
## [32] "Kettle Tortilla ChpsBtroot&Ricotta "
## [33] "WW Supreme Cheese   Corn Chips "
## [34] "Kettle Tortilla ChpsFeta&Garlic "
## [35] "WW Sour Cream &OnionStacked Chips "
## [36] "Natural ChipCo Sea  Salt & Vinegr "
```

Then get brand_names from `PROD_NAME`

```
transactionData3 <- transactionData3 %>%
  mutate(brand = str_sub(PROD_NAME, 1, 4)) %>%
  mutate(brand_name = case_when(
      brand == "Natu" ~ "Natural Chips Co",
      brand == "Smit" ~ "Smiths Crinkles",
      brand == "Kett" ~ "Kettle Tortilla Chips",
      brand == "Thin" ~ "Thins Chips",
      brand == "Dori" ~ "Doritos Corn Chips",
      brand == "Cobs" ~ "Cobs Popd",
      brand == "Fren" ~ "French Fries",
      brand == "WW O" ~ "WW",
      brand == "WW D" ~ "WW",
      brand == "WW S" ~ "WW"))
```

6. Visualize the number bought for each brand & size

```
transactionData3 %>%
  ggplot(aes(x = SIZE, fill = brand_name)) +
  geom_histogram(bins = 50)
```



We see that the most popular size & brand is Thins Chips, all packaged in 175g Followed by Cobs Popd, all packaged in 110g & Doritos Corn Chips - 170g

a. More about Thins Chips

```
thins <- transactionData3 %>%
  select(SIZE, brand_name) %>%
  filter(brand_name == "Thins Chips")

View(thins)
```

To confirm Thins Chips are only packaged in 175g

```
thins <- thins %>%
  filter(SIZE != 175)
```

b. More about Cobs Popd

```
cobs <- transactionData3 %>%
  select(SIZE, brand_name) %>%
  filter(brand_name == "Cobs Popd")
```

To confirm Cobs Popd are only packaged in 110g

```
cobs <- cobs %>%
  filter(SIZE != 110)
```

c. More about Doritos Corn Chips

```
doritos <- transactionData3 %>%
  select(SIZE, brand_name) %>%
  filter(brand_name == "Doritos Corn Chips")

View(doritos)
```

7. Counting frequency for each brand

```
brand_summary <- transactionData3 %>%
  group_by(brand_name) %>%
  summarize(number_bought = n(),
            avg_qnty = mean(PROD_QTY),
            avg_sales = mean(TOT_SALES),
            min_qnty = min(PROD_QTY),
            max_qnty = max(PROD_QTY)) %>%
  arrange(-avg_sales)

View(brand_summary)
```

Minimum quantity for each brand is 1 and maximum quantity ranges between 3-5 except Doritos Corn Chips has a maximum quantity of 200 in one transaction. Which needs to be investigated further

```
Q200 <- transactionData3 %>%
  filter(PROD_QTY == 200)

View(Q200)
```

2 obs have `PROD_QTY` =200, both were bought using the same `LYLTY_CARD_NBR` Investigate whether the same customer has other purchases

```
card_226000 <- transactionData3 %>%
  filter(LYLTY_CARD_NBR == 226000)

View(card_226000)
```

There are only 2 transactions for this customer and it can be assumed that it’s not a retail customer. The transactions are also months apart, it can be concluded that it’s not a regular purchase. We can therefore exclude these 2 obs from analysis because they are outliers.

```
transactionData3 <- transactionData3 %>%
  filter(PROD_QTY != 200)
```

Re-do the brand summary

```
brand_summary2 <- transactionData3 %>%
  group_by(brand_name) %>%
  summarize(transactions_no = n(),
            quantity_bought =sum(PROD_QTY),
            avg_qnty = mean(PROD_QTY),
            avg_sales = mean(TOT_SALES),
            min_qnty = min(PROD_QTY),
            max_qnty = max(PROD_QTY)) %>%
  arrange(-avg_sales)

View(brand_summary2)
```

Impact that the outlier had it: - reduced number_bought from 19059 to 19057 ** use total quantity?? - reduced avg_qnty from 1.935988 to 1.915202 - reduced avg_sales from 8.812073 to 8.744781

From the brand_summary, it is evident that most clients buy 2 packets of chips on average

Trend Analysis

```
date_summary <- transactionData3 %>%
  group_by(DATE) %>%
  summarise(tranx_per_day = n()) %>%
  arrange(DATE)
```

1. Sort the dates in ascending order

```
date_summary <- date_summary %>%
  mutate(DATE = as.Date(DATE, "%m/%d/%Y")) %>%
  arrange(DATE)

View(date_summary)
```

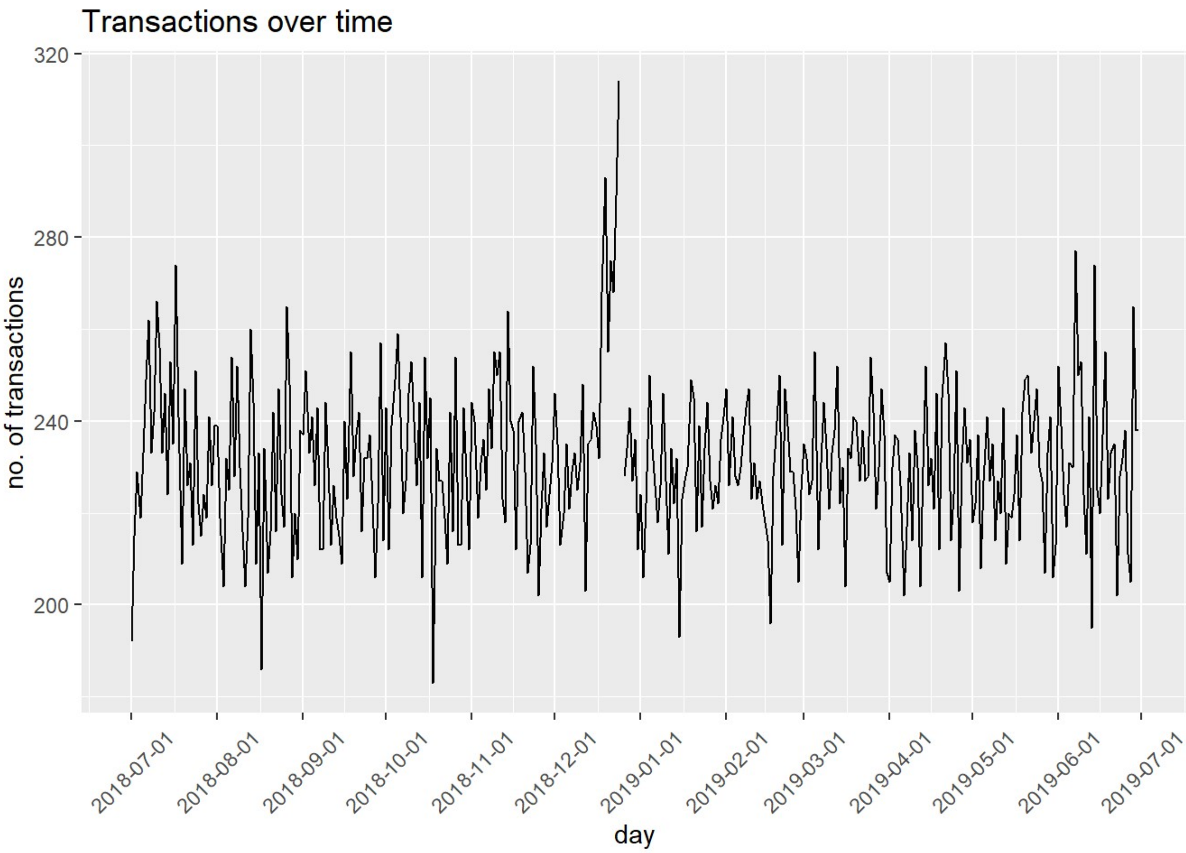
Alternatively, create a sequence for dates then merge with date_summary

```
alldates <- data.table(seq(as.Date("2018-07-01"),as.Date("2019-06-30"),by ="day"))
names(alldates)[1] <- "DATE"                             #to rename column from V1
```

```
transactions_by_day <- merge(alldates, date_summary, all.x=TRUE)
```

2. Plot the dates

```
ggplot(data = transactions_by_day, aes(x = DATE, y=tranx_per_day)) +
  geom_line() +
  labs(x = "day", y = "no. of transactions", title = "Transactions over time") +
  scale_x_date(breaks = "1 month") +
  theme(axis.text.x = element_text(angle = 45, vjust = 0.5))
```



We can see there's a sharp increase in Dec & sharp decrease in Oct & Aug Focus on these months & look at individual days

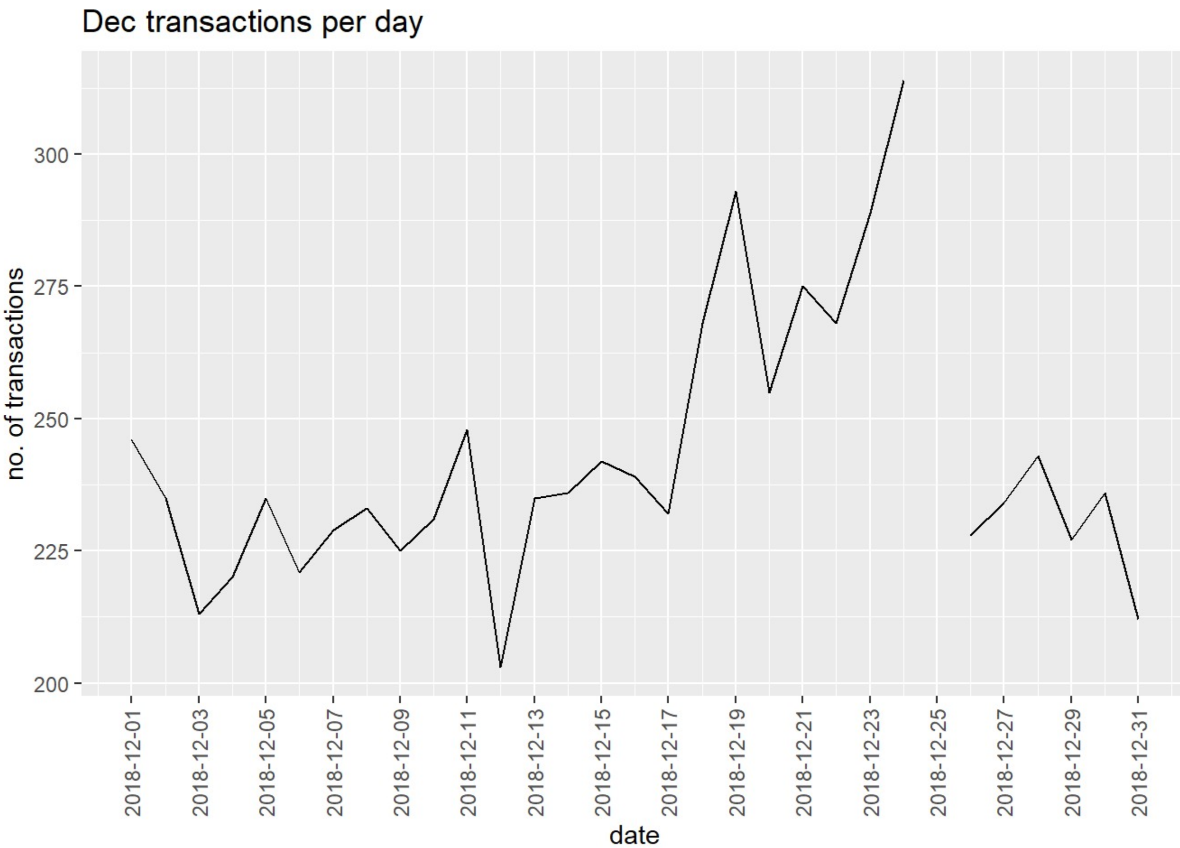
a. More about Dec data

```
dec_summary <- transactions_by_day %>%
  filter(DATE >= '2018-12-01' & DATE <= '2018-12-31')

View(dec_summary)
```

Visualize Dec data only

```
transactions_by_day %>%
  filter(DATE >= '2018-12-01' & DATE <= '2018-12-31') %>%
  ggplot(aes(x = DATE, y = tranx_per_day)) +
  geom_line() +
  labs(x = "date", y = "no. of transactions", title = "Dec transactions per day") +
  scale_x_date(breaks = "2 day") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```



The sharp increase can be attributed to an increase in sales in the week leading to Christmas Christmas also has 0 sales because it's a public holiday and in most cases, most if not all shops are closed on that day This also explains why our date_summary has 364 entries as opposed to the 365 days in a year

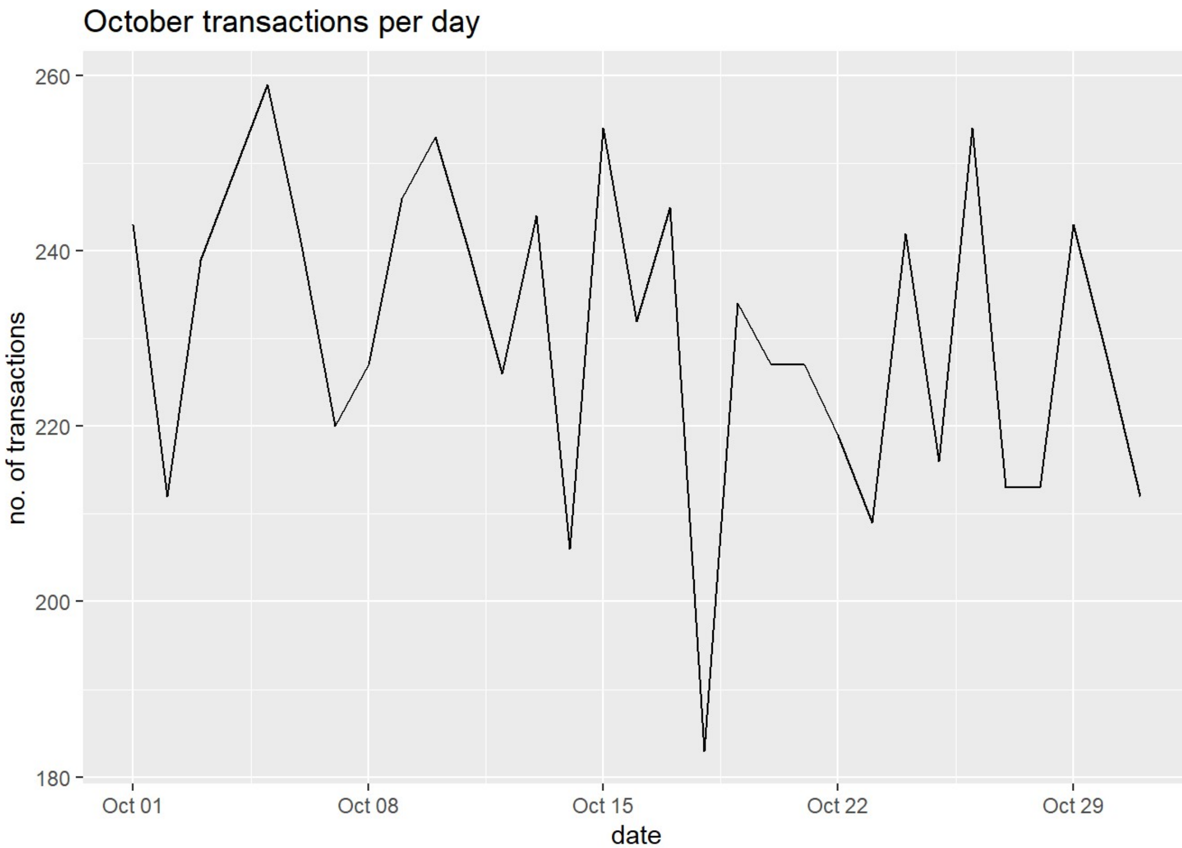
b. More about Oct data

```
oct_summary <- date_summary %>%
  filter(DATE >= '2018-10-01' & DATE <= '2018-10-31')

View(oct_summary)
```

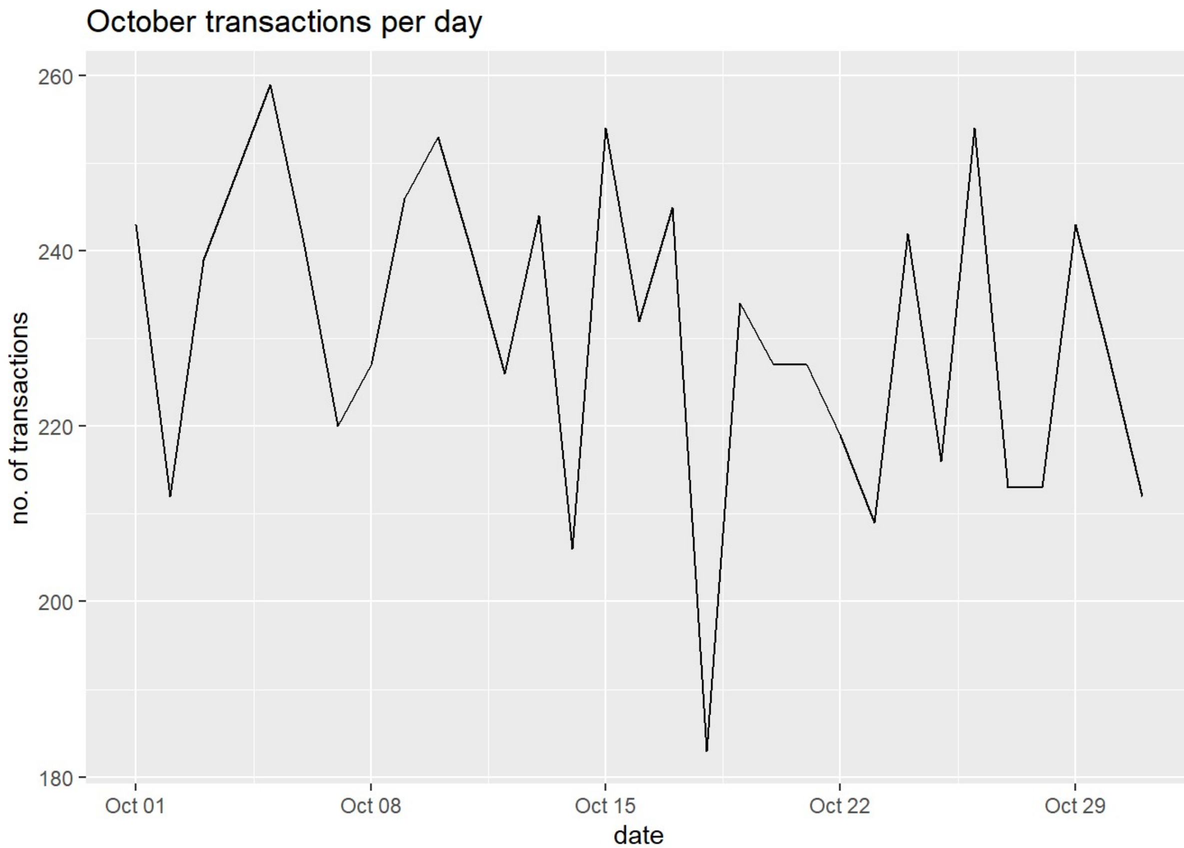
Visualize Oct data only

```
date_summary %>%
  filter(DATE >= '2018-10-01' & DATE <= '2018-10-31') %>%
  ggplot(aes(x = DATE, y = tranx_per_day)) +
  geom_line() +
  labs(x = "date", y = "no. of transactions", title = "October transactions per day")
```



or

```
oct_summary %>%
  ggplot(aes(x = DATE, y = tranx_per_day)) +
  geom_line() +
  labs(x = "date", y = "no. of transactions", title = "October transactions per day")
```



There was a dip on 18th which pulled down the sales in October. Why? **

We're happy with the transactions data. We'll proceed with data cleaning, manipulation & analysis of the customer data

Examining customer data

```
View(customerData)
skimr::skim_without_charts(customerData)
```

Data summary

Name	customerData
Number of rows	72637
Number of columns	3
Key	NULL
________________________	
Column type frequency:	
character	2
numeric	1
________________________	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
LIFESTAGE	0	1	8	22	0	7	0
PREMIUM_CUSTOMER	0	1	6	10	0	3	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
LYLTY_CARD_NBR	0	1	136185.9	89892.93	1000	66202	134040	203375	2373711

There are no missing values or whitespaces. Thus, we'll be working with 72637 observations & 8 variables.

Exploratory Data Analysis of customer data

```
life_stage <- customerData %>%
  pull(LIFESTAGE) %>%
  unique

print(life_stage)
```

```
## [1] "YOUNG SINGLES/COUPLES"  "YOUNG FAMILIES"         "OLDER SINGLES/COUPLES"
## [4] "MIDAGE SINGLES/COUPLES" "NEW FAMILIES"           "OLDER FAMILIES"
## [7] "RETIREES"
```

Above code displays unique entries in the LIFESTAGE column

A summary of `LIFESTAGE`

```
life_stage <- customerData %>%
  group_by(LIFESTAGE) %>%
  summarise(number = n()) %>%
  arrange(-number)

View(life_stage)

customerData[, .N, by = LIFESTAGE][order(-N)]
```

```
##                 LIFESTAGE     N
## 1:               RETIREES 14805
## 2:  OLDER SINGLES/COUPLES 14609
## 3:  YOUNG SINGLES/COUPLES 14441
## 4:         OLDER FAMILIES  9780
## 5:         YOUNG FAMILIES  9178
## 6: MIDAGE SINGLES/COUPLES  7275
## 7:           NEW FAMILIES  2549
```

The highest number of customers fall under the RETIREES category, followed by OLDER SINGLES/COUPLES. The least number of customers are in NEW FAMILIES category

Display unique entries in `PREMIUM_CUSTOMER` column

```
customer_category <- customerData %>%
  pull(PREMIUM_CUSTOMER) %>%
  unique

print(customer_category)
```

```
## [1] "Premium"    "Mainstream" "Budget"
```

A summary of `PREMIUM_CUSTOMER` column

```
customer_category <- customerData %>%
  group_by(PREMIUM_CUSTOMER) %>%
  summarize(number = n()) %>%
  arrange(-number)

View(customer_category)

customerData[, .N, by = PREMIUM_CUSTOMER][order(-N)]
```

```
##    PREMIUM_CUSTOMER     N
## 1:       Mainstream 29245
## 2:           Budget 24470
## 3:          Premium 18922
```

We see the most customers are in Mainstream segment, followed by Budget then Premium customers

Customer summary according to `LYLTY_CARD_NBR`

```
customer_summary <- customerData %>%
group_by(LYLTY_CARD_NBR) %>%
summarise(tranxn_no = n())

View(customer_summary)
```

```
customer_summary1 <- customer_summary %>%
  filter(tranxn_no != 1)
```

There's only transaction per card

Merge customerData & transactionData

Since we want to keep all observations in transactionData we'll use the left join to merge with the customerData

```
combined_data <- merge(transactionData3, customerData, all.x = TRUE)

View(combined_data)
```

or

```
combined_data2 <- transactionData3 %>%
  left_join(customerData, by = "LYLTY_CARD_NBR")

View(combined_data2)
```

1. Delete unnecessary columns

```
combined_data <- combined_data %>%
  select(c(-9,-11))                              #deleted CHIPS & brand columns respectively
```

2. Standardize column names

```
names(combined_data) <- tolower(names(combined_data))
```

3. Rename columns so they're easier to remember

```
names(combined_data)[1] <- "loyalty_card_number"    #from lylty_card_nbr
names(combined_data)[5] <- "product_number"      #from prod_nbr
names(combined_data)[6] <- "product_name"      #from prod_name
names(combined_data)[7] <- "product_quantity"      #from prod_qty
names(combined_data)[8] <- "total_sales"      #from tot_sales
names(combined_data)[9] <- "package_size"      #from size
names(combined_data)[12] <- "segment"      #from premium_customer
```

4. Confirm if there's any customer who wasn't matched to a transaction

```
combined_data3 <- combined_data %>%
  filter(segment == "NA" | lifestage == "NA")
```

Alternative method

```
combined_data3[is.null(lifestage), .N]
```

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

```
combined_data3[is.null(segment), .N]
```

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

5. Save merged data frame for later using write_csv or fwrite()

```
fwrite(combined_data3, paste0("QVI_data.csv"))
```

Data preparation is over now, time for data analysis on customer segments # Define some metrics

a. Calculate total sales by `lifestage` and `segment` & plot the split by these segments to describe which customer segment contributes most to chip sales.

```
total_sales_lifestage <- combined_data %>%
  group_by(lifestage, segment) %>%
  summarise(total_sales = sum(total_sales)) %>%
  arrange(-total_sales)
```

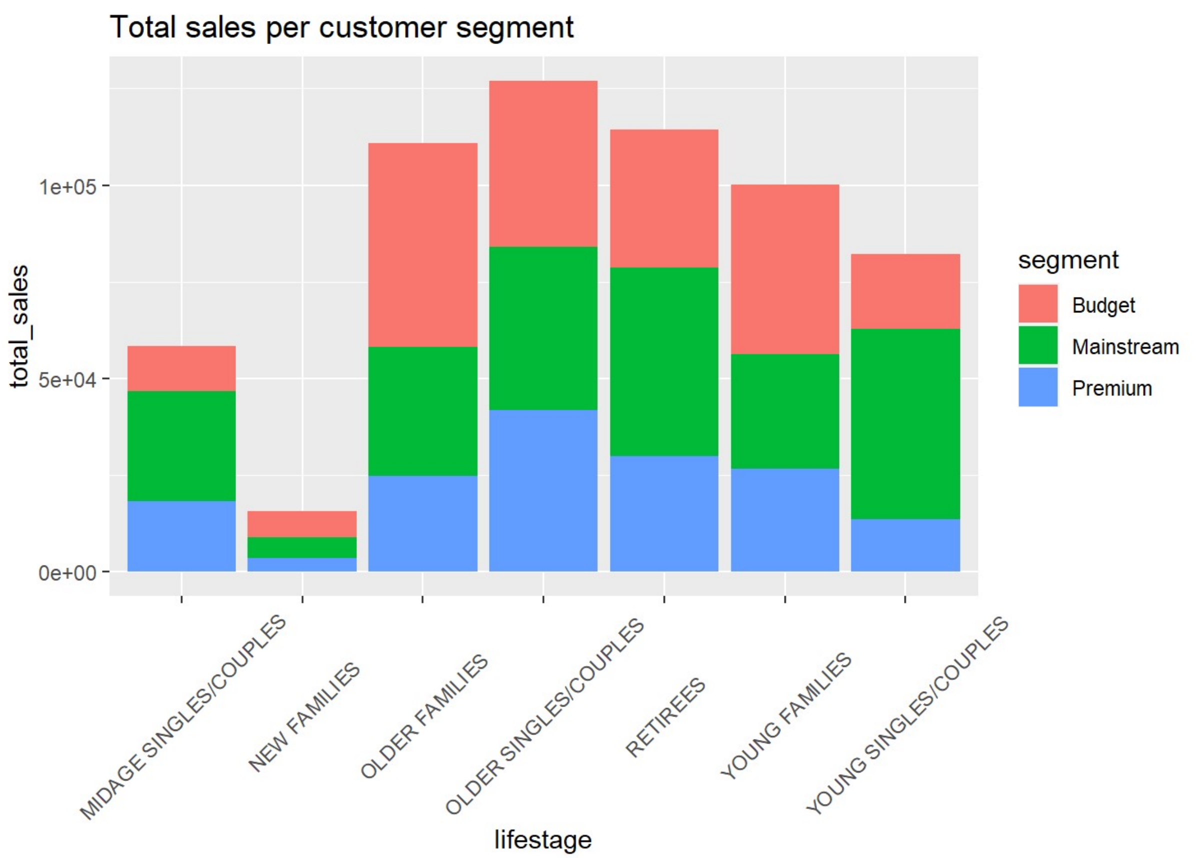
```
## `summarise()` has grouped output by 'lifestage'. You can override using the
## `.groups` argument.
```

or

```
total_sales_lifestage2 <- combined_data[, .(sales = sum(total_sales)), .(lifestage, segment)]
```

A visualization for the same

```
combined_data %>%
  ggplot(aes(x = lifestage, y = total_sales, fill = segment)) +
  geom_col() +
  labs(y = "total_sales", title = "Total sales per customer segment") +     #exact figures to show on y-
axis**
  theme(axis.text.x = element_text(angle = 45, vjust = 0.5))
```

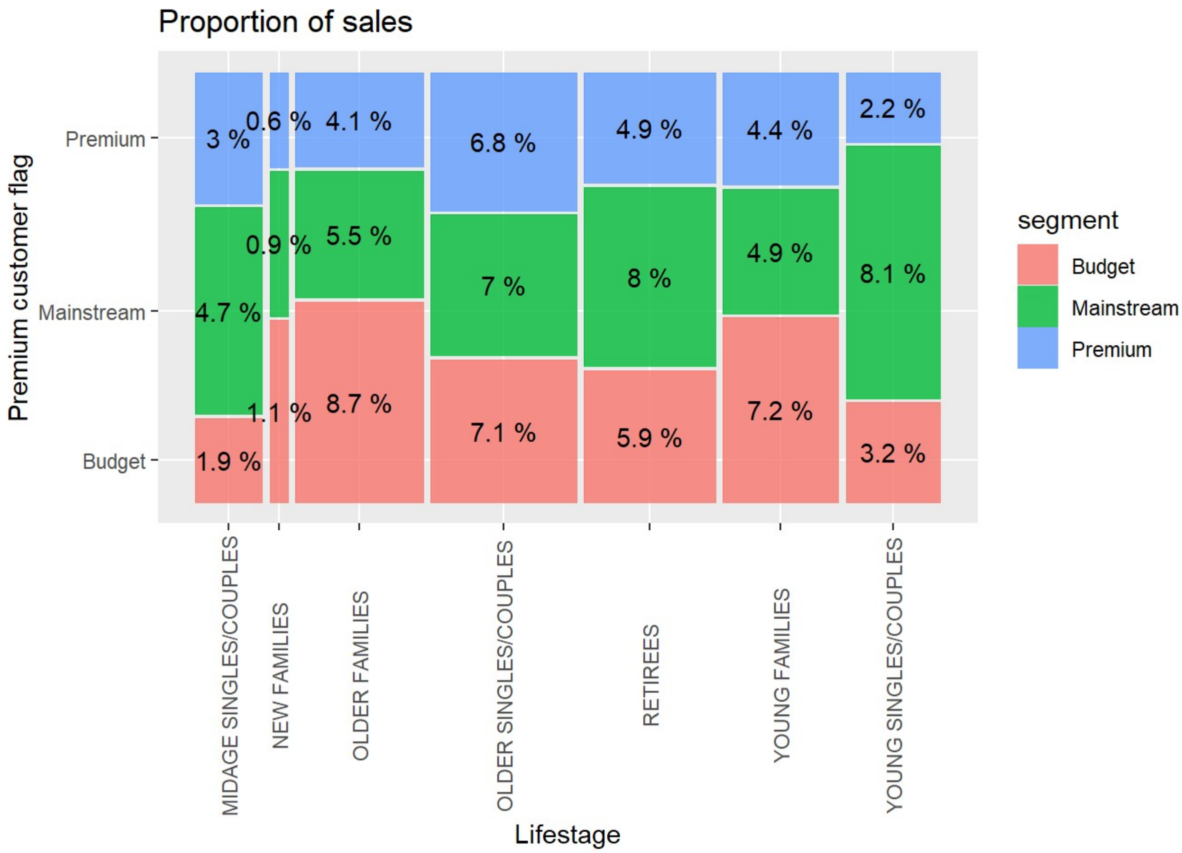


The customer category with the most sales are older families - budget segment & young singles/couples - mainstream segment and Mainstream Retirees. Alternatively, plot using

```
p <- ggplot(data = total_sales_lifestage2) +
  geom_mosaic(aes(weight = sales, x = product(segment, lifestage), fill = segment)) +
  labs(x = "Lifestage", y = "Premium customer flag", title = "Proportion of sales") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))

p + geom_text(data = ggplot_build(p)$data[[1]],
            aes(x = (xmin + xmax)/2, y = (ymin + ymax)/2, label = as.character(paste(round(.wt/sum(.wt),
3)*100,'%'))))
```

```
## 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 <]8;;https://github.com/haleyjeppson/ggmosaichttps://github.com/haleyjep
pson/ggmosaic]8;;>.
```



Investigate if the higher sales are due to there being more customers who buy chips

b. Number of customers by lifestage and segment

```
segment_tranxns <- combined_data %>%
  group_by(lifestage, segment)%>%
  summarise(number = n()) %>%
  arrange(-number)
```

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

```
View(segment_tranxns)
```

Try with wider data

```
segment_tranxns_wider <- segment_tranxns %>%
  pivot_wider(names_from = segment,
              values_from = number)        #longer data has better visibility, sort in descending order

View(segment_tranxns_wider)
```

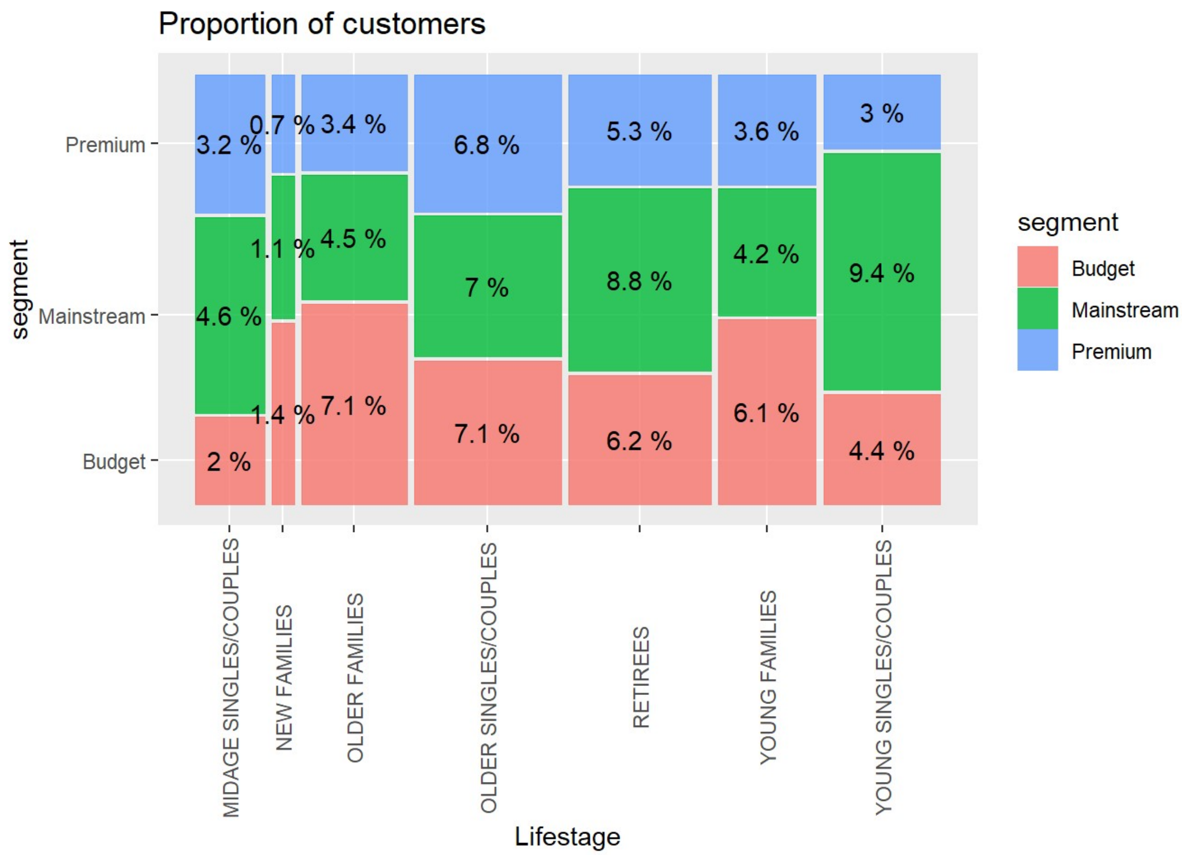
Alternatively use

```
customers <- combined_data[, .(customers = uniqueN(loyalty_card_number)), .(lifestage, segment)][order(-
customers)]
View(customers)
```

Visualize using

```
q <- ggplot(data = customers) +
  geom_mosaic(aes(weight = customers, x = product(segment,lifestage), fill = segment)) +
  labs(x = "Lifestage", y = "segment", title = "Proportion of customers") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))

q + geom_text(data = ggplot_build(q)$data[[1]],
              aes(x = (xmin + xmax)/2 , y = (ymin + ymax)/2, label = as.character(paste(round(.wt/sum(.w
t),3)*100,'%'))))
```



There are more mainstream - YOUNG SINGLES/COUPLES & mainstream RETIREES who buy chips. This contributes to more sales in these 2 categories but doesn't seem to be the main driver of sales in Budget - OLDER FAMILIES.

This implies it's not about having more customers who buy chips. If not, then let's consider quantity bought

c. Consider, average number of units per customer by LIFESTAGE and PREMIUM_CUSTOMER

```
segment_qnty <- combined_data %>%
  group_by(lifestage, segment)%>%
  summarise(avg_qnty = mean(product_quantity)) %>%
  arrange(-avg_qnty)
```

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

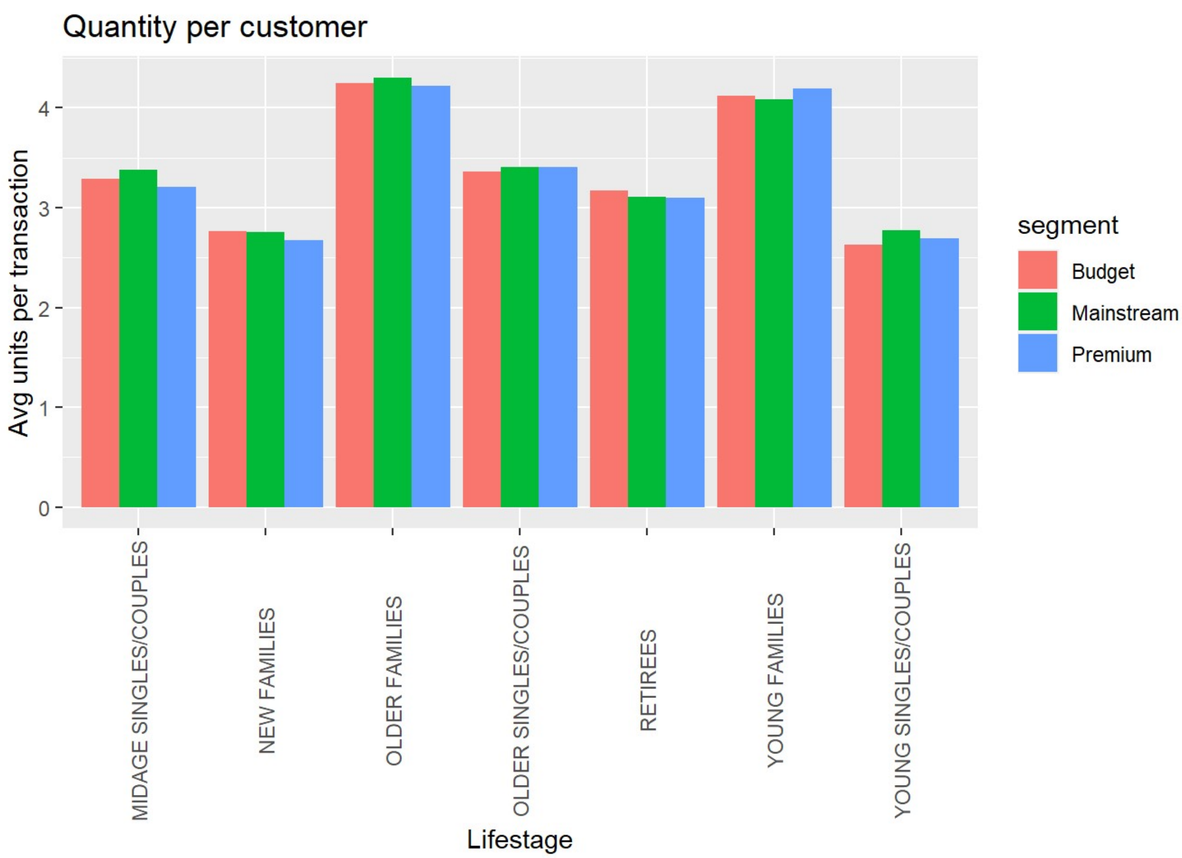
```
View(segment_qnty)
```

Calculate average manually, considering unique cards only

```
segment_qnty2 <- combined_data[, .(avg_qnty = sum(product_quantity)/uniqueN(loyalty_card_number)),.(life
stage, segment)][order(-avg_qnty)]
```

Visualize

```
ggplot(data = segment_qnty2, aes(weight = avg_qnty, x = lifestage, fill = segment)) +
  geom_bar(position = position_dodge()) +
  labs(x = "Lifestage", y = "Avg units per transaction", title = "Quantity per customer") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```



Generally, Older & Young families buy more quantities on average

d. Let's also find out the average price in each customer segment since it's a driver of total sales

```
segment_price_avg <- combined_data %>%
  group_by(lifestage, segment)%>%
  summarise(avg_price = mean(total_sales)) %>%
  arrange(-avg_price)
```

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

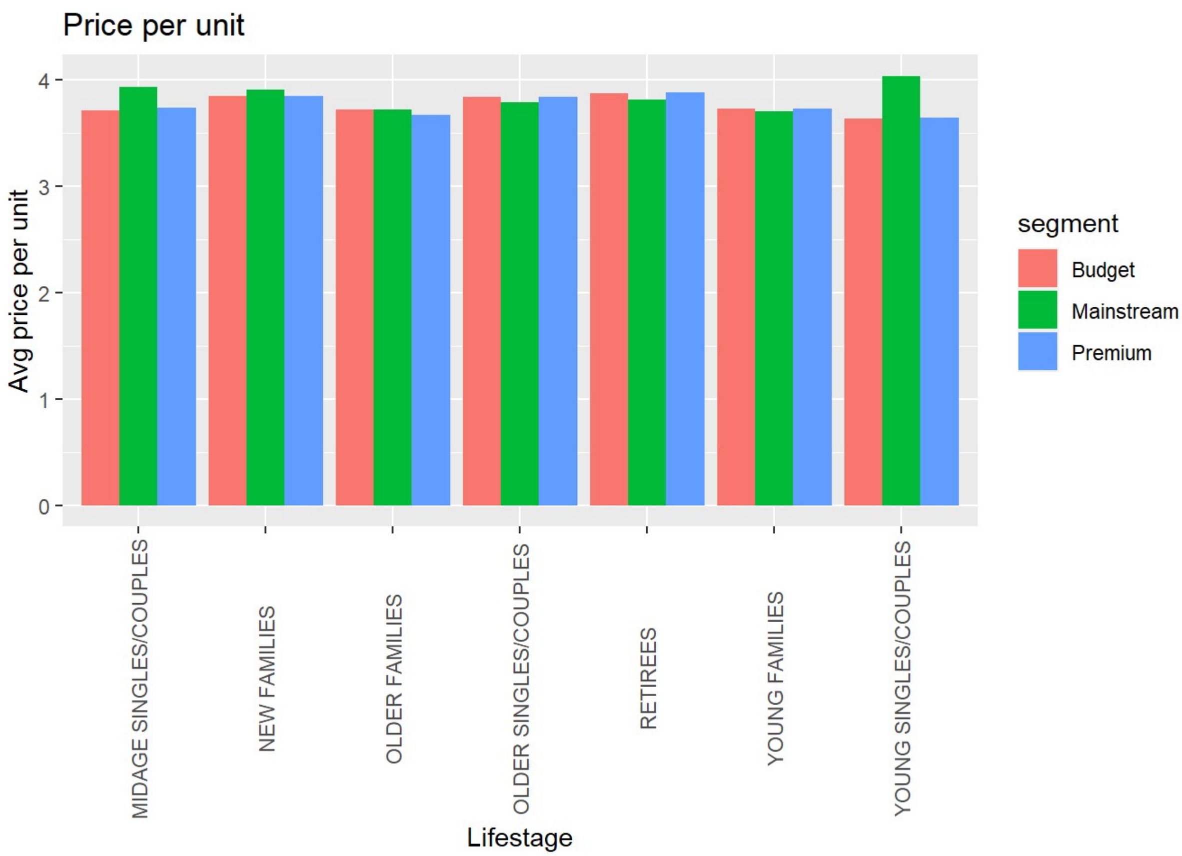
```
View(segment_price_avg)
```

Calculate average price manually

```
segment_price_avg2 <- combined_data[, .(avg_price = sum(total_sales)/sum(product_quantity)), .(lifestag
e, segment)][order(-avg_price)]
```

Visualize using segment_price_avg2

```
ggplot(data = segment_price_avg2,
       aes(weight = avg_price, x = lifestage, fill = segment)) +
  geom_bar(position = position_dodge()) +
  labs(x = "Lifestage", y = "Avg price per unit", title = "Price per unit") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```



On average, Mainstream - YOUNG SINGLES/COUPLES and MIDAGE SINGLES/COUPLES are willing to spend more on a packet of chips, compared to their premium & budget counterparts.

Mainstream could in other words be referred to as middle_income class. Premium and budget refer to high-income and low-income socioeconomic classes respectively

In that case then, this could be explained by the fact that premium customers are more likely to purchase healthier snacks & occasionally they buy chips for “entertainment” purposes. This is also supported by there being fewer Premium - YOUNG SINGLES/COUPLES and MIDAGE SINGLES/COUPLES buying chips compared to their Mainstream counterparts.

We can confirm if price per unit is statistically significant since the difference in avg_price isn’t large.

Statistical analysis

We can perform an independent t-test between mainstream vs premium & budget MIDAGE SINGLES/COUPLES & YOUNG SINGLES/COUPLES

```
pricePerUnit <- combined_data[, price := total_sales/product_quantity]

t.test(combined_data[lifestage %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES") & segment == "M
ainstream", price]
       , combined_data[lifestage %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES") & segment !=
"Mainstream", price]
       , alternative = "greater")
```

```
##
##  Welch Two Sample t-test
##
## data:  combined_data[lifestage %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES") & segment ==
"Mainstream", price] and combined_data[lifestage %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLE
S") & segment != "Mainstream", price]
## t = 21.394, df = 19147, p-value < 2.2e-16
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  0.3059117       Inf
## sample estimates:
## mean of x mean of y
##  3.988826  3.657433
```

The t-test results in a p-value < 2.2e-16 which is statistically significant. The t-test is used to test the hypothesis that unit price for Mainstream, YOUNG SINGLES/COUPLES and MIDAGE SINGLES/COUPLES is significantly higher than that of Budget or Premium, YOUNG SINGLES/COUPLES and MIDAGE SINGLES/COUPLES.

We might want to target customer segments that contribute the most sales e.g Mainstream, YOUNG SINGLES/COUPLES. Let's focus on that category since they featured highly in 3/4 metrics i.e. their proportion of sales, customers and price per unit.

```
mainstream_YSC <- combined_data[lifestage == "YOUNG SINGLES/COUPLES" & segment == "Mainstream",]
other_segments <- combined_data[!(lifestage == "YOUNG SINGLES/COUPLES" & segment == "Mainstream"),]
```

a. which brands do they tend to buy most?

```
MYSG_brand <- combined_data %>%
  filter(segment == "Mainstream" & lifestage == "YOUNG SINGLES/COUPLES") %>%
  group_by(brand_name) %>%
  summarise(number = n()) %>%
  arrange(-number)

View(MYSG_brand)
```

We can see that Doritos Corn Chips tops the list followed by Thins Chips We can also use the affinity analysis or a-priori analysis to find out their most preferred brand **

```
quantity_MYSG <- mainstream_YSC[, sum(product_quantity)]
print(quantity_MYSG)
```

```
## [1] 12185
```

```
quantity_others <- other_segments[, sum(product_quantity)]
print(quantity_others)
```

```
## [1] 148262
```

```
quantity_MYSG_by_brand <- mainstream_YSC[, .(MYSG = sum(product_quantity)/quantity_MYSG), by = brand_nam
e]
quantity_other_by_brand <- other_segments[, .(other = sum(product_quantity)/quantity_others), by = brand
_name]
```

```
brand_proportions <- merge(quantity_MYSG_by_brand, quantity_other_by_brand)[, affinityToBrand := MYSG/ot
her]
brand_proportions[order(-affinityToBrand)]
```

```
##               brand_name       MYSG      other affinityToBrand
## 1:    Doritos Corn Chips 0.27566680 0.22351648       1.2333176
## 2: Kettle Tortilla Chips 0.13746410 0.11249680       1.2219379
## 3:             Cobs Popd 0.13270414 0.11435162       1.1604920
## 4:           Thins Chips 0.17948297 0.16688025       1.0755195
## 5:       Smiths Crinkles 0.16979893 0.20187236       0.8411203
## 6:          French Fries 0.01173574 0.01686204       0.6959858
## 7:      Natural Chips Co 0.04743537 0.07315428       0.6484292
## 8:                    WW 0.04571194 0.09086617       0.5030689
```

We see that: - Mainstream YOUNG SINGLES/COUPLES are 23% more likely to purchase Doritos Corn Chips compared to the other segments - Mainstream YOUNG SINGLES/COUPLES are 50% less likely to purchase WW Chips compared to the other segments

b. Which package_size do they tend to buy most?

```
MYSG_size <- combined_data %>%
  filter(segment == "Mainstream" & lifestage == "YOUNG SINGLES/COUPLES") %>%
  group_by(package_size) %>%
  summarise(number = n()) %>%
  arrange(-number)

View(MYSG_size)
```

We can see that 175g tops the list followed by 150g

Conlcusion

In summary, we've noted the following: i) Larger proportions of sales are from the Budget - OLDER FAMILIES, Mainstream - YOUNG SINGLES/COUPLES, and Mainstream - RETIREES customers. ii) The high sales proportion by Mainstream - YOUNG SINGLES/COUPLES is due to there being more of them compared to other buyers. iii) Mainstream - YOUNG SINGLES/COUPLES are also likely to pay more per packet of chips compared to other customer categories. This suggests that there could be more impulsive buying among clients in this category. iv) Mainstream - YOUNG SINGLES/COUPLES are 23% more likely to buy Doritos Corn Chips compared to the rest of the population.

Recommendation & next steps:

i. The category manager may strategically place the Doritos Corn Chips near shelves that are most frequented by Mainstream - YOUNG SINGLES/COUPLES, especially the smaller sizes packaged in 150g & 170g.
ii. Quantium can help the Category Manager with recommendations of where these shelves are and further help them with measuring the impact of the changed placement.