Transaction and Customer Behavior for Chip Products Analysis

Alif Safwan

2025-01-06

Contents

1 Import the library

```r
library(dplyr) #manipulate data frame
library(ggplot2) #visualization
library(readxl) #read excel file
library(stringr) #manipulate string
library(tidyr) #reshape data frame format
library(tidytext) #text mining
library(car) #levene Test
library(PMCMRplus) # Post Hoc Test for Anova with unequal variance cases
library(arules) # Apriori algorithm
library(arulesViz) # Apriori algorithm visualisation
```

2 Import the Transation and Customer Behavior Data

```r
#from excel file
Trans = read_excel("C:/Users/User/Documents/Project Forage/Quantium/QVI_transaction_data.xlsx"
```

```r
#from csv file
Cust = read.csv("C:/Users/User/Documents/Project Forage/Quantium/QVI_purchase_behaviour.csv")
```

3 Exploratory Data Analysis for Transaction Data

```r
str(Trans)
```

```
## tibble [264,836 x 8] (S3: tbl_df/tbl/data.frame)
##  $ DATE          : num [1:264836] 43390 43599 43605 43329 43330 ...
##  $ STORE_NBR     : num [1:264836] 1 1 1 2 2 4 4 4 5 7 ...
##  $ LYLTY_CARD_NBR: num [1:264836] 1000 1307 1343 2373 2426 ...
##  $ TXN_ID        : num [1:264836] 1 348 383 974 1038 ...
##  $ PROD_NBR      : num [1:264836] 5 66 61 69 108 57 16 24 42 52 ...
##  $ PROD_NAME     : chr [1:264836] "Natural Chip        Compny SeaSalt175g" "CCs Nacho Chees
##  $ 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 ...
```

There are 8 variables and 264,836 observations in transaction data. 1) DATE : The date of transaction occurred 2) STORE_NBR : The unique ID assigned to each store 3) LYLTY_CARD_NBR : The unique ID for each customer 4) TXN_ID : The unique ID for each transaction 5) PROD_NBR : The unique ID for each chip product 6) PROD_NAME : The name of the chip product 7) PROD_QTY : The quantity purchased by the customer per transaction 8) TOT_SALES : The total price per transaction for chip product

Based on DATE variable, the date of the transaction is stored as an integer. Therefore, for better readability, we need to convert it to a date data type. Only two variables, PROD_QTY and TOT_SALES are true numerical variables. The rest may be stored as numeric data types, but they are actually just ID numbers.

```r
# csv and excel store dates as integers, where ) represents December 30, 1899
Trans <- Trans %>%
  mutate(DATE = as.Date(DATE, origin = "1899-12-30"))
```

```r
sum(duplicated(Trans))
```

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

It seems there is one observation that entirely duplicated across all variables.

```
Trans %>%
  filter(duplicated(.))
```

```
## # A tibble: 1 x 8
##   DATE       STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR PROD_NAME          PROD_QTY
##   <date>         <dbl>          <dbl>  <dbl>    <dbl> <chr>                 <dbl>
## 1 2018-10-01       107         107024 108462       45 Smiths Thinly Cu~        2
## # i 1 more variable: TOT_SALES <dbl>
```

```
Trans %>%
  filter(TXN_ID == 108462)
```

```
## # A tibble: 3 x 8
##   DATE       STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR PROD_NAME          PROD_QTY
##   <date>         <dbl>          <dbl>  <dbl>    <dbl> <chr>                 <dbl>
## 1 2018-10-01       107         107024 108462       45 Smiths Thinly Cu~        2
## 2 2018-10-01       107         107024 108462       18 Cheetos Chs & Ba~        2
## 3 2018-10-01       107         107024 108462       45 Smiths Thinly Cu~        2
## # i 1 more variable: TOT_SALES <dbl>
```

```
Trans <- Trans %>%
  distinct()
```

Next, we want to know whether each observation represents a unique transaction.

```
nrow(Trans) == n_distinct(Trans$TXN_ID)
```

```
## [1] FALSE
```

The result shows FALSE, indicating that the number of unique TXN_IDs is less than the number of observations, meaning that different products were purchased in the same transaction.

```
Trans %>%
  group_by(TXN_ID) %>%
  summarise(unique_cust = n_distinct(LYLTY_CARD_NBR)) %>%
  filter(unique_cust > 1)
```

```
## # A tibble: 2 x 2
##   TXN_ID unique_cust
##    <dbl>       <int>
## 1 155468           2
## 2 155469           2
```

There are two 'TXN_ID's with more than one customer.

```r
Trans %>%
  filter(TXN_ID == 155468)
```

```
## # 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 2018-11-16       155         155072 155468       29 French Fries Pot~        2
## 2 2018-08-13       155         155010 155468       10 RRD SR Slow Rst ~        2
## # i 1 more variable: TOT_SALES <dbl>
```

```r
Trans %>%
  filter(TXN_ID == 155469)
```

```
## # 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 2018-11-30       155         155072 155469      110 WW Original Corn~        2
## 2 2019-06-05       155         155010 155469       77 Doritos Corn Chi~        2
## # i 1 more variable: TOT_SALES <dbl>
```

Its look like both transactions occurred at STORE_NBR 155 and involved the same two customers.Perhaps by looking into the purchas history of these customers based on their 'LYLTY_CARD_NBR's we can gain more insight.

```r
Trans %>%
  filter(LYLTY_CARD_NBR == 155072)
```

```
## # A tibble: 4 x 8
##   DATE       STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR PROD_NAME         PROD_QTY
##   <date>         <dbl>          <dbl>  <dbl>    <dbl> <chr>                <dbl>
## 1 2018-07-19       155         155072 155466       52 Grain Waves Sour~        2
## 2 2018-11-15       155         155072 155467      105 Woolworths Chees~        2
## 3 2018-11-16       155         155072 155468       29 French Fries Pot~        2
## 4 2018-11-30       155         155072 155469      110 WW Original Corn~        2
## # i 1 more variable: TOT_SALES <dbl>
```

```r
Trans %>%
  filter(LYLTY_CARD_NBR == 155010) %>%
  arrange(DATE)
```

```
## # A tibble: 8 x 8
##   DATE       STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR PROD_NAME         PROD_QTY
##   <date>         <dbl>          <dbl>  <dbl>    <dbl> <chr>                <dbl>
## 1 2018-08-13       155         155010 155468       10 RRD SR Slow Rst ~        2
## 2 2019-01-15       155         155010 155061       47 Doritos Corn Chi~        2
## 3 2019-02-03       155         155010 155060        2 Cobs Popd Sour C~        2
## 4 2019-02-20       155         155010 155062      103 RRD Steak &     ~        2
## 5 2019-03-24       155         155010 155063       49 Infuzions SourCr~        2
## 6 2019-04-02       155         155010 155064       88 Kettle Honey Soy~        2
## 7 2019-04-05       155         155010 155065       70 Tyrrells Crisps ~        2
## 8 2019-06-05       155         155010 155469       77 Doritos Corn Chi~        2
## # i 1 more variable: TOT_SALES <dbl>
```

Since ’TXN_ID’s are uniquely assigned ID, we should leave them unchanged. However, it must be noted that if our analysis involves combining data based on ’TXN_ID, we should consult the client about these transactions ID in the given data to ensure data integrity is maintained.

There is only 1 category variable, which is ‘PROD_NAME’. Let’s examine the unique groups in this variable.

```r
unique(Trans$PROD_NAME)
```

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

In total, there is 114 different products in this variable. 'PROD_NAME' includes both the pack size and brand name. Our goal is to clear and separate these components to ensure they are correctly split and labeled. This will help with further analysis, such as identifying whether any of the products are not a chip products.

```r
Trans <- Trans %>%
  mutate(
  # Extract packed_size (e.g., "175g", "300g")
  packed_size = str_extract(PROD_NAME, "[0-9]+"),
  # Extract product_brand (first word or phrase before product details)
  product_brand = str_extract(PROD_NAME, "^[^0-9]+?\\b"),

  # Remove brand and packed size to isolate product_name
  product_name = str_remove(PROD_NAME, paste0(packed_size, "[a-zA-Z]+\\b")) %>%
    str_remove(product_brand) %>%
    str_squish() # Remove extra spaces
  )
```

In the code above, 'packed_size' is extracted from 'PROD_NAME' by identifying a sequence of digits. Based on the unique product names, it's safe to assume that each 'PROD_NAME' contains only one numeric sequence, which is followed by the character "g" or "G".

As for 'product_brand', all brand names appear at the beginning of the 'PROD_NAME'. Therefore, extracting the first word is a good initial step, but further cleaning is needed.

Lastly, 'product_name' is obtained by removing both the 'packed_size' and 'product_brand' from the original 'PROD_NAME'. However, additional cleaning is necessary.

Further refinement is required for 'product_brand' because some brand names consist of more than one word, whereas our current method extracts only the first word. Also, some brands appear under different variations—such as "Red Rock Deli" and "RRD"—even though they refer to the same product

```
Trans <- Trans %>%
  mutate(product_brand = case_when(
    product_brand == "Natural" ~ "Natural Chip Co.",
    product_brand == "Old" ~ "Old El Paso",
    product_brand == "Grain" ~ "Grain Waves",
    product_brand == "Burger" ~ "Burger Rings",
    product_brand == "Infzns" ~ "Infuzions",
    product_brand == "Red" ~ "Red Rock Deli",
    product_brand == "Dorito" ~ "Doritos",
    product_brand == "Smith" ~ "Smiths",
    product_brand == "GrnWves" ~ "Grain Waves",
    product_brand == "Tyrrells" ~ "Tyrrells Crisps",
    product_brand == "Cobs" ~ "Cobs",
    product_brand == "French" ~ "French Fries",
    product_brand == "RRD" ~ "Red Rock Deli",
    product_brand == "Snbts" ~ "Sunbites",
    product_brand == "NCC" ~ "Natural Chip Co.",
    product_brand == "WW" ~ "Woolworths",
    TRUE ~ product_brand
  ))
unique(Trans$product_brand)
```

```
##  [1] "Natural Chip Co." "CCs"              "Smiths"           "Kettle"
##  [5] "Old El Paso"      "Grain Waves"      "Doritos"          "Twisties"
##  [9] "Woolworths"        "Thins"            "Burger Rings"       "Cheezels"
## [13] "Infuzions"         "Red Rock Deli"    "Pringles"           "Tyrrells Crisps"
## [17] "Cobs"              "French Fries"     "Tostitos"           "Cheetos"
## [21] "Sunbites"
```

After cleaning, there are 21 product brands. Next, we want to look at the list of unique products under each brand.

```r
Trans %>%
  dplyr::select(PROD_NBR,product_brand,product_name) %>%
  filter(product_brand == "Natural Chip Co.") %>%
  unique()
```

```
## # A tibble: 5 x 3
##   PROD_NBR product_brand    product_name
##      <dbl> <chr>            <chr>
## 1        5 Natural Chip Co. Chip Compny SeaSalt
## 2       98 Natural Chip Co. Sour Cream & Garden Chives
## 3      106 Natural Chip Co. ChipCo Hony Soy Chckn
## 4       12 Natural Chip Co. Chip Co Tmato Hrb&Spce
## 5       80 Natural Chip Co. ChipCo Sea Salt & Vinegr
```

```r
Trans %>%
  dplyr::select(PROD_NBR,product_brand,product_name) %>%
  filter(product_brand == "CCs") %>%
  unique()
```

```
## # A tibble: 3 x 3
##   PROD_NBR product_brand product_name
##      <dbl> <chr>         <chr>
## 1       66 CCs           Nacho Cheese
## 2       54 CCs           Original
## 3       91 CCs           Tasty Cheese
```

```r
Trans %>%
  dplyr::select(PROD_NBR,product_brand,product_name) %>%
  filter(product_brand == "Smiths") %>%
  unique()
```

```
## # A tibble: 18 x 3
##    PROD_NBR product_brand product_name
##       <dbl> <chr>         <chr>
##  1       61 Smiths        Crinkle Cut Chips Chicken
##  2       69 Smiths        Chip Thinly S/Cream&Onion
##  3       16 Smiths        Crinkle Chips Salt & Vinegar
##   4        7 Smiths        Crinkle Original
##   5      111 Smiths        Chip Thinly Cut Original
##   6       37 Smiths        Thinly Swt Chli&S/Cream
##   7      107 Smiths        Crinkle Cut French OnionDip
##   8       45 Smiths        Thinly Cut Roast Chicken
##   9       39 Smiths        Crinkle Cut Tomato Salsa
##  10       82 Smiths        Crinkle Cut Mac N Cheese
##  11       73 Smiths        Crinkle Cut Salt & Vinegar
##  12        8 Smiths        Crinkle Cut Chips Original
##  13      100 Smiths        Crinkle Cut Chips Chs&Onion
##  14       14 Smiths        Crnkle Chip Orgnl Big Bag
##  15       79 Smiths        Chip Thinly CutSalt/Vinegr
##  16        1 Smiths        Crinkle Cut Chips Barbecue
##  17       19 Smiths        Crinkle Cut Snag&Sauce
##  18       43 Smiths        Crinkle Cut Bolognese
```

```
Trans %>%
  dplyr::select(PROD_NBR,product_brand,product_name) %>%
  filter(product_brand == "Kettle") %>%
  unique()
```

```
## # A tibble: 13 x 3
##    PROD_NBR product_brand product_name
##       <dbl> <chr>         <chr>
##   1      108 Kettle        Tortilla ChpsHny&Jlpno Chili
##   2      114 Kettle        Sensations Siracha Lime
##   3       32 Kettle        Sea Salt And Vinegar
##   4       46 Kettle        Original
##   5       36 Kettle        Chilli
##   6      102 Kettle        Mozzarella Basil & Pesto
##   7        3 Kettle        Sensations Camembert & Fig
##   8       88 Kettle        Honey Soy Chicken
##   9       89 Kettle        Sweet Chilli And Sour Cream
##  10       63 Kettle        Swt Pot Sea Salt
##  11       17 Kettle        Sensations BBQ&Maple
##  12        9 Kettle        Tortilla ChpsBtroot&Ricotta
##  13       60 Kettle        Tortilla ChpsFeta&Garlic
```

```
Trans %>%
  dplyr::select(PROD_NBR,product_brand,product_name) %>%
  filter(product_brand == "Old El Paso") %>%
  unique()
```

```
## # A tibble: 3 x 3
##   PROD_NBR product_brand product_name
##      <dbl> <chr>         <chr>
## 1       57 Old El Paso   El Paso Salsa Dip Tomato Mild
## 2       65 Old El Paso   El Paso Salsa Dip Chnky Tom Ht
## 3       59 Old El Paso   El Paso Salsa Dip Tomato Med
```

```r
Trans %>%
  dplyr::select(PROD_NBR,product_brand,product_name) %>%
  filter(product_brand == "Grain Waves") %>%
  unique()
```

```
## # A tibble: 3 x 3
##   PROD_NBR product_brand product_name
##      <dbl> <chr>         <chr>
## 1       24 Grain Waves   Waves Sweet Chilli
## 2       52 Grain Waves   Waves Sour Cream&Chives
## 3       84 Grain Waves   Plus Btroot & Chilli Jam
```

```r
Trans %>%
  dplyr::select(PROD_NBR,product_brand,product_name) %>%
  filter(product_brand == "Doritos") %>%
  unique()
```

```
## # A tibble: 10 x 3
##    PROD_NBR product_brand product_name
##       <dbl> <chr>         <chr>
##  1       42 Doritos       Corn Chip Mexican Jalapeno
##  2       93 Doritos       Corn Chip Southern Chicken
##  3       51 Doritos       Mexicana
##  4        4 Doritos       Corn Chp Supreme
##  5      101 Doritos       Salsa Medium
##  6       47 Doritos       Corn Chips Original
##  7       30 Doritos       Corn Chips Cheese Supreme
##  8       77 Doritos       Corn Chips Nacho Cheese
##  9       20 Doritos       Cheese Supreme
## 10       41 Doritos       Salsa Mild
```

```r
Trans %>%
  dplyr::select(PROD_NBR,product_brand,product_name) %>%
  filter(product_brand == "Twisties") %>%
  unique()
```

```
## # A tibble: 3 x 3
##   PROD_NBR product_brand product_name
##      <dbl> <chr>         <chr>
## 1       15 Twisties      Cheese
## 2      113 Twisties      Chicken
## 3       71 Twisties      Cheese Burger
```

```r
Trans %>%
  dplyr::select(PROD_NBR,product_brand,product_name) %>%
  filter(product_brand == "Burger Rings") %>%
  unique()
```

```
## # A tibble: 1 x 3
##   PROD_NBR product_brand product_name
##      <dbl> <chr>         <chr>
## 1       94 Burger Rings  Rings
```

```r
Trans %>%
  dplyr::select(PROD_NBR,product_brand,product_name) %>%
  filter(product_brand == "Thins") %>%
  unique()
```

```
## # A tibble: 5 x 3
##   PROD_NBR product_brand product_name
##      <dbl> <chr>         <chr>
## 1       44 Thins         Chips Light& Tangy
## 2       22 Thins         Chips Originl saltd
## 3       40 Thins         Chips Seasonedchicken
## 4       28 Thins         Potato Chips Hot & Spicy
## 5       78 Thins         Chips Salt & Vinegar
```

```r
Trans %>%
  dplyr::select(PROD_NBR,product_brand,product_name) %>%
  filter(product_brand == "Cheezels") %>%
  unique()
```

```
## # A tibble: 2 x 3
##   PROD_NBR product_brand product_name
##      <dbl> <chr>         <chr>
## 1       56 Cheezels      Cheese Box
## 2       23 Cheezels      Cheese
```

```r
Trans %>%
  dplyr::select(PROD_NBR,product_brand,product_name) %>%
  filter(product_brand == "Infuzions") %>%
  unique()
```

```
## # A tibble: 5 x 3
##   PROD_NBR product_brand product_name
##      <dbl> <chr>         <chr>
## 1       31 Infuzions     Crn Crnchers Tangy Gcamole
## 2      104 Infuzions     Thai SweetChili PotatoMix
## 3       87 Infuzions     BBQ Rib Prawn Crackers
## 4       49 Infuzions     SourCream&Herbs Veg Strws
## 5       38 Infuzions     Mango Chutny Papadums
```

```
Trans %>%
  dplyr::select(PROD_NBR,product_brand,product_name) %>%
  filter(product_brand == "Red Rock Deli") %>%
  unique()
```

```
## # A tibble: 12 x 3
##    PROD_NBR product_brand product_name
##       <dbl> <chr>         <chr>
##  1       13 Red Rock Deli Rock Deli Thai Chilli&Lime
##  2       64 Red Rock Deli Rock Deli SR Salsa & Mzzrlla
##  3       48 Red Rock Deli Rock Deli Sp Salt & Truffle
##  4       67 Red Rock Deli Chilli& Coconut
##  5       53 Red Rock Deli Sweet Chilli & Sour Cream
##  6      103 Red Rock Deli Steak & Chimuchurri
##  7       85 Red Rock Deli Honey Soy Chicken
##  8       97 Red Rock Deli Salt & Vinegar
##  9        6 Red Rock Deli Lime & Pepper
## 10       58 Red Rock Deli Rock Deli Chikn&Garlic Aioli
## 11       10 Red Rock Deli SR Slow Rst Pork Belly
## 12       11 Red Rock Deli Pc Sea Salt
```

```
Trans %>%
  dplyr::select(PROD_NBR,product_brand,product_name) %>%
  filter(product_brand == "Pringles") %>%
  unique()
```

```
## # A tibble: 8 x 3
##   PROD_NBR product_brand product_name
##      <dbl> <chr>         <chr>
## 1       99 Pringles      Sthrn FriedChicken
## 2       26 Pringles      Sweet&Spcy BBQ
## 3       25 Pringles      SourCream Onion
## 4       81 Pringles      Original Crisps
## 5       68 Pringles      Chicken Salt Crips
## 6      109 Pringles      Barbeque
## 7       62 Pringles      Mystery Flavour
## 8       34 Pringles      Slt Vingar
```

```
Trans %>%
  dplyr::select(PROD_NBR,product_brand,product_name) %>%
  filter(product_brand == "Tyrrells Crisps") %>%
  unique()
```

```
## # A tibble: 2 x 3
##   PROD_NBR product_brand   product_name
##      <dbl> <chr>           <chr>
## 1       70 Tyrrells Crisps Crisps Lightly Salted
## 2      112 Tyrrells Crisps Crisps Ched & Chives
```

```
Trans %>%
  dplyr::select(PROD_NBR,product_brand,product_name) %>%
  filter(product_brand == "Cobs") %>%
  unique()
```

```
## # A tibble: 3 x 3
##   PROD_NBR product_brand product_name
##      <dbl> <chr>         <chr>
## 1       33 Cobs          Popd Swt/Chlli &Sr/Cream Chips
## 2       75 Cobs          Popd Sea Salt Chips
## 3        2 Cobs          Popd Sour Crm &Chives Chips
```

```
Trans %>%
  dplyr::select(PROD_NBR,product_brand,product_name) %>%
  filter(product_brand == "Woolworths") %>%
  unique()
```

```
## # A tibble: 10 x 3
##    PROD_NBR product_brand product_name
##       <dbl> <chr>         <chr>
##  1       92 Woolworths    Crinkle Cut Chicken
##  2       35 Woolworths    Mild Salsa
##  3      110 Woolworths    Original Corn Chips
##  4       83 Woolworths    D/Style Chip Sea Salt
##  5       96 Woolworths    Original Stacked Chips
##  6       72 Woolworths    Crinkle Cut Original
##  7       76 Woolworths    Medium Salsa
##  8      105 Woolworths    Cheese Rings
##  9       27 Woolworths    Supreme Cheese Corn Chips
## 10       21 Woolworths    Sour Cream &OnionStacked Chips
```

```
Trans %>%
  dplyr::select(PROD_NBR,product_brand,product_name) %>%
  filter(product_brand == "French Fries") %>%
  unique()
```

```
## # A tibble: 1 x 3
##   PROD_NBR product_brand product_name
##      <dbl> <chr>         <chr>
## 1       29 French Fries  Fries Potato Chips
```

```
Trans %>%
  dplyr::select(PROD_NBR,product_brand,product_name) %>%
  filter(product_brand == "Tostitos") %>%
  unique()
```

```
## # A tibble: 3 x 3
##   PROD_NBR product_brand product_name
##      <dbl> <chr>         <chr>
## 1       50 Tostitos      Lightly Salted
## 2       74 Tostitos      Splash Of Lime
## 3       90 Tostitos      Smoked Chipotle
```

```
Trans %>%
  dplyr::select(PROD_NBR,product_brand,product_name) %>%
  filter(product_brand == "Cheetos") %>%
  unique()
```

```
## # A tibble: 2 x 3
##   PROD_NBR product_brand product_name
##      <dbl> <chr>         <chr>
## 1       86 Cheetos       Puffs
## 2       18 Cheetos       Chs & Bacon Balls
```

```
Trans %>%
  dplyr::select(PROD_NBR,product_brand,product_name) %>%
  filter(product_brand == "Sunbites") %>%
  unique()
```

```
## # A tibble: 2 x 3
##   PROD_NBR product_brand product_name
##      <dbl> <chr>         <chr>
## 1       55 Sunbites      Whlgrn Crisps Cheddr&Mstrd
## 2       95 Sunbites      Whlegrn Crisps Frch/Onin
```

We rename certain products to proper name.

```
Trans <- Trans %>%
  mutate(product_name = case_when(
    product_name == "Chip Compny SeaSalt" ~ "Sea Salt",
    product_name == "ChipCo Hony Soy Chckn" ~ "Honey Soy Chicken",
    product_name == "Chip Co Tmato Hrb&Spce" ~ "Tomato Herbs & Spices",
    product_name == "ChipCO Sea Salt & Vinegr" ~ "Sea Salt & Vinegar",
    product_name == "Chip Thinly S/Cream&Onion" ~ "Thinly Cut Sour Cream & Onion",
    product_name == "Chip Thinly Cut Original" ~ "Thinly Cut Original",
    product_name == "Thinly Swt Chli&S/Cream" ~ "Thinly Cut Sweet Chilli & Sour Cream",
    product_name == "Crinkle Cut French OnionDip" ~ "Crinkle Cut French Onion Dip",
    product_name == "Crinkle Cut Chips Chs&Onion" ~ "Crinkle Cut Chips Cheese & Onion",
    product_name == "Crnkle Chip Orgnl Big Bag" ~ "Crinkle Chip Original Big Bag",
    product_name == "Chip Thinly CutSalt/Vinegr" ~ "Thinly Cut Salt & Vinegar",
    product_name == "Crinkle Cut Snag&Sauce" ~ "Crinkle Cut Snag & Sauce",
    product_name == "Tortilla ChpsHny&Jlpno Chili" ~ "Tortilla Chips Honey & Jalapeno Chilli",
    product_name == "Swt Pot Sea Salt" ~ "Sweet Potato Sea Salt",
    product_name == "Sensations BBQ&Maple" ~ "Sensations BBQ & Maple",
    product_name == "Tortilla ChpsBtroot&Ricotta" ~ "Tortilla Chips Beetroot & Ricotta",
    product_name == "Tortilla ChpsFeta&Garlic" ~ "Tortilla Chips Feta & Garlic",
    product_name == "El Paso Salsa Dip Tomato Mild" ~ "Salsa Dip Tomato Mild",
    product_name == "El Paso Salsa Dip Chnky Tom Ht" ~ "Salsa Dip Chunky Tomato Hot",
    product_name == "El Paso Salsa Dip Tomato Med" ~ "Salsa Dip Tomato Medium",
    product_name == "Waves Sweet Chilli" ~ "Sweet Chilli",
    product_name == "Waves Sour Cream&Chives" ~ "Sour Cream & Chives",
    product_name == "Plus Btroot & Chilli Jam" ~ "Plus Beetroot & Chilli Jam",
    product_name == "Corn Chp Supreme" ~ "Corn Chip Supreme",
    product_name == "Sour Cream &OnionStacked Chips" ~ "Sour Cream & Onion Stacked Chips",
    product_name == "Chips Light& Tangy" ~ "Chips Light & Tangy",
    product_name == "Chips Originl saltd" ~ "Chips Original salted",
    product_name == "Chips Seasonedchicken" ~ "Chips Seasoned Chicken",
    product_name == "Crn Crnchers Tangy Gcamole" ~ "Corn Crunchers Tangy Gucamole",
    product_name == "Thai SweetChili PotatoMix" ~ "Thai Sweet Chili Potato Mix",
    product_name == "SourCream&Herbs Veg Strws" ~ "Sour Cream & Herbs Veggie Straws",
    product_name == "Mango Chutny Papadums" ~ "Mango Chutney Papadams",
    product_name == "Rock Deli Thai Chilli&Lime" ~ "Thai Chilli & Lime",
    product_name == "Rock Deli SR Salsa & Mzzrlla" ~ "SR Salsa & Mozzarella",
    product_name == "Rock Deli Sp Salt & Truffle" ~ "Sp Salt & Truffle",
    product_name == "Chilli& Coconut" ~ "Chilli & Coconut",
    product_name == "Rock Deli Chikn&Garlic Aioli" ~ "Chicken & Garlic Aioli",
    product_name == "SR Slow Rst Pork Belly" ~ "SR Slow Roast Pork Belly",
    product_name == "Sthrn FriedChicken" ~ "Southern Fried Chicken",
    product_name == "Sweet&Spcy BBQ" ~ "Sweet & Spicy BBQ",
    product_name == "SourCream Onion" ~ "Sour Cream Onion",
    product_name == "Slt Vingar" ~ "Salt Vinegar",
    product_name == "Popd Swt/Chlli &Sr/Cream Chips" ~ "Pop'd Sweet Chilli & Sour Cream Chips"
    product_name == "Popd Sea Salt Chips" ~ "Pop'd Sea Salt Chips",
    product_name == "Popd Sour Crm &Chives Chips" ~ "Pop'd Sour Cream & Chives Chips",
    product_name == "Fries Potato Chips" ~ "French Fries Potato Chips",
    product_name == "Rings" ~ "Burger Rings",
    product_name == "Chs & Bacon Balls" ~ "Cheese & Bacon Balls",
    product_name == "Whlgrn Crisps Cheddr&Mstrd" ~ "Wholegrain Crisps Cheddar & Mustard",
    product_name == "Whlegrn Crisps Frch/Onin" ~ "Wholegrain Crisps French Onion",
    TRUE ~ product_name
  ))
```

After reviewing all the products, we noticed that some of them are not chip products but are actually salsa dip.

```
Trans %>%
  # Separate product_name into individual words
  mutate(product_name = str_replace_all(product_name, "\\s+", " ")) %>%
  unnest_tokens(word, product_name) %>%
  # Group by word and count occurrences
  count(word, sort = TRUE)
```

```
## # A tibble: 124 x 2
##    word         n
##    <chr>    <int>
##  1 chips    59350
##  2 salt     33979
##  3 cheese   30850
##  4 chicken  27667
##  5 crinkle  27193
##  6 cream    26381
##  7 sour     26381
##  8 original 26234
##  9 corn     25207
## 10 cut      25127
## # i 114 more rows
```

Since, “salsa” and “dip” are likely not chip products, we need to review them first. If they are indeed not chips, we should remove them from our dataset.

```
Trans %>%
  dplyr::select(product_brand,product_name) %>%
  filter(str_detect(product_name, regex("salsa|dip",ignore_case = TRUE))) %>% unique()
```

```
## # A tibble: 10 x 2
##    product_brand product_name
##    <chr>         <chr>
##  1 Old El Paso   Salsa Dip Tomato Mild
##  2 Red Rock Deli SR Salsa & Mozzarella
##  3 Smiths        Crinkle Cut French Onion Dip
##  4 Smiths        Crinkle Cut Tomato Salsa
##  5 Doritos       Salsa Medium
##  6 Old El Paso   Salsa Dip Chunky Tomato Hot
##  7 Woolworths    Mild Salsa
##  8 Old El Paso   Salsa Dip Tomato Medium
##  9 Woolworths    Medium Salsa
## 10 Doritos       Salsa Mild
```

Some products still qualify as chip products even though their names contain “salsa” or “dip”. However, the rest are not chip products. Therefore, we will remove the non-chip products from our dataset.

```
Trans <- Trans %>%
  filter(!(product_name %in% c("Salsa Dip Tomato Mild","Salsa Medium","Salsa Dip Chunky Tomato
```

```
unique(Trans$product_brand)
```

```
##  [1] "Natural Chip Co." "CCs"              "Smiths"           "Kettle"
##  [5] "Grain Waves"      "Doritos"          "Twisties"         "Woolworths"
##  [9] "Thins"            "Burger Rings"     "Cheezels"         "Infuzions"
## [13] "Red Rock Deli"    "Pringles"         "Tyrrells Crisps"  "Cobs"
## [17] "French Fries"     "Tostitos"         "Cheetos"          "Sunbites"
```

now that we've remove the non-chip products, we can see here only 20 product brands remain.

```
str(Trans$packed_size)
```

```
##  chr [1:251160] "175" "175" "170" "175" "150" "330" "210" "150" "210" "330" ...
```

```
unique(Trans$packed_size)
```

```
##  [1] "175" "170" "150" "330" "210" "270" "220" "125" "110" "134" "380" "180"
## [13] "165" "135" "250" "300" "200" "160" "190" "90"  "70"
```

The 'packed_size' is stored as a character type because we extracted it as a string. There are 21 different pack sizes across all products.

```
colSums(is.na(Trans))
```

```
##           DATE      STORE_NBR LYLTY_CARD_NBR         TXN_ID       PROD_NBR
##              0              0              0              0              0
##      PROD_NAME       PROD_QTY      TOT_SALES    packed_size  product_brand
##              0              0              0              0              0
##   product_name
##              0
```

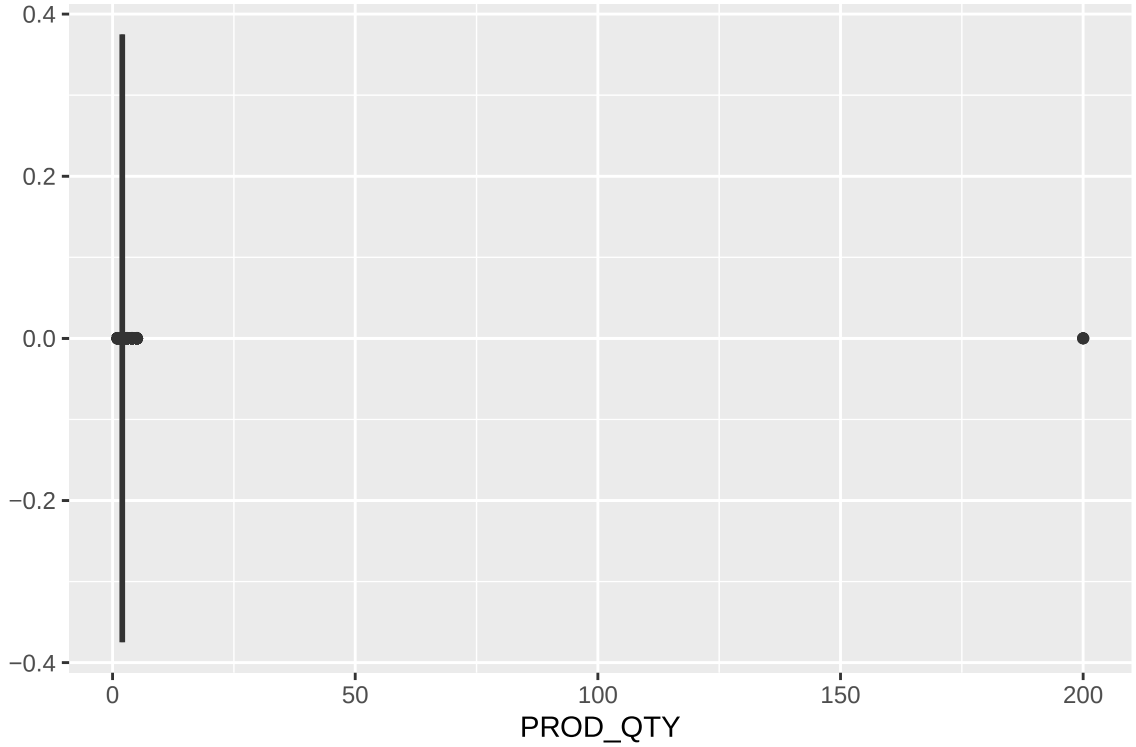
There are no missing values in any of the columns. Next, we check a simple summary for each column.

```
summary(Trans)
```

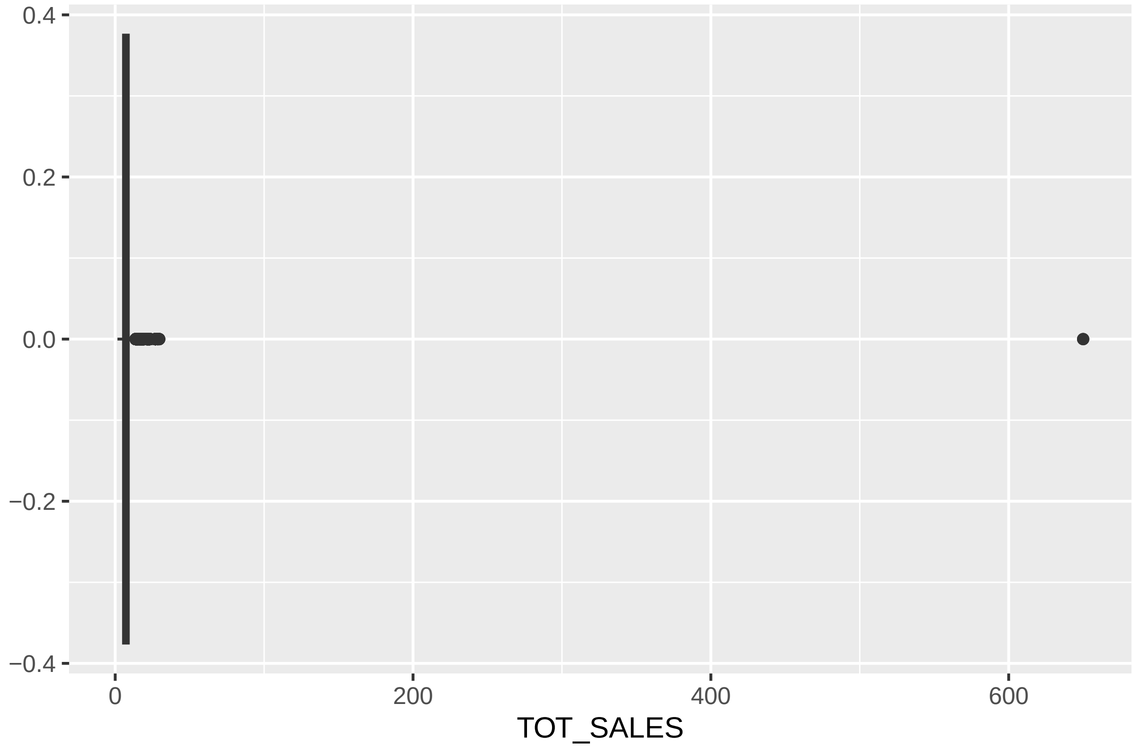
```
##       DATE              STORE_NBR     LYLTY_CARD_NBR        TXN_ID
##  Min.   :2018-07-01   Min.   :  1   Min.   :   1000   Min.   :      1
##  1st Qu.:2018-09-30   1st Qu.: 70   1st Qu.:  70017   1st Qu.:  67576
##  Median :2018-12-30   Median :130   Median : 130352   Median : 135111
##  Mean   :2018-12-30   Mean   :135   Mean   : 135507   Mean   : 135112
##  3rd Qu.:2019-03-31   3rd Qu.:203   3rd Qu.: 203076   3rd Qu.: 202619
##  Max.   :2019-06-30   Max.   :272   Max.   :2373711   Max.   :2415841
##     PROD_NBR       PROD_NAME             PROD_QTY         TOT_SALES
##  Min.   :  1.00   Length:251160      Min.   :  1.000   Min.   :  1.500
##  1st Qu.: 27.00   Class :character   1st Qu.:  2.000   1st Qu.:  5.600
##  Median : 52.00   Mode  :character   Median :  2.000   Median :  7.400
##  Mean   : 56.17                      Mean   :  1.908   Mean   :  7.268
##  3rd Qu.: 86.00                      3rd Qu.:  2.000   3rd Qu.:  8.800
##  Max.   :114.00                      Max.   :200.000   Max.   :650.000
##  packed_size        product_brand       product_name
##  Length:251160      Length:251160      Length:251160
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
```

As we see, the maximum values for 'PROD_QTY' and 'TOT_SALES' are quite extreme. So, let's take a look at the box plots for both of them.

```
Trans %>%
  ggplot(aes(x = PROD_QTY)) +
  geom_boxplot()
```



```
Trans %>%
  ggplot(aes(x = TOT_SALES)) +
  geom_boxplot()
```



From the box plots, we can see that the extreme values are far away from the rest, they clearly stand out as outliers. So, let's take a closer look at these outliers.

```
Trans %>%
  filter(PROD_QTY == 200 | TOT_SALES == 650)
```

```
## # A tibble: 2 x 11
##   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 4 more variables: TOT_SALES <dbl>, packed_size <chr>, product_brand <chr>,
## #   product_name <chr>
```

The result shows only two rows with extreme outliers, and both transactions were made by same loyalty card number.

```
Trans %>%
  filter(LYLTY_CARD_NBR == 226000)
```

```
## # A tibble: 2 x 11
##   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 4 more variables: TOT_SALES <dbl>, packed_size <chr>, product_brand <chr>,
## #   product_name <chr>
```

Based on output, this loyalty number made only those two outlier transactions. It’s possible that this person purchased for commercial purposes. So, we will exclude this customer from our dataset.

```r
Trans <- Trans %>%
  filter(LYLTY_CARD_NBR != 226000)
```

```r
summary(Trans)
```

```
##       DATE              STORE_NBR   LYLTY_CARD_NBR       TXN_ID
##  Min.   :2018-07-01   Min.   :  1   Min.   :   1000   Min.   :      1
##  1st Qu.:2018-09-30   1st Qu.: 70   1st Qu.:  70017   1st Qu.:  67575
##  Median :2018-12-30   Median :130   Median : 130352   Median : 135110
##  Mean   :2018-12-30   Mean   :135   Mean   : 135506   Mean   : 135111
##  3rd Qu.:2019-03-31   3rd Qu.:203   3rd Qu.: 203076   3rd Qu.: 202619
##  Max.   :2019-06-30   Max.   :272   Max.   :2373711   Max.   :2415841
##     PROD_NBR       PROD_NAME           PROD_QTY       TOT_SALES
##  Min.   :  1.00   Length:251158      Min.   :1.000   Min.   : 1.500
##  1st Qu.: 27.00   Class :character   1st Qu.:2.000   1st Qu.: 5.600
##  Median : 52.00   Mode  :character   Median :2.000   Median : 7.400
##  Mean   : 56.17                      Mean   :1.906   Mean   : 7.262
##  3rd Qu.: 86.00                      3rd Qu.:2.000   3rd Qu.: 8.800
##  Max.   :114.00                      Max.   :5.000   Max.   :29.500
##  packed_size        product_brand      product_name
##  Length:251158      Length:251158      Length:251158
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
```

By looking at the updated summary, it seems there are no longer any outliers in ‘PROD_QTY’ and “TOT_SALES.

Next, we want to check whether transactions occurred every day between 2018-07-01 and 2019-06-30. Specifically, how many days had no chip transactions?

```r
Trans %>%
  group_by(DATE) %>%
  summarise(total_trans_each_day = n()) %>%
  right_join(
    data.frame(DATE = seq.Date(from = min(Trans$DATE),to = max(Trans$DATE), by = "day"))) %>%
  filter(is.na(total_trans_each_day))
```

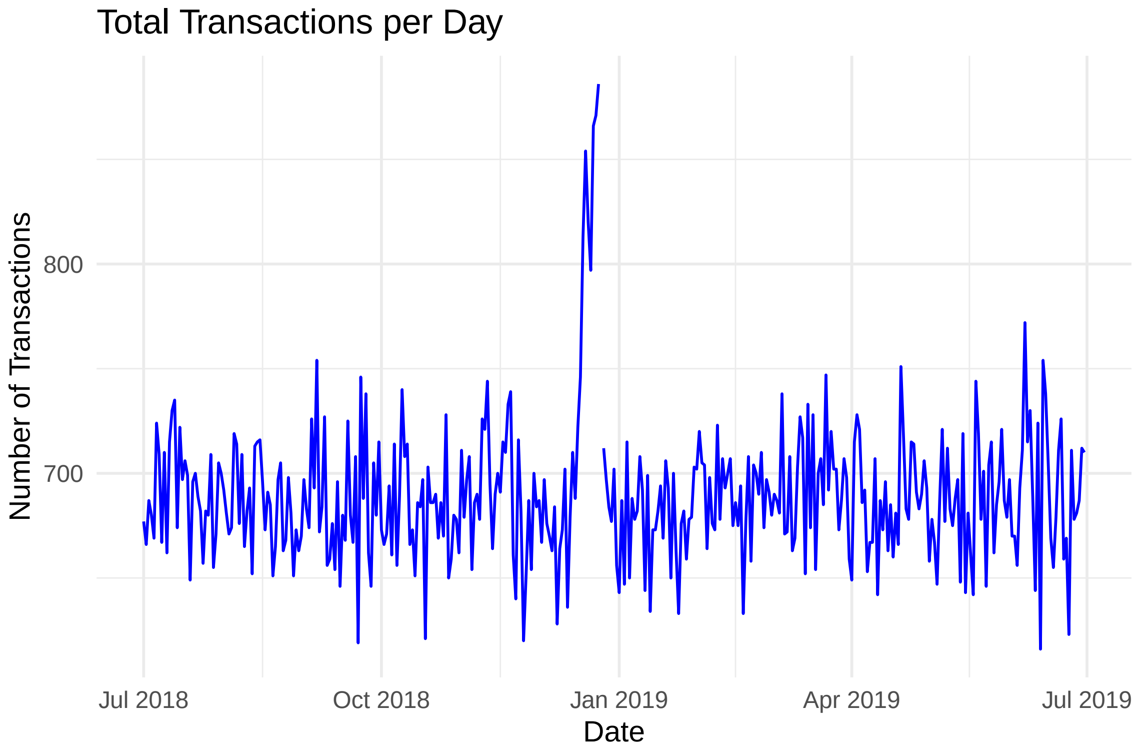
```
## Joining with `by = join_by(DATE)`
```

```
## # A tibble: 1 x 2
##   DATE       total_trans_each_day
##   <date>                    <int>
## 1 2018-12-25                   NA
```

We can see that only 25-12-2018 had no transactions, likely because it was Christmas and the stores were closed. There's a clear gap between 24-12-2018 and 26-12-2018, and transaction volume appears to increase just before Christmas, probably due to holiday preparations.

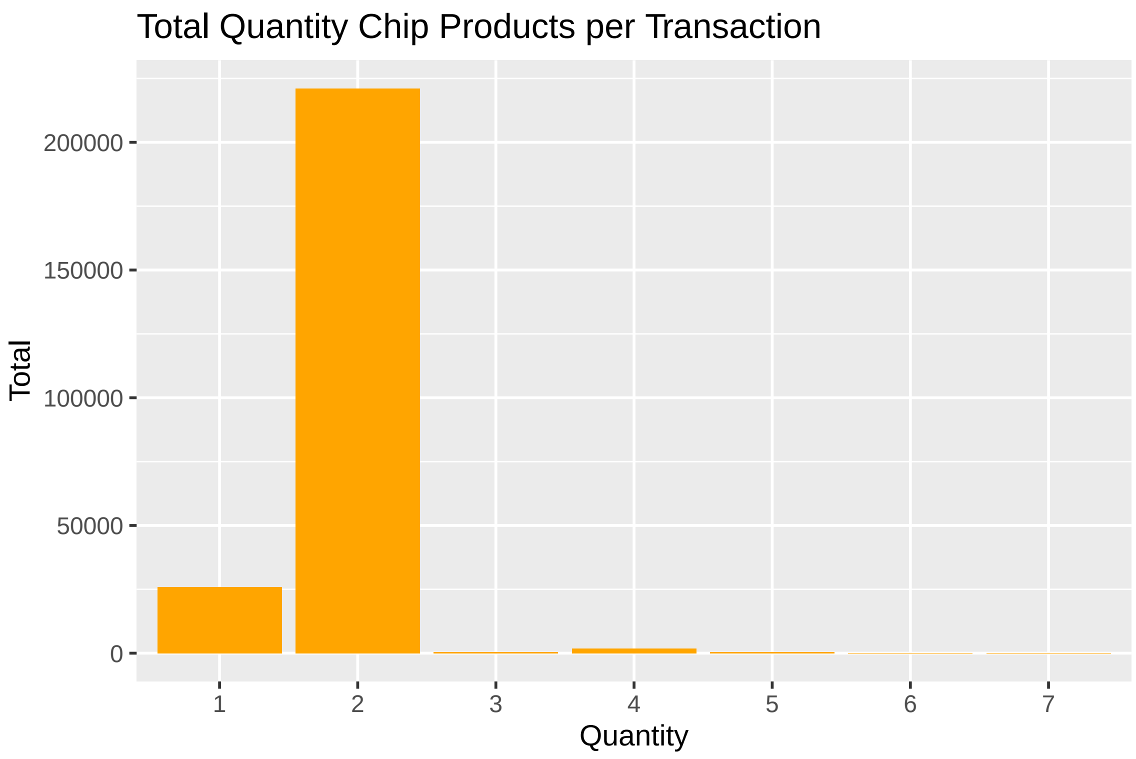
```
Trans %>%
  group_by(DATE) %>%
  summarise(total_trans_each_day = n()) %>%
  right_join(
    data.frame(DATE = seq.Date(from = min(Trans$DATE),to = max(Trans$DATE), by = "day"))) %>%
  ggplot(aes(x = DATE, y = total_trans_each_day)) +
  geom_line(color = "blue") +
  labs(title = "Total Transactions per Day", x = "Date", y = "Number of Transactions") +
  theme_minimal()
```

```
## Joining with `by = join_by(DATE)`
```

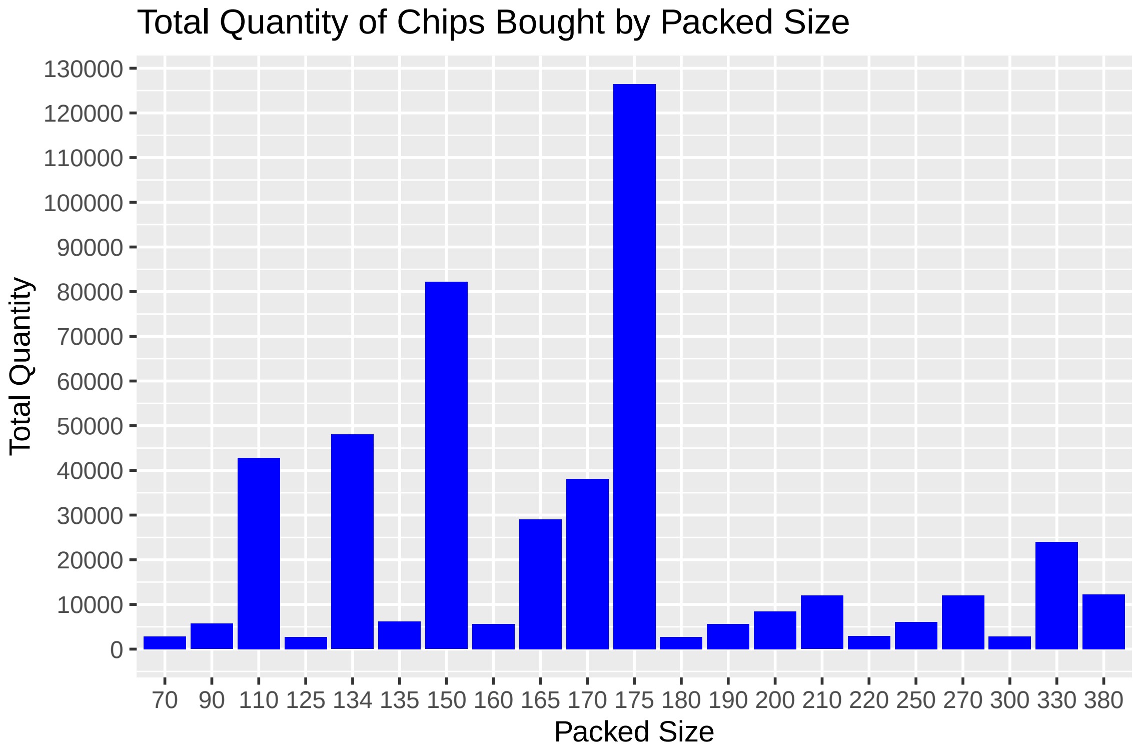


The line plot gives a clear view of daily transaction patterns. You can easily spot the gap on 25-12-2018, and observe some spikes around major dates, like just before Christmas, indicating increased activity likely tied to holidays or events.

```
Trans %>%
  group_by(TXN_ID) %>%
  summarise(Quantity = sum(PROD_QTY)) %>%
  ungroup() %>%
  ggplot(aes(x = as.factor(Quantity))) +
  geom_bar( fill = "orange") +
  labs( title =  "Total Quantity Chip Products per Transaction",
        x = "Quantity",
        y = "Total")
```

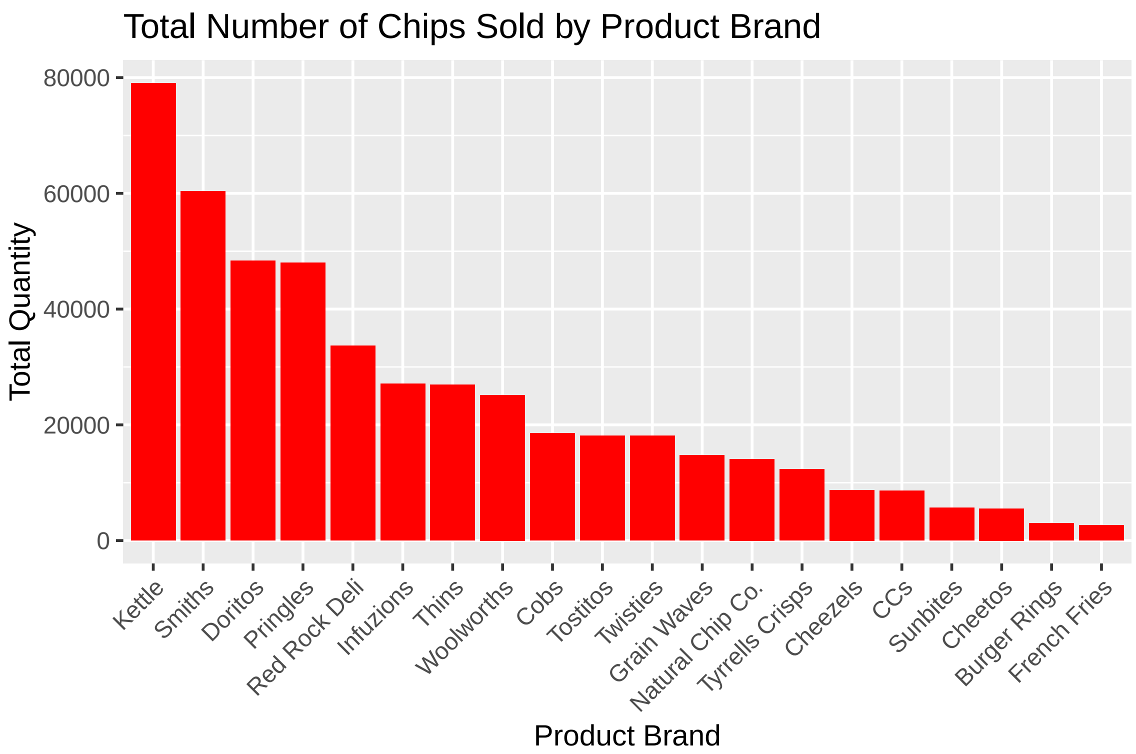


```
Trans %>%
  group_by(packed_size) %>%
  summarise(total_count = sum(PROD_QTY, na.rm = TRUE)) %>%
  ggplot(aes(x = reorder(as.integer(packed_size), as.integer(packed_size)), y= total_count)) +
  geom_bar(stat = "identity", fill = "blue") +
  labs(title = "Total Quantity of Chips Bought by Packed Size",
       x = "Packed Size",
       y = "Total Quantity") +
  scale_y_continuous(breaks = seq(0,1000000, by = 10000))
```



The chart shows that most customers purchased chips in the 175g pack size, followed by the 150g pack size. This suggests that mid-sized packs are the most popular choice among buyers.

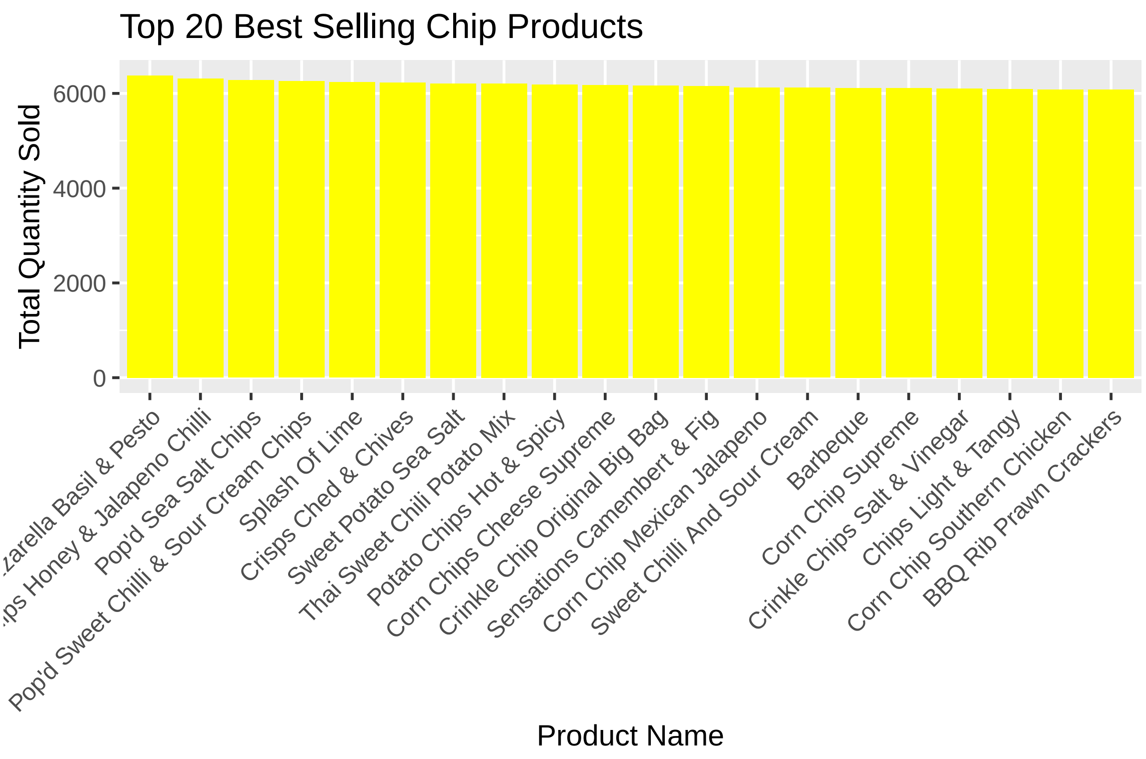
```r
Trans %>%
  group_by(product_brand) %>%
  summarise(total_count = sum(PROD_QTY)) %>%
  ggplot(aes(x = reorder(product_brand, - total_count), y = total_count)) +
  geom_bar(stat = "identity", fill = "red") +
  scale_x_discrete(guide = guide_axis(angle = 45)) +
  labs(title = "Total Number of Chips Sold by Product Brand", y = "Total Quantity", x = "Produ
```



The most popular chip brand is Kettle, with nearly 80,000 units sold, followed by Smiths with close to 60,000 units, and Doritos with almost 50,000 units.

```
Trans %>%
  group_by(product_name, product_brand) %>%
  summarise(total_count = sum(PROD_QTY)) %>%
  ungroup() %>%
  mutate(ranking = rank(desc(total_count))) %>%
  filter(ranking <= 20) %>%
  group_by(product_name, product_brand) %>%
  ggplot(aes(x = reorder(product_name, ranking), y = total_count)) +
  geom_bar(stat = "identity", fill = "yellow") +
  scale_x_discrete(guide = guide_axis(angle = 45)) +
  labs(title = "Top 20 Best Selling Chip Products", y = "Total Quantity Sold", x = "Product Na
```

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



All of the top 20 products have nearly identical sales figures, with each selling around 6,000 units.

We also want to determine whether the price of each product is fixed or varies

```
Trans %>%
  group_by(product_name) %>%
  summarise(avg_price = mean(TOT_SALES/PROD_QTY), sd_price = sd(TOT_SALES/PROD_QTY),
            max_price = max(TOT_SALES/PROD_QTY), min_price = min(TOT_SALES/PROD_QTY)) %>%
  filter(sd_price > 1e-03)
```

```
## # A tibble: 19 x 5
##    product_name                   avg_price sd_price max_price min_price
##    <chr>                              <dbl>    <dbl>     <dbl>     <dbl>
##  1 BBQ Rib Prawn Crackers              3.80   0.0337       3.8      1.9
##  2 Cheese                              5.15   0.550        5.7      4.6
##  3 Chilli                              5.40   0.0490       5.4      2.7
##  4 Chips Salt & Vinegar                3.30   0.0463       3.3      1.32
##  5 Corn Chip Supreme                   6.37   0.641        6.5      3.25
##  6 Corn Chips Cheese Supreme           4.40   0.0388       4.4      2.2
##  7 Crinkle Cut Chips Chicken           2.90   0.0376       2.9      1.45
##  8 Crisps Ched & Chives                4.20   0.0245       4.2      2.8
##  9 Honey Soy Chicken                   4.23   1.20         5.4      3
## 10 Original                            4.33   1.54         5.4      2.1
## 11 Original Crisps                     3.70   0.0329       3.7      1.85
## 12 SR Slow Roast Pork Belly            2.70   0.0346       2.7      1.35
## 13 Sensations Camembert & Fig          4.60   0.0486       4.6      1.84
## 14 Sour Cream & Chives                3.60   0.0215       3.6      2.4
## 15 Southern Fried Chicken             3.70   0.0400       3.7      1.48
## 16 Sweet Chilli                       3.60   0.0384       3.6      1.44
## 17 Sweet Chilli And Sour Cream        5.40   0.0573       5.4      2.16
## 18 Thai Chilli & Lime                 2.70   0.0349       2.7      1.35
## 19 Tortilla Chips Feta & Garlic       4.60   0.0581       4.6      2.3
```

In conclusion, from this EDA of the transaction data, we've uncovered several insights:

1) There was no transaction on December 25th, 2018, likely because stores were closed for the Christmas holiday. However, total sales in the days leading up to it were among the highest, suggesting that chips are a popular item for holiday preparations.

2) Most customers typically purchase 175g chip packs, followed by 150g. Larger pack sizes aren't necessarily the most purchased, and it's common for customers to buy two bags per transaction.

3) Overall, Kettle is the most popular chip brand, followed by Smiths, Doritos, and Pringles.

4) The top 20 chip products each recorded sales around 6,000 units.

5) There are 19 chip products with inconsistent pricing.

4 Exploratory Data Analysis for Customer Behavior Data

```
str(Cust)
```

```
## '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"
##  $ PREMIUM_CUSTOMER: chr  "Premium" "Mainstream" "Budget" "Mainstream" ...
```

There are 3 variables: the loyalty card number (LYLTY_CARD_NBR) for each customer, their lifestage (LIFESTAGE), and their premium customer status (PREMIUM_CUSTOMER).

```
colSums(is.na(Cust))
```

```
##   LYLTY_CARD_NBR        LIFESTAGE PREMIUM_CUSTOMER
##                0                0                0
```

There are no missing values in this dataset. Since it represents customer behavior, each 'LYLTY_CARD_NBR' should ideally be unique.

```
n_distinct(Cust$LYLTY_CARD_NBR) == nrow(Cust)
```

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

Since the result is TRUE, this confirms that each row represents a unique customer.

```
unique(Cust$LIFESTAGE)
```

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

There are 7 unique groups under the 'LIFESTAGE' category.

```
unique(Cust$PREMIUM_CUSTOMER)
```

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

There are 3 distinct groups in the PREMIUM_CUSTOMER category.

In conclusion from this section, we know that:

1) Each row represents a unique customer.
2) There are seven groups in 'LIFESTAGE' category.
3) There are three groups in 'PREMIUM_CUSTOMER' category.

5 Joining the Trans and Cust data frames.

We need to understand customer segmentation for chip products, so we join the Cust data into the Trans data.

```
Chip <- Trans %>%
  left_join(Cust)
```

```
## Joining with `by = join_by(LYLTY_CARD_NBR)`
```

```
ncol(Trans) + ncol(Cust) - 1 == ncol(Chip)
```

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

```r
nrow(Trans) == nrow(Chip)
```

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

The number of columns and rows match, so the left join is correct.

Next, we check for missing values in case any loyalty card numbers in Trans don't exist in Cust.

```r
colSums(is.na(Chip))
```

```
##             DATE        STORE_NBR   LYLTY_CARD_NBR           TXN_ID
##                0                0                0                0
##         PROD_NBR        PROD_NAME         PROD_QTY        TOT_SALES
##                0                0                0                0
##      packed_size    product_brand     product_name        LIFESTAGE
##                0                0                0                0
## PREMIUM_CUSTOMER
##                0
```

There are no missing values, so each customer's transaction data matches perfectly with their behavioral data.

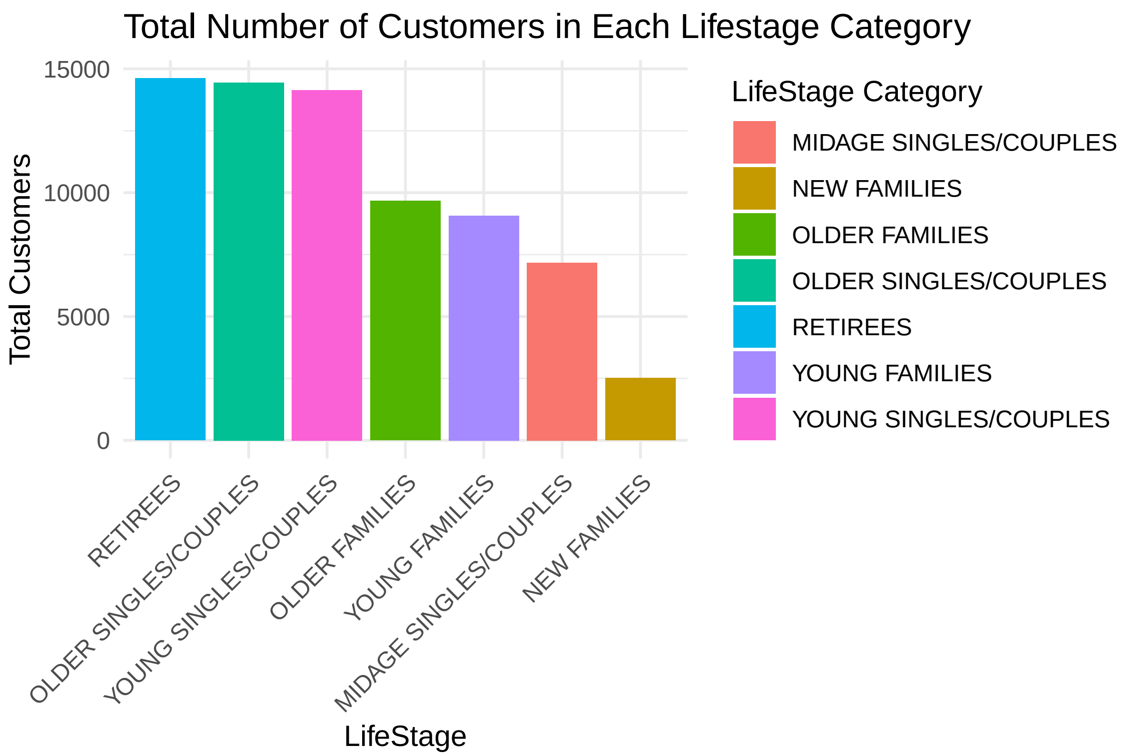
6 Exploratory Data Analysis for Chip Data

Now we want to explore customer behavior.

1) How many customers are in each group of 'LIFESTAGE' category?

```r
Chip %>%
  group_by(LIFESTAGE, LYLTY_CARD_NBR) %>%
  summarise(total_count = n()) %>%
  group_by(LIFESTAGE) %>%
  summarise(total_count = n()) %>%
  ggplot(aes(x = reorder(LIFESTAGE, - total_count), y = total_count, fill = LIFESTAGE)) +
  geom_bar(stat = "identity") +
  scale_x_discrete(guide = guide_axis(angle = 45)) +
  labs(title = "Total Number of Customers in Each Lifestage Category", x = "LifeStage",
       y = "Total Customers", fill = "LifeStage Category") +
  theme_minimal()
```

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

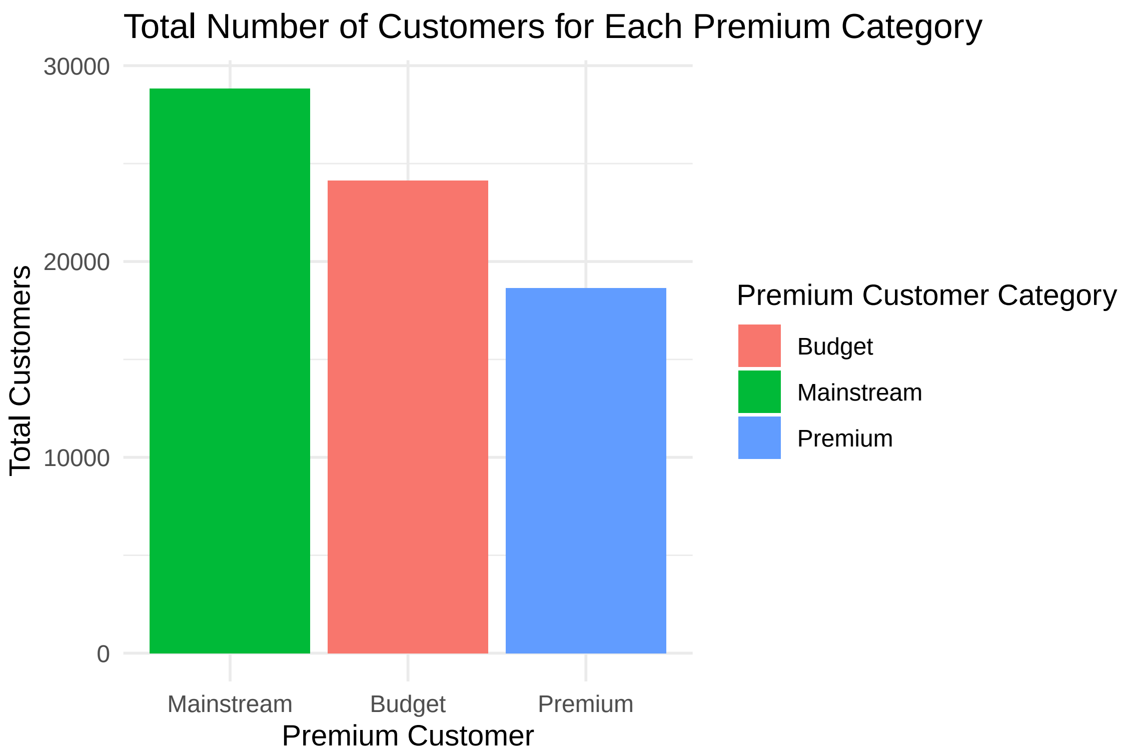


The customers who are retirees, older singles/couples, and young singles/couples are the main groups that purchase the most chip products.

2) How many customers are in each group of the ‘PREMIUM_CUSTOMER’ category?

```
Chip %>%
  group_by(PREMIUM_CUSTOMER, LYLTY_CARD_NBR) %>%
  summarise(total_count = n()) %>%
  group_by(PREMIUM_CUSTOMER) %>%
  summarise(total_count = n()) %>%
  ggplot(aes(x = reorder(PREMIUM_CUSTOMER,-total_count), y = total_count, fill = PREMIUM_CUSTO
  geom_bar(stat = "identity") +
  labs(
    title = "Total Number of Customers for Each Premium Category", y = "Total Customers",
    x =  "Premium Customer", fill = "Premium Customer Category"
  ) +
  theme_minimal()
```

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

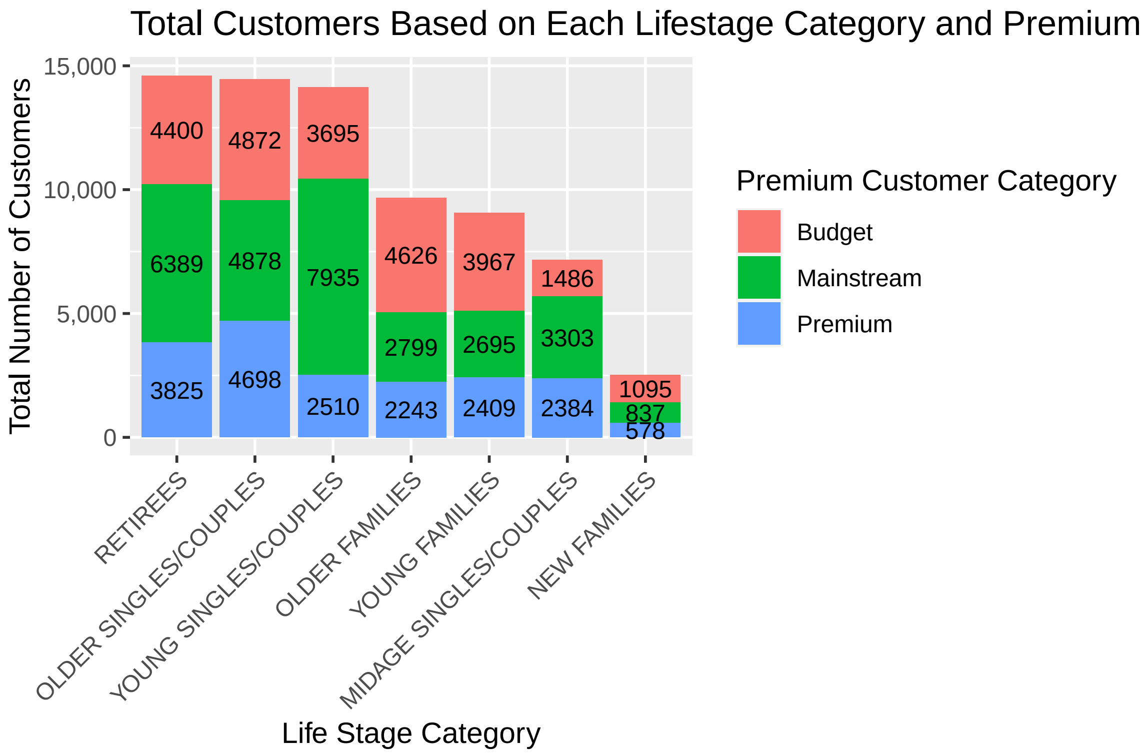


Most of the customers belong to the mainstream group, followed by the budget and premium groups.

Q3) How many customers are in each group of 'LIFESTAGE' and 'PREMIUM_CUSTOMER' category?

```
Chip %>%
  group_by(LIFESTAGE, PREMIUM_CUSTOMER, LYLTY_CARD_NBR) %>%
  summarise(total_count = n()) %>%
  group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarise(total_count = n()) %>%
  ggplot(aes(x = reorder(LIFESTAGE, - total_count), y = total_count, fill = PREMIUM_CUSTOMER))
  geom_bar(stat = "identity") +
  geom_text(aes(label = total_count),
            position = position_stack(vjust = 0.5), size = 3, color = "black") +
  labs(
    title = "Total Customers Based on Each Lifestage Category and Premium Customer",
    x = "Life Stage Category",
    y = "Total Number of Customers",
    fill = "Premium Customer Category"
  ) +
  scale_y_continuous(labels = function(x) format(x, big.mark = ",", scientific = FALSE)) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

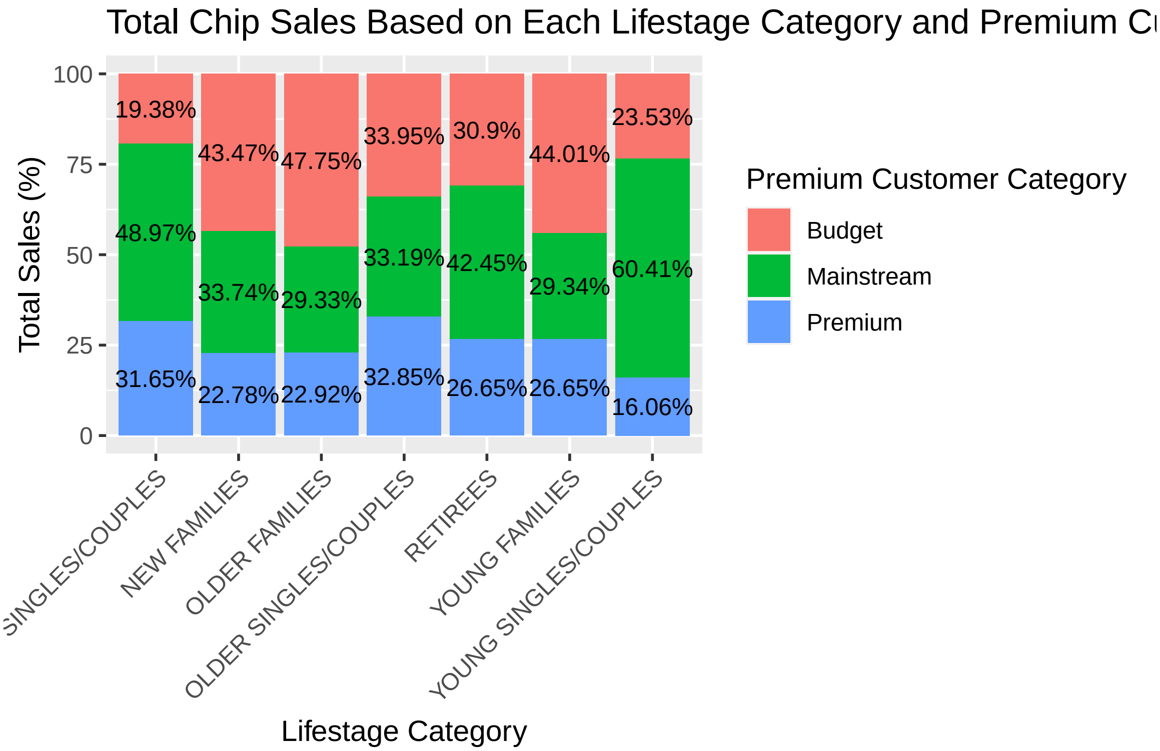
```
## `summarise()` has grouped output by 'LIFESTAGE', 'PREMIUM_CUSTOMER'. You can
## override using the `.groups` argument.
## `summarise()` has grouped output by 'LIFESTAGE'. You can override using the
## `.groups` argument.
```



Q4) What is the Total Chip Sales Based on Each Lifestage Category and Premium Customer?

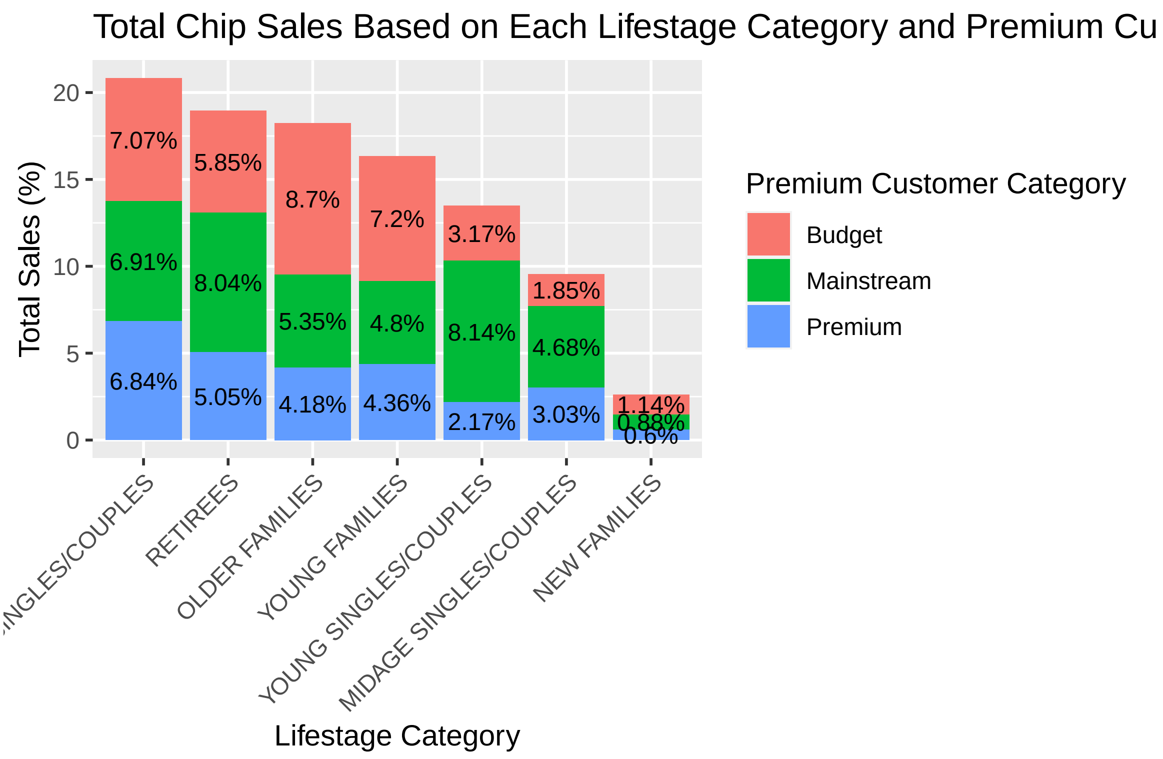
```r
Chip %>%
  group_by(LIFESTAGE,PREMIUM_CUSTOMER) %>%
  summarise(total_sales = sum(TOT_SALES, na.rm = TRUE)) %>%
  ungroup(PREMIUM_CUSTOMER) %>%
  mutate(percent_tot_sales = total_sales/sum(total_sales, na.rm = TRUE)*100) %>%
  ggplot(aes(x = LIFESTAGE, y = percent_tot_sales, fill = PREMIUM_CUSTOMER)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = paste0(round(percent_tot_sales,2),"%")),
            position = position_stack(vjust = 0.5), size = 3, color = "black") +
  labs(
    title = "Total Chip Sales Based on Each Lifestage Category and Premium Customer",
    x = "Lifestage Category",
    y = "Total Sales (%)",
    fill = "Premium Customer Category"
  ) +
  scale_y_continuous(labels = function(x) format(x, big.mark = ",", scientific = FALSE)) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

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



```
Chip %>%
  group_by(LIFESTAGE,PREMIUM_CUSTOMER) %>%
  summarise(total_sales = sum(TOT_SALES, na.rm = TRUE)) %>%
  ungroup(PREMIUM_CUSTOMER, LIFESTAGE) %>%
  mutate(percent_tot_sales = total_sales/sum(total_sales, na.rm = TRUE)*100) %>%
  ggplot(aes(x = reorder(LIFESTAGE, - percent_tot_sales), y = percent_tot_sales, fill = PREMIU
  geom_bar(stat = "identity") +
  geom_text(aes(label = paste0(round(percent_tot_sales,2),"%")),
            position = position_stack(vjust = 0.5), size = 3, color = "black") +
  labs(
    title = "Total Chip Sales Based on Each Lifestage Category and Premium Customer",
    x = "Lifestage Category",
    y = "Total Sales (%)",
    fill = "Premium Customer Category"
  ) +
  scale_y_continuous(labels = function(x) format(x, big.mark = ",", scientific = FALSE)) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

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



Q5) When did customers buy more than 2 products per transaction?

```r
Chip %>%
  filter(PROD_QTY > 2) %>%
  arrange(DATE, PROD_QTY) %>%
  group_by(DATE, PROD_QTY) %>%
  summarise(Total_Customer = n())
```

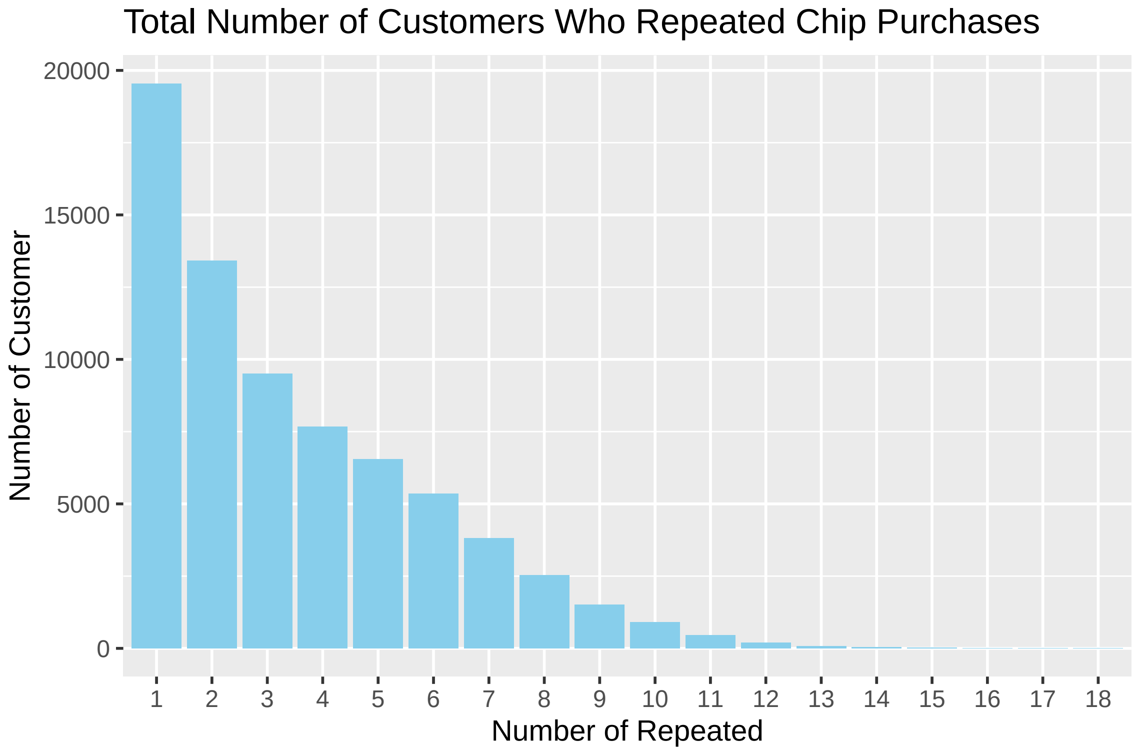
```
## `summarise()` has grouped output by 'DATE'. You can override using the
## `.groups` argument.

## # A tibble: 50 x 3
## # Groups:   DATE [22]
##    DATE       PROD_QTY Total_Customer
##    <date>        <dbl>          <int>
##  1 2018-07-08        3              1
##  2 2018-07-19        3              1
##  3 2018-08-14        3             38
##  4 2018-08-14        4             27
##  5 2018-08-14        5             21
##  6 2018-08-15        3             25
##  7 2018-08-15        4             33
##  8 2018-08-15        5             37
##  9 2018-08-16        3             25
## 10 2018-08-16        4             32
## # i 40 more rows
```

Between 14-08-2018 and 20-08-2018, as well as 14-05-2019 and 20-05-2019, some customers purchased more than 2 chip products per transaction.

Q6) How many customers repurchased chip products?

```
Chip %>%
  group_by(LYLTY_CARD_NBR) %>%
  summarise(cus_count = n_distinct(DATE)) %>%
  ungroup() %>%
  group_by(factor = as.factor(cus_count)) %>%
  summarise(count = n()) %>%
  ggplot(aes(x = factor, y = count)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(
    title = "Total Number of Customers Who Repeated Chip Purchases",
    x = "Number of Repeated",
    y = "Number of Customer"
  )
```



Q7)

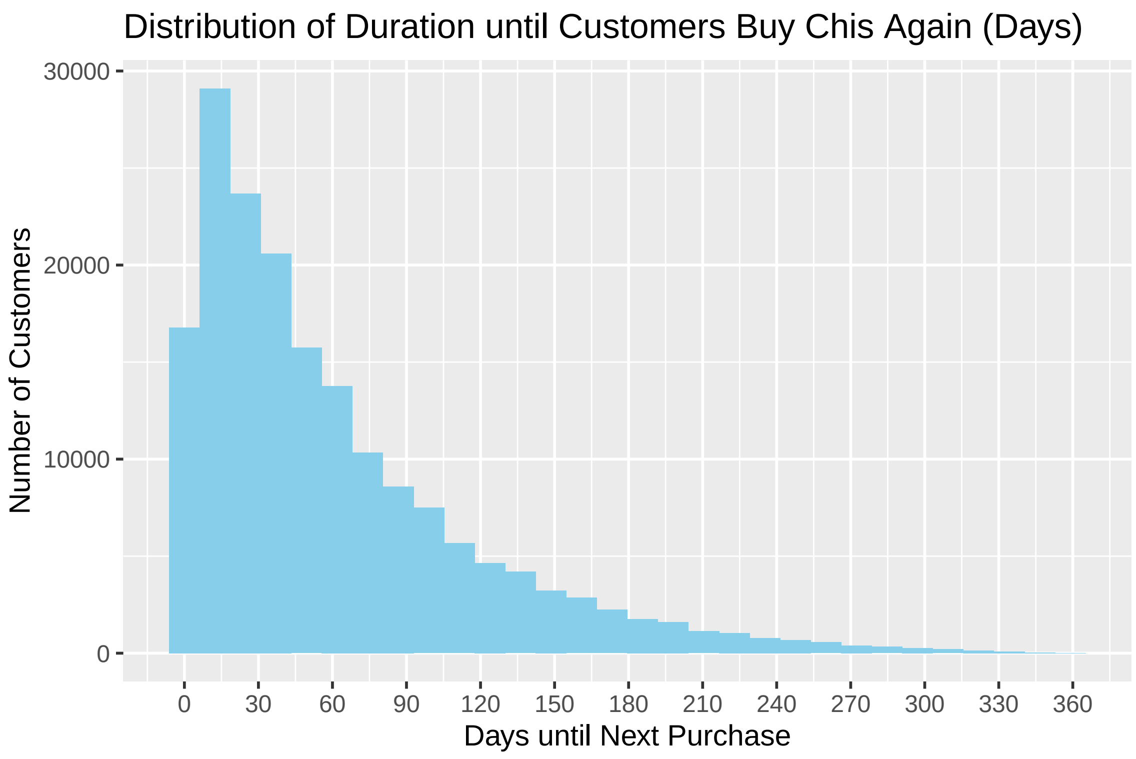
How long until the customer buy chip product again?

```
Chip %>%
  dplyr::select(LYLTY_CARD_NBR, DATE) %>%
  distinct(LYLTY_CARD_NBR, DATE) %>%
  arrange(LYLTY_CARD_NBR,DATE) %>%
  group_by(LYLTY_CARD_NBR) %>%
  mutate(Next_Buy = as.numeric(lead(DATE)- DATE)) %>%
  filter(Next_Buy != "NA days") %>%
  summary()
```

```
##  LYLTY_CARD_NBR        DATE               Next_Buy
##  Min.   :   1003   Min.   :2018-07-01   Min.   :  1.00
##  1st Qu.:  70227   1st Qu.:2018-09-10   1st Qu.: 18.00
##  Median : 130109   Median :2018-11-23   Median : 43.00
##  Mean   : 135247   Mean   :2018-11-29   Mean   : 60.54
##  3rd Qu.: 202344   3rd Qu.:2019-02-13   3rd Qu.: 85.00
##  Max.   :2370581   Max.   :2019-06-29   Max.   :360.00
```

```
Chip %>%
  dplyr::select(LYLTY_CARD_NBR, DATE) %>%
  distinct(LYLTY_CARD_NBR, DATE) %>%
  arrange(LYLTY_CARD_NBR,DATE) %>%
  group_by(LYLTY_CARD_NBR) %>%
  mutate(Next_Buy = as.numeric(lead(DATE)- DATE)) %>%
  filter(Next_Buy != "NA days") %>%
  ggplot(aes(x = Next_Buy)) +
  geom_histogram(fill = "skyblue") +
  labs(
    title = "Distribution of Duration until Customers Buy Chis Again (Days)",
    x = "Days until Next Purchase",
    y = "Number of Customers"
  ) +
  scale_x_continuous(breaks = seq(0, max(360, na.rm = TRUE), by = 30))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



This analysis shows how many days pass before customers repurchase chip products. The summary gives the distribution details, while the histogram helps visualize the frequency of repurchase intervals.

Q8) The difference in total sales between family groups and single/couple groups

```
t1 <- Chip %>%
  filter(LIFESTAGE %in% c("NEW FAMILIES","OLDER FAMILIES","YOUNG FAMILIES")) %>%
  pull(TOT_SALES)
t2 <- Chip %>%
  filter(LIFESTAGE %in% c("OLDER SINGLES/COUPLES","MIDAGE SINGLES/COUPLES","YOUNG SINGLES/COUP
  pull(TOT_SALES)
var.test(t1, t2)
```

```
##
##  F test to compare two variances
##
## data:  t1 and t2
## F = 0.94467, num df = 93970, denom df = 110012, p-value < 2.2e-16
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.9331147 0.9563780
## sample estimates:
## ratio of variances
##          0.9446719
```

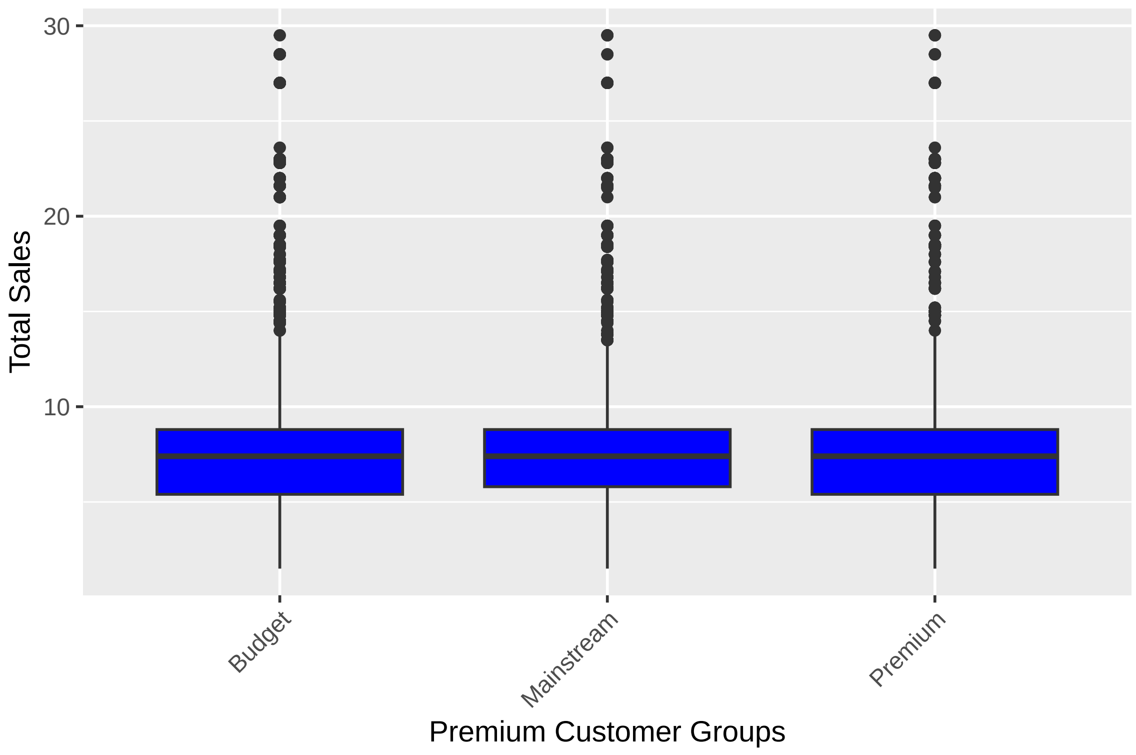
```r
t.test(t1, t2, var.equal = FALSE)
```

```
##
##  Welch Two Sample t-test
##
## data:  t1 and t2
## t = -4.6745, df = 200625, p-value = 2.949e-06
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.07325305 -0.02997158
## sample estimates:
## mean of x mean of y
##  7.219946  7.271558
```

The mean total sales for family groups and single/couple groups are 7.219946 and 7.271558 respectively. The t-test result shows a statistically significant difference between the two groups, suggesting that single and couple households tend to spend more on chip products compared to families.

Q9) Do sales differ significantly between 'PREMIUM_CUSTOMER' groups?

```r
Chip %>%
  ggplot(aes(x = PREMIUM_CUSTOMER, y = TOT_SALES)) +
  geom_boxplot(fill = "blue") +
  labs(
    x = "Premium Customer Groups",
    y = "Total Sales"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



```r
Chip %>%
  group_by(PREMIUM_CUSTOMER) %>%
  summarise(mean = mean(TOT_SALES, na.rm = TRUE), sd = sd(TOT_SALES, na.rm = TRUE))
```

```
## # A tibble: 3 x 3
##   PREMIUM_CUSTOMER  mean    sd
##   <chr>            <dbl> <dbl>
## 1 Budget            7.22  2.50
## 2 Mainstream        7.32  2.48
## 3 Premium           7.23  2.50
```

```r
leveneTest(TOT_SALES ~ PREMIUM_CUSTOMER, data = Chip)
```

```
## Warning in leveneTest.default(y = y, group = group, ...): group coerced to
## factor.
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##            Df F value    Pr(>F)
## group       2  13.839 9.778e-07 ***
##        251155
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
oneway.test(TOT_SALES ~ PREMIUM_CUSTOMER,
  data = Chip,
  var.equal = FALSE
)
```

```
##
##  One-way analysis of means (not assuming equal variances)
##
## data:  TOT_SALES and PREMIUM_CUSTOMER
## F = 45.159, num df = 2, denom df = 157781, p-value < 2.2e-16
```

```
gamesHowellTest(TOT_SALES ~ PREMIUM_CUSTOMER, data = Chip %>% mutate(PREMIUM_CUSTOMER = as.fac
```

```
##
##  Pairwise comparisons using Games-Howell test
```

```
## data: TOT_SALES by PREMIUM_CUSTOMER
```

```
##            Budget  Mainstream
## Mainstream 2.9e-14 -
## Premium    0.97    1.4e-13
```

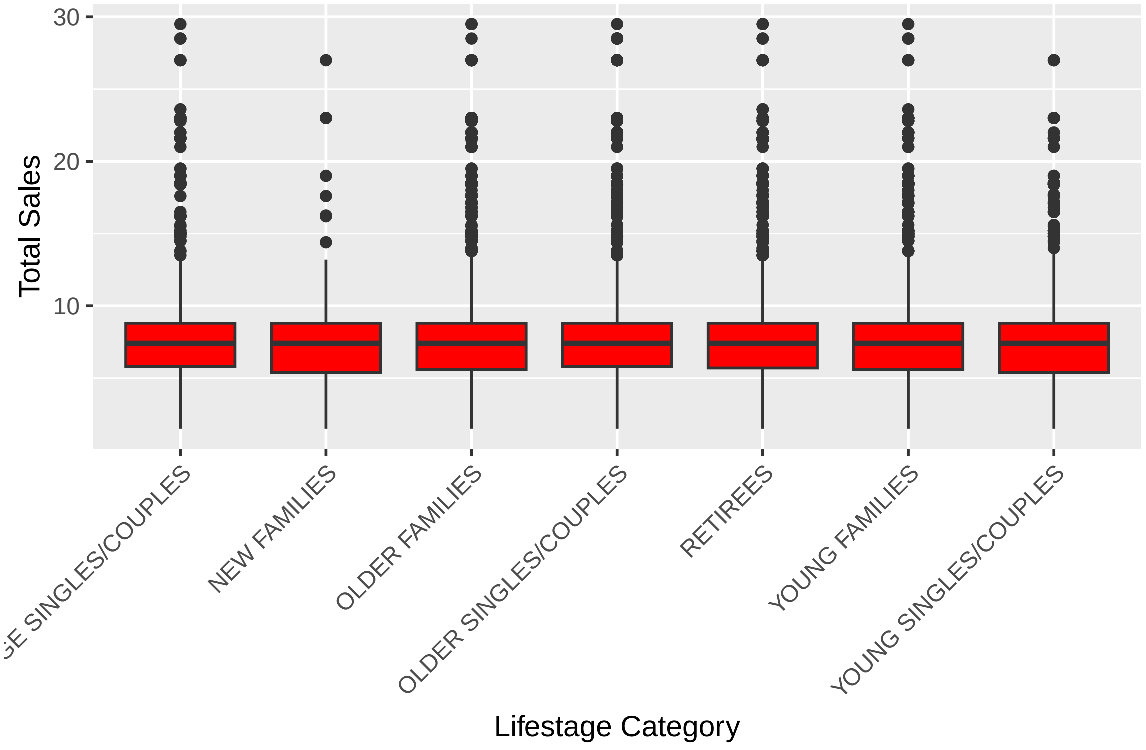
```
##
## P value adjustment method: none
```

```
## alternative hypothesis: two.sided
```

The mean total sales for budget, mainstream, and premium groups are 7.223785, 7.321973, and 7.226957 respectively. ANOVA indicates that there is a significant difference in total sales among at least one of the groups. The Games-Howell post hoc test reveals that the mainstream group differs significantly from both budget and premium groups, while the budget and premium groups are not significantly different from each other. This suggests that mainstream customers tend to spend more on chip products.

Q10) Do total sales significantly differ between lifestage groups?

```
Chip %>%
  ggplot(aes(x = LIFESTAGE, y = TOT_SALES)) +
  geom_boxplot(fill = "red") +
  labs(
    x = "Lifestage Category",
    y = "Total Sales"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



```
Chip %>%
  group_by(LIFESTAGE) %>%
  summarise(mean = mean(TOT_SALES, na.rm = TRUE), sd = sd(TOT_SALES, na.rm = TRUE))
```

```
## # A tibble: 7 x 3
##   LIFESTAGE               mean    sd
##   <chr>                  <dbl> <dbl>
## 1 MIDAGE SINGLES/COUPLES  7.32  2.50
## 2 NEW FAMILIES            7.24  2.53
## 3 OLDER FAMILIES          7.22  2.45
## 4 OLDER SINGLES/COUPLES   7.35  2.48
## 5 RETIREES                7.33  2.49
## 6 YOUNG FAMILIES          7.22  2.45
## 7 YOUNG SINGLES/COUPLES   7.12  2.60
```

```
leveneTest(TOT_SALES ~ LIFESTAGE, data = Chip)
```

```
## Warning in leveneTest.default(y = y, group = group, ...): group coerced to
## factor.
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##           Df F value    Pr(>F)
## group      6  36.603 < 2.2e-16 ***
##       251151
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
oneway.test(TOT_SALES ~ LIFESTAGE,
  data = Chip,
  var.equal = FALSE
)
```

```
##
##  One-way analysis of means (not assuming equal variances)
##
## data:  TOT_SALES and LIFESTAGE
## F = 39.476, num df = 6, denom df = 62089, p-value < 2.2e-16
```

```
gamesHowellTest(TOT_SALES ~ LIFESTAGE, data = Chip %>% mutate(LIFESTAGE= as.factor(LIFESTAGE))
```

```
##
##  Pairwise comparisons using Games-Howell test

## data: TOT_SALES by LIFESTAGE

##                        MIDAGE SINGLES/COUPLES NEW FAMILIES OLDER FAMILIES
## NEW FAMILIES           0.240                  -            -
## OLDER FAMILIES         9.3e-06                0.998        -
## OLDER SINGLES/COUPLES  0.716                  0.013        8.1e-14
## RETIREES               1.000                  0.118        1.1e-09
## YOUNG FAMILIES         9.1e-06                0.996        1.000
## YOUNG SINGLES/COUPLES  7.6e-14                0.011        1.2e-06
##                        OLDER SINGLES/COUPLES RETIREES YOUNG FAMILIES
## NEW FAMILIES           -                     -        -
## OLDER FAMILIES         -                     -        -
## OLDER SINGLES/COUPLES  -                     -        -
## RETIREES               0.735                 -        -
## YOUNG FAMILIES         9.4e-14               1.7e-09  -
## YOUNG SINGLES/COUPLES  < 2e-16               1.1e-14  4.2e-06

##
## P value adjustment method: none

## alternative hypothesis: two.sided
```

Average Sales: The average total sales are fairly similar across all lifestage groups, ranging from 7.12 to 7.35.

Highest: Older Singles/Couples (7.35)

Lowest: Young Singles/Couples (7.12)

Statistical Differences (Games-Howell Test): Significant differences in total sales were mostly observed between:

Young Singles/Couples and almost every other group (very small p-values, e.g., <2e-16 vs Older Singles/Couples).

Older Families vs:

Older Singles/Couples (p = 8.1e-14)

Retirees (p = 1.1e-09)

Young Singles/Couples (p = 1.2e-06)

Midage Singles/Couples and:

Older Families (p = 9.3e-06)

Young Families (p = 9.1e-06)

Young Singles/Couples (p = 7.6e-14)

No Significant Differences were found among:

Midage Singles/Couples, Retirees, and Older Singles/Couples (p > 0.7)

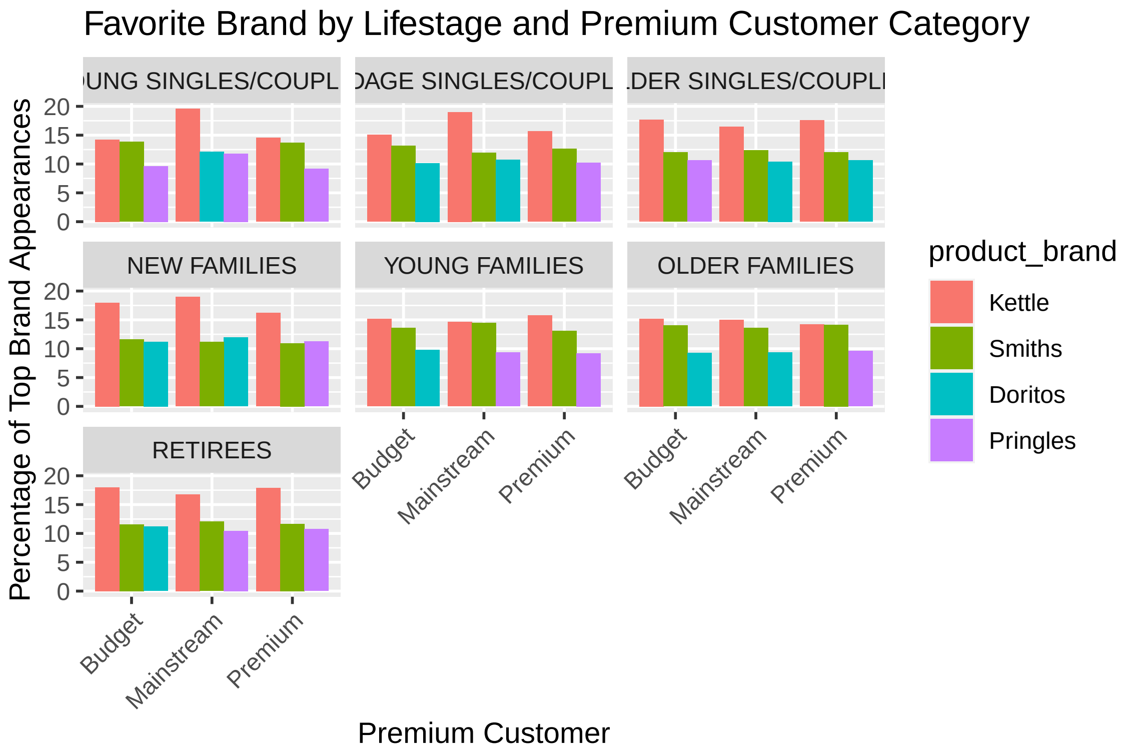
New Families, Older Families, and Young Families (p = 0.99)

While the average total sales values are close across lifestage segments, Young Singles/Couples consistently show significantly lower sales compared to most other groups. On the other hand, Older and Retired groups tend to spend more, but not always at significantly higher levels.

Q11) Which brand is the most popular among different 'LIFESTAGE' and 'PREMIUM_CUSTOMER' groups?

```
Chip %>%
  group_by(LIFESTAGE,PREMIUM_CUSTOMER, product_brand) %>%
  summarise(total_count = sum(PROD_QTY)) %>%
  ungroup() %>%
  group_by(LIFESTAGE,PREMIUM_CUSTOMER) %>%
  mutate(total_percent = total_count/sum(total_count)*100) %>%
  slice_max(order_by = total_percent, n = 3) %>%
  ungroup() %>%
  mutate(
    product_brand = reorder(product_brand, -total_percent),
    LIFESTAGE= factor(LIFESTAGE,
           levels = c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES","OLDER SINGLES/COUPLES"
    ) %>%
  ggplot(aes(x = PREMIUM_CUSTOMER, y = total_percent, fill = product_brand)) +
  geom_bar(position = "dodge", stat = "identity") +
  facet_wrap(~ LIFESTAGE) +
  labs(title = "Favorite Brand by Lifestage and Premium Customer Category",
       y = "Percentage of Top Brand Appearances",
       x = "Premium Customer") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

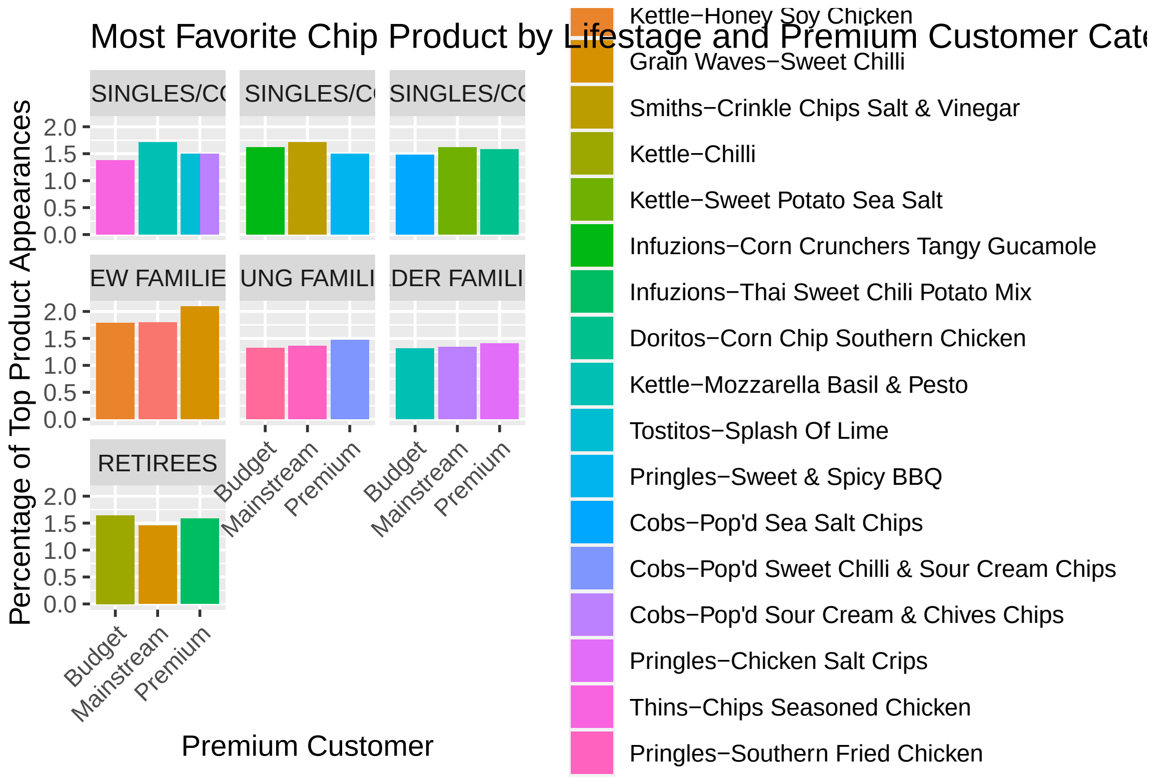
```
## `summarise()` has grouped output by 'LIFESTAGE', 'PREMIUM_CUSTOMER'. You can
## override using the `.groups` argument.
```



Q12) What is the most popular chip product among 'LIFESTAGE' and 'PREMIUM_CUSTOMER' groups?

```r
Chip %>%
  mutate(product_name = paste(product_brand,product_name, sep = "-")) %>%
  group_by(LIFESTAGE,PREMIUM_CUSTOMER, product_name) %>%
  summarise(total_count = sum(PROD_QTY)) %>%
  ungroup() %>%
  group_by(LIFESTAGE,PREMIUM_CUSTOMER) %>%
  mutate(total_percent = total_count/sum(total_count)*100) %>%
  slice_max(order_by = total_percent, n = 1) %>%
  ungroup() %>%
  mutate(
    product_name = reorder(product_name, -total_percent),
    LIFESTAGE= factor(LIFESTAGE,
           levels = c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES","OLDER SINGLES/COUPLES"
    ) %>%
  ggplot(aes(x = PREMIUM_CUSTOMER, y = total_percent, fill = product_name)) +
  geom_bar(position = "dodge", stat = "identity") +
  facet_wrap(~ LIFESTAGE) +
  labs(title = "Most Favorite Chip Product by Lifestage and Premium Customer Category",
       y = "Percentage of Top Product Appearances",
       x = "Premium Customer",
       fill = "Product Name") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

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

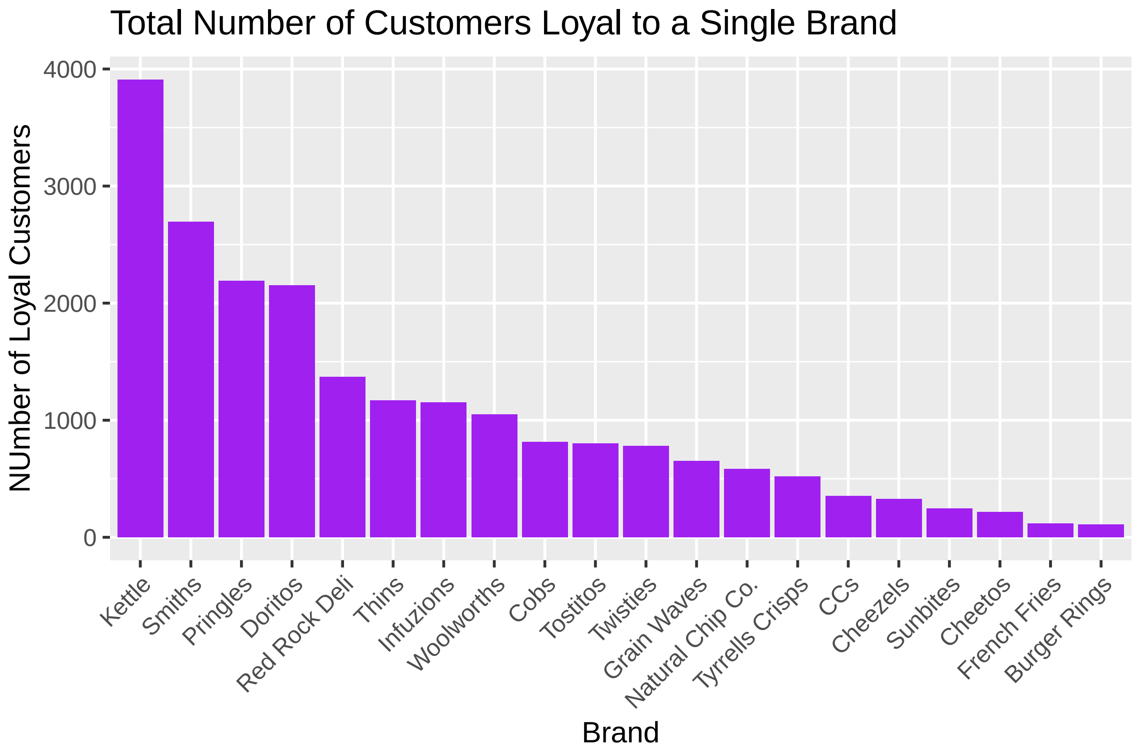


This chart highlights the most preferred chip product in each combination of ‘LIFESTAGE’ and ‘PREMIUM_CUSTOMER’ category. The product name is a combination of the brand and item name, making it easy to see specific preferences. Each bar shows the product with the highest share of purchases within that segment.

7 Assosiacion Rule Analysis

a) Association Rule Analysis: Brand Loyalty and Relationships

```
Chip %>%
  group_by(LYLTY_CARD_NBR) %>%
  mutate(tot_brand = n_distinct(product_brand)) %>%
  filter(tot_brand == 1) %>%
  ungroup() %>%
  group_by(product_brand) %>%
  summarise(loyal = n_distinct(LYLTY_CARD_NBR)) %>%
  ggplot(aes(x = reorder(product_brand, - loyal ), y = loyal)) +
  geom_bar(stat = "identity", fill = "purple") +
  labs(
    title = "Total Number of Customers Loyal to a Single Brand",
    x = "Brand",
    y = "NUmber of Loyal Customers"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



```
Chip %>%
  group_by(LYLTY_CARD_NBR) %>%
  summarise(tot_uniq_brand = n_distinct(product_brand)) %>%
  ungroup() %>%
  summarise(percent_customer = n_distinct(LYLTY_CARD_NBR[tot_uniq_brand >1])/n_distinct(LYLTY_C
```

```
## # A tibble: 1 x 1
##   percent_customer
##              <dbl>
## 1             70.4
```

```
Chip %>%
  group_by(LYLTY_CARD_NBR, LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarise(tot_uniq_brand = n_distinct(product_brand)) %>%
  ungroup() %>%
  group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarise(percent_customer = n_distinct(LYLTY_CARD_NBR[tot_uniq_brand >1])/n_distinct(LYLTY_C
```

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

```
## # A tibble: 21 x 3
## # Groups:   LIFESTAGE [7]
##    LIFESTAGE              PREMIUM_CUSTOMER percent_customer
##    <chr>                  <chr>                       <dbl>
##  1 MIDAGE SINGLES/COUPLES Budget                       68.8
##  2 MIDAGE SINGLES/COUPLES Mainstream                   72.8
##  3 MIDAGE SINGLES/COUPLES Premium                      67.8
##  4 NEW FAMILIES           Budget                       62.3
##  5 NEW FAMILIES           Mainstream                   65.0
##  6 NEW FAMILIES           Premium                      68.7
##  7 OLDER FAMILIES         Budget                       77.4
##  8 OLDER FAMILIES         Mainstream                   77.8
##  9 OLDER FAMILIES         Premium                      77.2
## 10 OLDER SINGLES/COUPLES  Budget                       75.5
## # i 11 more rows
```

```
basket_data <- Chip %>%
  group_by(LYLTY_CARD_NBR) %>%
  summarise(items = list(unique(product_brand)))
```

```
trans_list <- as(basket_data$items, "transactions")
```

```
rules <- apriori(trans_list, parameter = list(supp = 0.01, conf = 0.1), appearance = list(lhs =
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.1    0.1    1 none FALSE            TRUE       5    0.01      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 716
```

```
##
## set item appearances ...[1 item(s)] done [0.00s].
## set transactions ...[20 item(s), 71624 transaction(s)] done [0.03s].
## sorting and recoding items ... [20 item(s)] done [0.01s].
## creating transaction tree ... done [0.04s].
## checking subsets of size 1 2 done [0.01s].
## writing ... [22 rule(s)] done [0.00s].
## creating S4 object  ... done [0.01s].
```

```
sorted_rules <- sort(rules, by = "lift", decreasing = TRUE)
inspect(sorted_rules[1:10])
```

```
##      lhs         rhs           support    confidence coverage  lift     count
## [1]  {Kettle} => {Cobs}        0.06051044 0.1436241  0.4213113 1.147583 4334
## [2]  {Kettle} => {Twisties}    0.05847202 0.1387858  0.4213113 1.138778 4188
## [3]  {Kettle} => {Tostitos}    0.05805317 0.1377916  0.4213113 1.126620 4158
## [4]  {Kettle} => {Thins}       0.08304479 0.1971103  0.4213113 1.121263 5948
## [5]  {Kettle} => {Doritos}     0.13577851 0.3222760  0.4213113 1.114837 9725
## [6]  {Kettle} => {Infuzions}   0.08261197 0.1960830  0.4213113 1.111096 5917
## [7]  {Kettle} => {Pringles}    0.13481515 0.3199894  0.4213113 1.109499 9656
## [8]  {Kettle} => {Grain Waves} 0.04705127 0.1116782  0.4213113 1.104507 3370
## [9]  {Kettle} => {Smiths}      0.13790070 0.3273131  0.4213113 1.012896 9877
## [10] {}       => {Grain Waves} 0.10111136 0.1011114  1.0000000 1.000000 7242
```

```
rules <- apriori(trans_list, parameter = list(supp = 0.01, conf = 0.1), appearance = list(lhs =
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.1    0.1    1 none FALSE            TRUE       5    0.01      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 716
##
## set item appearances ...[1 item(s)] done [0.00s].
## set transactions ...[20 item(s), 71624 transaction(s)] done [0.03s].
## sorting and recoding items ... [20 item(s)] done [0.00s].
## creating transaction tree ... done [0.05s].
## checking subsets of size 1 2 done [0.00s].
## writing ... [23 rule(s)] done [0.00s].
## creating S4 object  ... done [0.01s].
```

```r
sorted_rules <- sort(rules, by = "lift", decreasing = TRUE)
inspect(sorted_rules[1:10])
```

```
##      lhs         rhs                  support    confidence coverage  lift
## [1]  {Smiths} => {Natural Chip Co.} 0.04883838 0.1511342  0.3231459 1.621212
## [2]  {Smiths} => {Red Rock Deli}    0.09908690 0.3066321  0.3231459 1.600745
## [3]  {Smiths} => {Woolworths}       0.07866078 0.2434219  0.3231459 1.586284
## [4]  {Smiths} => {Grain Waves}      0.03733385 0.1155325  0.3231459 1.142626
## [5]  {Smiths} => {Thins}            0.06211605 0.1922229  0.3231459 1.093462
## [6]  {Smiths} => {Infuzions}        0.06185078 0.1914020  0.3231459 1.084571
## [7]  {Smiths} => {Tostitos}         0.04156428 0.1286239  0.3231459 1.051662
## [8]  {Smiths} => {Pringles}         0.09626661 0.2979045  0.3231459 1.032924
## [9]  {Smiths} => {Doritos}          0.09640623 0.2983366  0.3231459 1.032024
## [10] {Smiths} => {Cobs}             0.04134089 0.1279326  0.3231459 1.022205
##      count
## [1]  3498
## [2]  7097
## [3]  5634
## [4]  2674
## [5]  4449
## [6]  4430
## [7]  2977
## [8]  6895
## [9]  6905
## [10] 2961
```

```r
rules <- apriori(trans_list, parameter = list(supp = 0.01, conf = 0.1), appearance = list(lhs =
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.1    0.1    1 none FALSE            TRUE       5    0.01      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 716
##
## set item appearances ...[1 item(s)] done [0.00s].
## set transactions ...[20 item(s), 71624 transaction(s)] done [0.04s].
## sorting and recoding items ... [20 item(s)] done [0.00s].
## creating transaction tree ... done [0.05s].
## checking subsets of size 1 2 done [0.00s].
## writing ... [22 rule(s)] done [0.00s].
## creating S4 object  ... done [0.01s].
```

```
sorted_rules <- sort(rules, by = "lift", decreasing = TRUE)
inspect(sorted_rules[1:10])
```

```
##      lhs          rhs              support    confidence coverage  lift
## [1]  {Doritos} => {Twisties}       0.04088015 0.1414151  0.2890791 1.160352
## [2]  {Doritos} => {Infuzions}      0.05917011 0.2046849  0.2890791 1.159838
## [3]  {Doritos} => {Cobs}           0.04195521 0.1451340  0.2890791 1.159647
## [4]  {Doritos} => {Thins}          0.05820675 0.2013523  0.2890791 1.145394
## [5]  {Doritos} => {Tostitos}       0.04019602 0.1390485  0.2890791 1.136896
## [6]  {Doritos} => {Pringles}       0.09408857 0.3254769  0.2890791 1.128526
## [7]  {Doritos} => {Kettle}         0.13577851 0.4696933  0.2890791 1.114837
## [8]  {Doritos} => {Grain Waves}    0.03183291 0.1101183  0.2890791 1.089080
## [9]  {Doritos} => {Smiths}         0.09640623 0.3334943  0.2890791 1.032024
## [10] {Doritos} => {Red Rock Deli} 0.05576343 0.1929003  0.2890791 1.007018
##      count
## [1]  2928
## [2]  4238
## [3]  3005
## [4]  4169
## [5]  2879
## [6]  6739
## [7]  9725
## [8]  2280
## [9]  6905
## [10] 3994
```

```
rules <- apriori(trans_list, parameter = list(supp = 0.01, conf = 0.1), appearance = list(lhs
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.1    0.1    1 none FALSE            TRUE       5    0.01      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 716
##
```

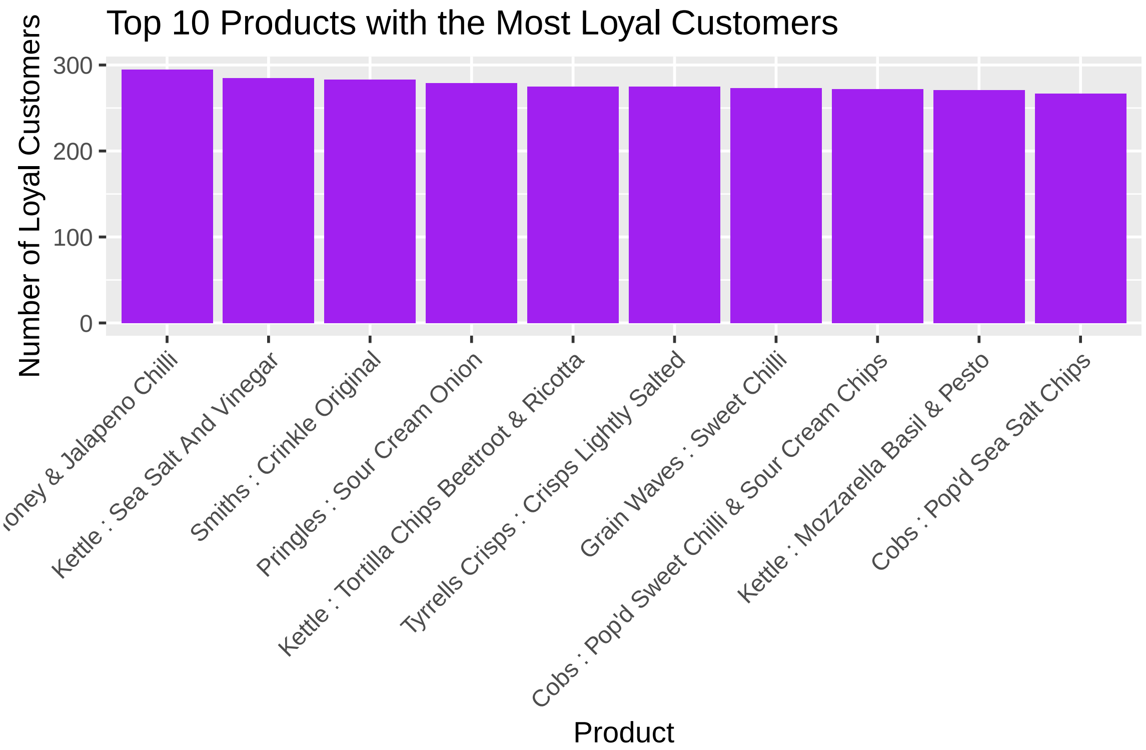
```
## set item appearances ...[1 item(s)] done [0.00s].
## set transactions ...[20 item(s), 71624 transaction(s)] done [0.07s].
## sorting and recoding items ... [20 item(s)] done [0.00s].
## creating transaction tree ... done [0.05s].
## checking subsets of size 1 2 done [0.00s].
## writing ... [23 rule(s)] done [0.00s].
## creating S4 object  ... done [0.01s].
```

```
sorted_rules <- sort(rules, by = "lift", decreasing = TRUE)
inspect(sorted_rules[1:10])
```

```
##      lhs           rhs                support    confidence coverage  lift
## [1]  {Pringles} => {Tyrrells Crisps} 0.02891489 0.1002566  0.2884089 1.175442
## [2]  {Pringles} => {Cobs}            0.04174578 0.1447451  0.2884089 1.156540
## [3]  {Pringles} => {Twisties}        0.04029376 0.1397105  0.2884089 1.146366
## [4]  {Pringles} => {Tostitos}        0.04034960 0.1399041  0.2884089 1.143892
## [5]  {Pringles} => {Infuzions}       0.05816486 0.2016750  0.2884089 1.142782
## [6]  {Pringles} => {Doritos}         0.09408857 0.3262332  0.2884089 1.128526
## [7]  {Pringles} => {Thins}           0.05715961 0.1981895  0.2884089 1.127402
## [8]  {Pringles} => {Grain Waves}     0.03283815 0.1138597  0.2884089 1.126082
## [9]  {Pringles} => {Kettle}          0.13481515 0.4674444  0.2884089 1.109499
## [10] {Pringles} => {Smiths}          0.09626661 0.3337852  0.2884089 1.032924
##      count
## [1]  2071
## [2]  2990
## [3]  2886
## [4]  2890
## [5]  4166
## [6]  6739
## [7]  4094
## [8]  2352
## [9]  9656
## [10] 6895
```

b) Association Rule Analysis: Chip Products Loyalty and Relationships

```
Chip %>%
  group_by(LYLTY_CARD_NBR) %>%
  mutate(product_name = paste(product_brand, product_name, sep = " : "),tot_prod = n_distinct(
  filter(tot_prod == 1) %>%
  ungroup() %>%
  group_by(product_name) %>%
  summarise(loyal = n_distinct(LYLTY_CARD_NBR)) %>%
  ungroup() %>%
  slice_max(order_by = loyal, n = 10) %>%
  ggplot(aes(x = reorder(product_name, - loyal ), y = loyal)) +
  geom_bar(stat = "identity", fill = "purple") +
  labs(
    title = "Top 10 Products with the Most Loyal Customers",
    x = "Product",
    y = "Number of Loyal Customers"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



```
Chip %>%
  group_by(LYLTY_CARD_NBR) %>%
  summarise(tot_uniq_prod = n_distinct(product_name)) %>%
  ungroup() %>%
  summarise(percent_customer = n_distinct(LYLTY_CARD_NBR[tot_uniq_prod >1])/n_distinct(LYLTY_C
```

```
## # A tibble: 1 x 1
##   percent_customer
##              <dbl>
## 1             72.5
```

```
Chip %>%
  group_by(LYLTY_CARD_NBR, LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarise(tot_uniq_prod = n_distinct(product_name)) %>%
  ungroup() %>%
  group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
  summarise(percent_customer = n_distinct(LYLTY_CARD_NBR[tot_uniq_prod >1])/n_distinct(LYLTY_CA
```

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

```
## # A tibble: 21 x 3
## # Groups:   LIFESTAGE [7]
##    LIFESTAGE              PREMIUM_CUSTOMER percent_customer
##    <chr>                  <chr>                       <dbl>
##  1 MIDAGE SINGLES/COUPLES Budget                       70.6
##  2 MIDAGE SINGLES/COUPLES Mainstream                   74.7
##  3 MIDAGE SINGLES/COUPLES Premium                      70.0
##  4 NEW FAMILIES           Budget                       64.6
##  5 NEW FAMILIES           Mainstream                   67.4
##  6 NEW FAMILIES           Premium                      71.1
##  7 OLDER FAMILIES         Budget                       79.0
##  8 OLDER FAMILIES         Mainstream                   79.5
##  9 OLDER FAMILIES         Premium                      79.2
## 10 OLDER SINGLES/COUPLES  Budget                       77.2
## # i 11 more rows
```

```
basket_data <- Chip %>%
  group_by(LYLTY_CARD_NBR) %>%
  mutate(product_name = paste(product_brand, product_name, sep = " : ")) %>%
  summarise(items = list(unique(product_name)))
```

```
trans_list <- as(basket_data$items, "transactions")
```

```
rules <- apriori(trans_list, parameter = list(supp = 0.001, conf = 0.05), appearance = list(lhs
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.05    0.1    1 none FALSE            TRUE       5   0.001      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
```

```
## Absolute minimum support count: 71
##
## set item appearances ...[1 item(s)] done [0.00s].
## set transactions ...[108 item(s), 71624 transaction(s)] done [0.05s].
## sorting and recoding items ... [108 item(s)] done [0.02s].
## creating transaction tree ... done [0.07s].
## checking subsets of size 1 2 done [0.01s].
## writing ... [33 rule(s)] done [0.00s].
## creating S4 object  ... done [0.01s].
```

```r
sorted_rules <- sort(rules, by = "lift", decreasing = TRUE)
inspect(sorted_rules[1:10])
```

```
##      lhs                                  rhs
## [1]  {Kettle : Mozzarella Basil & Pesto} => {Kettle : Original}                            0.0
## [2]  {Kettle : Mozzarella Basil & Pesto} => {Tostitos : Smoked Chipotle}                   0.0
## [3]  {Kettle : Mozzarella Basil & Pesto} => {Kettle : Sensations Camembert & Fig}          0.0
## [4]  {Kettle : Mozzarella Basil & Pesto} => {Infuzions : Corn Crunchers Tangy Gucamole}    0.0
## [5]  {Kettle : Mozzarella Basil & Pesto} => {Kettle : Tortilla Chips Beetroot & Ricotta}   0.0
## [6]  {Kettle : Mozzarella Basil & Pesto} => {Doritos : Corn Chip Southern Chicken}         0.0
## [7]  {Kettle : Mozzarella Basil & Pesto} => {Pringles : Mystery Flavour}                   0.0
## [8]  {Kettle : Mozzarella Basil & Pesto} => {Tyrrells Crisps : Crisps Ched & Chives}       0.0
## [9]  {Kettle : Mozzarella Basil & Pesto} => {Twisties : Cheese}                            0.0
## [10] {Kettle : Mozzarella Basil & Pesto} => {Thins : Potato Chips Hot & Spicy}             0.0
```

```r
rules <- apriori(trans_list, parameter = list(supp = 0.001, conf = 0.05), appearance = list(lhs
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.05    0.1    1 none FALSE            TRUE       5   0.001      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 71
##
## set item appearances ...[1 item(s)] done [0.00s].
## set transactions ...[108 item(s), 71624 transaction(s)] done [0.04s].
## sorting and recoding items ... [108 item(s)] done [0.00s].
## creating transaction tree ... done [0.06s].
## checking subsets of size 1 2 done [0.01s].
## writing ... [36 rule(s)] done [0.00s].
## creating S4 object  ... done [0.01s].
```

```
sorted_rules <- sort(rules, by = "lift", decreasing = TRUE)
inspect(sorted_rules[1:10])
```

```
##      lhs                                                  rhs
## [1]  {Kettle : Tortilla Chips Honey & Jalapeno Chilli} => {Doritos : Cheese Supreme}
## [2]  {Kettle : Tortilla Chips Honey & Jalapeno Chilli} => {Cobs : Pop'd Sour Cream & Chives
## [3]  {Kettle : Tortilla Chips Honey & Jalapeno Chilli} => {Pringles : Southern Fried Chicke
## [4]  {Kettle : Tortilla Chips Honey & Jalapeno Chilli} => {Infuzions : Sour Cream & Herbs Ve
## [5]  {Kettle : Tortilla Chips Honey & Jalapeno Chilli} => {Smiths : Crinkle Chip Original Bi
## [6]  {Kettle : Tortilla Chips Honey & Jalapeno Chilli} => {Tostitos : Splash Of Lime}
## [7]  {Kettle : Tortilla Chips Honey & Jalapeno Chilli} => {Doritos : Corn Chips Cheese Supre
## [8]  {Kettle : Tortilla Chips Honey & Jalapeno Chilli} => {Thins : Potato Chips Hot & Spicy}
## [9]  {Kettle : Tortilla Chips Honey & Jalapeno Chilli} => {Doritos : Corn Chip Supreme}
## [10] {Kettle : Tortilla Chips Honey & Jalapeno Chilli} => {Pringles : Mystery Flavour}
```

```
rules <- apriori(trans_list, parameter = list(supp = 0.001, conf = 0.05), appearance = list(lh
```

```
## Apriori
## 
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.05    0.1    1 none FALSE            TRUE       5   0.001      1
##  maxlen target  ext
##      10  rules TRUE
## 
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
## 
## Absolute minimum support count: 71 
## 
## set item appearances ...[1 item(s)] done [0.00s].
## set transactions ...[108 item(s), 71624 transaction(s)] done [0.06s].
## sorting and recoding items ... [108 item(s)] done [0.00s].
## creating transaction tree ... done [0.05s].
## checking subsets of size 1 2 done [0.01s].
## writing ... [40 rule(s)] done [0.00s].
## creating S4 object  ... done [0.01s].
```

```
sorted_rules <- sort(rules, by = "lift", decreasing = TRUE)
inspect(sorted_rules[1:10])
```

```
##      lhs                              rhs                                           support
## [1]  {Cobs : Pop'd Sea Salt Chips} => {Kettle : Sensations BBQ & Maple}         0.002555009
## [2]  {Cobs : Pop'd Sea Salt Chips} => {Pringles : Salt Vinegar}                 0.002527086
## [3]  {Cobs : Pop'd Sea Salt Chips} => {Pringles : Mystery Flavour}              0.002513124
## [4]  {Cobs : Pop'd Sea Salt Chips} => {Tyrrells Crisps : Crisps Lightly Salted} 0.002513124
## [5]  {Cobs : Pop'd Sea Salt Chips} => {Twisties : Chicken}                      0.002499162
## [6]  {Cobs : Pop'd Sea Salt Chips} => {Kettle : Chilli}                         0.002401430
## [7]  {Cobs : Pop'd Sea Salt Chips} => {Kettle : Sea Salt And Vinegar}           0.002499162
## [8]  {Cobs : Pop'd Sea Salt Chips} => {Kettle : Honey Soy Chicken}              0.002485200
## [9]  {Cobs : Pop'd Sea Salt Chips} => {Cheezels : Cheese}                       0.002457277
## [10] {Cobs : Pop'd Sea Salt Chips} => {Doritos : Corn Chips Cheese Supreme}     0.002513124
```

```
rules <- apriori(trans_list, parameter = list(supp = 0.001, conf = 0.05), appearance = list(lhs
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.05    0.1    1 none FALSE            TRUE       5   0.001      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 71
##
## set item appearances ...[1 item(s)] done [0.00s].
## set transactions ...[108 item(s), 71624 transaction(s)] done [0.04s].
## sorting and recoding items ... [108 item(s)] done [0.01s].
## creating transaction tree ... done [0.07s].
## checking subsets of size 1 2 done [0.01s].
## writing ... [40 rule(s)] done [0.00s].
## creating S4 object  ... done [0.01s].
```

```
sorted_rules <- sort(rules, by = "lift", decreasing = TRUE)
inspect(sorted_rules[1:10])
```

```
##      lhs                                        rhs
## [1]  {Tyrrells Crisps : Crisps Ched & Chives} => {Tostitos : Lightly Salted}
## [2]  {Tyrrells Crisps : Crisps Ched & Chives} => {Thins : Potato Chips Hot & Spicy}
## [3]  {Tyrrells Crisps : Crisps Ched & Chives} => {Doritos : Corn Chips Nacho Cheese}
## [4]  {Tyrrells Crisps : Crisps Ched & Chives} => {Twisties : Cheese Burger}
## [5]  {Tyrrells Crisps : Crisps Ched & Chives} => {Infuzions : Thai Sweet Chili Potato Mix}
## [6]  {Tyrrells Crisps : Crisps Ched & Chives} => {Grain Waves : Sweet Chilli}
## [7]  {Tyrrells Crisps : Crisps Ched & Chives} => {Pringles : Sour Cream Onion}
## [8]  {Tyrrells Crisps : Crisps Ched & Chives} => {Cheezels : Cheese}                       0
## [9]  {Tyrrells Crisps : Crisps Ched & Chives} => {Infuzions : BBQ Rib Prawn Crackers}      0
## [10] {Tyrrells Crisps : Crisps Ched & Chives} => {Doritos : Corn Chip Mexican Jalapeno}    0
```

```
rules <- apriori(trans_list, parameter = list(supp = 0.001, conf = 0.05), appearance = list(lhs
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.05    0.1    1 none FALSE            TRUE       5   0.001      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 71
##
## set item appearances ...[1 item(s)] done [0.00s].
## set transactions ...[108 item(s), 71624 transaction(s)] done [0.06s].
## sorting and recoding items ... [108 item(s)] done [0.01s].
## creating transaction tree ... done [0.05s].
## checking subsets of size 1 2 done [0.01s].
## writing ... [37 rule(s)] done [0.00s].
## creating S4 object  ... done [0.01s].
```

```
sorted_rules <- sort(rules, by = "lift", decreasing = TRUE)
inspect(sorted_rules[1:10])
```

```
##      lhs                                 rhs                                       suppo
## [1]  {Kettle : Sweet Potato Sea Salt} => {Grain Waves : Sour Cream & Chives}       0.0025829
## [2]  {Kettle : Sweet Potato Sea Salt} => {Cobs : Pop'd Sour Cream & Chives Chips}  0.0024852
## [3]  {Kettle : Sweet Potato Sea Salt} => {Pringles : Sweet & Spicy BBQ}            0.0024572
## [4]  {Kettle : Sweet Potato Sea Salt} => {Tyrrells Crisps : Crisps Lightly Salted} 0.0024293
## [5]  {Kettle : Sweet Potato Sea Salt} => {Thins : Potato Chips Hot & Spicy}        0.0024852
## [6]  {Kettle : Sweet Potato Sea Salt} => {Kettle : Honey Soy Chicken}              0.0024153
## [7]  {Kettle : Sweet Potato Sea Salt} => {Thins : Chips Light & Tangy}             0.0024293
## [8]  {Kettle : Sweet Potato Sea Salt} => {Doritos : Corn Chips Original}           0.0023874
## [9]  {Kettle : Sweet Potato Sea Salt} => {Pringles : Original Crisps}              0.0024153
## [10] {Kettle : Sweet Potato Sea Salt} => {Pringles : Sour Cream Onion}             0.0024014
```

8 Export Chip Data to Excel

```
# Install package if not already installed
# install.packages("openxlsx")

# Load the package
#library(openxlsx)
# Set working directory
#setwd("C:/Users/User/Documents/Project Forage/Quantium")
# Export to Excel
#write.xlsx(Chip, "chip_data.xlsx")
```