Quantium_analysis

Sayid_Mufaqih

2024-01-17

Introduction

Quantium's retail analytics team have been approached by the client, the Category Manager for Chips, who wants to better understand the types of customers who purchase Chips and their purchasing behaviour within the region. The insights from the analysis will feed into the supermarket's strategic plan for the chip category in the next half year.

Business Task

Understand customer segmentation and the current purchasing trends and behaviours.

Data Analysis Task

Examine transaction data – look for inconsistencies, missing data across the data set, outliers, correctly identified category items, numeric data across all tables. If you determine any anomalies make the necessary changes in the dataset and save it. Having clean data will help when it comes to your analysis.

Examine customer data – check for similar issues in the customer data, look for nulls and when you are happy merge the transaction and customer data together so it's ready for the analysis ensuring you save your files along the way.

Data analysis and customer segments – in your analysis make sure you define the metrics – look at total sales, drivers of sales, where the highest sales are coming from etc. Explore the data, create charts and graphs as well as noting any interesting trends and/or insights you find. These will all form part of our report to Julia.

Deep dive into customer segments – define your recommendation from your insights, determine which segments we should be targeting, if packet sizes are relative and form an overall conclusion based on your analysis.

Loading Packages

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
```

```
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(lubridate)
library(dplyr)
library(ggplot2)
library(tidyr)
library(skimr)
library(here)
```

```
## here() starts at D:/VIRTUAL_INTERNSHIP/Quantium
```

```
library(janitor)
```

```
##
## Attaching package: 'janitor'
##
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

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

```
##
## Attaching package: 'data.table'
##
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
##
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
##
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
library(stringr)
```

Importing Dataset

```
##   LYLTY_CARD_NBR              LIFESTAGE PREMIUM_CUSTOMER
## 1           1000  YOUNG SINGLES/COUPLES          Premium
## 2           1002  YOUNG SINGLES/COUPLES       Mainstream
```

```
## 3           1003         YOUNG FAMILIES          Budget
## 4           1004  OLDER SINGLES/COUPLES       Mainstream
## 5           1005 MIDAGE SINGLES/COUPLES       Mainstream
## 6           1007  YOUNG SINGLES/COUPLES          Budget
```

```
## # A tibble: 6 x 8
##    DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR PROD_NAME    PROD_QTY TOT_SALES
##   <dbl>     <dbl>          <dbl>  <dbl>    <dbl> <chr>           <dbl>     <dbl>
## 1 43390         1           1000      1        5 Natural Chi~        2       6
## 2 43599         1           1307    348       66 CCs Nacho C~        3       6.3
## 3 43605         1           1343    383       61 Smiths Crin~        2       2.9
## 4 43329         2           2373    974       69 Smiths Chip~        5      15
## 5 43330         2           2426   1038      108 Kettle Tort~        3      13.8
## 6 43604         4           4074   2982       57 Old El Paso~        1       5.1
```

Examining Transaction Data

```r
transaction_df <- clean_names(transaction)
skim_without_charts(transaction_df)
```

Table 1: Data summary

Name	transaction_df
Number of rows	264836
Number of columns	8
Column type frequency:	
character	1
numeric	7
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
prod_name	0	1	17	40	0	114	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
date	0	1	43464.04	105.39	43282.0	43373.0	43464.0	43555.0	43646
store_nbr	0	1	135.08	76.78	1.0	70.0	130.0	203.0	272
lylty_card_nbr	0	1	135549.48	80579.98	1000.0	70021.0	130357.5	203094.2	2373711
txn_id	0	1	135158.31	78133.03	1.0	67601.5	135137.5	202701.2	2415841
prod_nbr	0	1	56.58	32.83	1.0	28.0	56.0	85.0	114
prod_qty	0	1	1.91	0.64	1.0	2.0	2.0	2.0	200
tot_sales	0	1	7.30	3.08	1.5	5.4	7.4	9.2	650

```r
transaction_df$date <- as.Date(transaction_df$date, origin = "1899-12-30")
head(transaction_df)
```

Convert date column to a date format

```
## # A tibble: 6 x 8
##   date       store_nbr lylty_card_nbr txn_id prod_nbr prod_name          prod_qty
##   <date>         <dbl>          <dbl>  <dbl>    <dbl> <chr>                 <dbl>
## 1 2018-10-17         1           1000      1        5 Natural Chip    ~         2
## 2 2019-05-14         1           1307    348       66 CCs Nacho Cheese~         3
## 3 2019-05-20         1           1343    383       61 Smiths Crinkle C~         2
## 4 2018-08-17         2           2373    974       69 Smiths Chip Thin~         5
## 5 2018-08-18         2           2426   1038      108 Kettle Tortilla ~         3
## 6 2019-05-19         4           4074   2982       57 Old El Paso Sals~         1
## # i 1 more variable: tot_sales <dbl>
```

```r
product_words <- data.table(unlist(strsplit(unique(transaction_df$prod_name), " ")))
print(product_words)
```

Examine the words in prode_name to see if there are any incorrect entries such as products that are not chips

```
##           V1
##   1: Natural
##   2:    Chip
##   3:
##   4:
##   5:
##  ---
## 819: Doritos
## 820:   Salsa
## 821:    Mild
## 822:
## 823:    300g
```

```r
##### Remove characters

words_data <- str_replace_all(product_words, "[^[:alnum:]]", " ")
```

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

```
## Warning in stri_replace_all_regex(string, pattern,
## fix_replacement(replacement), : argument is not an atomic vector; coercing
```

```r
words_data
```

```
## [1] "c  Natural    Chip                                Compny    SeaSalt175g    CCs    Nacho    Chee
```

```r
#### Remove digits
words_clean <- gsub('[[:digit:]]+', '',words_data)
words_clean
```

```
## [1] "c  Natural    Chip                                Compny    SeaSaltg    CCs    Nacho    Cheese
```

```r
#### Make a table
words_product <- data.table(unlist(strsplit(unique(words_clean)," ")))
setnames(words_product, "words")
words_product
```

```
##         words
##    1:       c
##    2:
##    3: Natural
##    4:
##    5:
##   ---
## 3344:
## 3345:
## 3346:
## 3347:       g
## 3348:
```

```r


#### Remove blank, count, and sort

words_product %>%
mutate(words = na_if(words, "")) %>%
    filter(!is.na(words)) %>%
    group_by(words) %>%
    count(words, sort= TRUE)
```

Look at the most common words by counting the number of times a word appears and sorting them by this frequency in order of highest to lowest frequency

```
## # A tibble: 205 x 2
## # Groups:   words [205]
##    words        n
##    <chr>    <int>
##  1 g          105
##  2 Chips       21
##  3 Smiths      16
##  4 Crinkle     14
##  5 Cut         14
```

```
##  6 Kettle      13
##  7 Cheese      12
##  8 Salt        12
##  9 Original    10
## 10 Chip         9
## # i 195 more rows
```

There are salsa products in the dataset

```r
#### create salsa phrase
remove_salsa <- c('salsa', 'Salsa','SALSA')
#### remove rows than contain salsa on transaction dataset
clean_transaction <- transaction_df[ !grepl(paste(remove_salsa, collapse="|"), transaction_df$prod_name
```

Remove SALSA product

```r
summary(clean_transaction)
```

Summarise the data to check for nulls and possible outliers

```
##       date              store_nbr     lylty_card_nbr        txn_id       
##  Min.   :2018-07-01   Min.   :  1.0   Min.   :   1000   Min.   :      1  
##  1st Qu.:2018-09-30   1st Qu.: 70.0   1st Qu.:  70015   1st Qu.:  67569  
##  Median :2018-12-30   Median :130.0   Median : 130367   Median : 135183  
##  Mean   :2018-12-30   Mean   :135.1   Mean   : 135531   Mean   : 135131  
##  3rd Qu.:2019-03-31   3rd Qu.:203.0   3rd Qu.: 203084   3rd Qu.: 202654  
##  Max.   :2019-06-30   Max.   :272.0   Max.   :2373711   Max.   :2415841  
##     prod_nbr       prod_name            prod_qty         tot_sales      
##  Min.   :  1.00   Length:246742      Min.   :  1.000   Min.   :  1.700  
##  1st Qu.: 26.00   Class :character   1st Qu.:  2.000   1st Qu.:  5.800  
##  Median : 53.00   Mode  :character   Median :  2.000   Median :  7.400  
##  Mean   : 56.35                      Mean   :  1.908   Mean   :  7.321  
##  3rd Qu.: 87.00                      3rd Qu.:  2.000   3rd Qu.:  8.800  
##  Max.   :114.00                      Max.   :200.000   Max.   :650.000  
```

There are no nulls in the columns but product quantity appears to have an outlier which we should investigate further. Let's investigate further the case where 200 packets of chips are bought in one transaction.

```r
clean_transaction %>% group_by(prod_name) %>% filter(prod_qty == 200)
```

Filter the dataset to find the outlier

```
## # A tibble: 2 x 8
## # Groups:   prod_name [1]
##   date       store_nbr lylty_card_nbr txn_id prod_nbr prod_name          prod_qty
##   <date>         <dbl>          <dbl>  <dbl>    <dbl> <chr>                 <dbl>
## 1 2018-08-19       226         226000 226201        4 Dorito Corn Chp ~      200
## 2 2019-05-20       226         226000 226210        4 Dorito Corn Chp ~      200
## # i 1 more variable: tot_sales <dbl>
```

There are two transactions where 200 packets of chips are bought in one transaction and both of these transactions were by the same customer

```r
clean_transaction %>% filter (lylty_card_nbr==226000)
```

See if the customer has had other transactions

```
## # A tibble: 2 x 8
##   date       store_nbr lylty_card_nbr txn_id prod_nbr prod_name          prod_qty
##   <date>         <dbl>          <dbl>  <dbl>    <dbl> <chr>                 <dbl>
## 1 2018-08-19       226         226000 226201        4 Dorito Corn Chp ~      200
## 2 2019-05-20       226         226000 226210        4 Dorito Corn Chp ~      200
## # i 1 more variable: tot_sales <dbl>
```

It looks like this customer (226000) has only had the two transactions over the year and is not an ordinary retail customer. The customer might be buying chips for commercial purposes instead. We'll remove this loyalty card number from further analysis

```r
new_transaction <- clean_transaction %>% filter(lylty_card_nbr!=226000)
```

Filter out the customer based on the loyalty card numbe Look at the number of transaction lines over time to see if there are any obvious data issues such as missing data #### Count the number of transactions by date

```r
new_transaction_dt <- as.data.table(new_transaction)
new_transaction_dt[, .N, by = date]
```

```
##            date   N
##   1: 2018-10-17 682
##   2: 2019-05-14 705
##   3: 2019-05-20 707
##   4: 2018-08-17 663
##   5: 2018-08-18 683
##  ---
## 360: 2018-12-08 622
## 361: 2019-01-30 689
## 362: 2019-02-09 671
## 363: 2018-08-31 658
## 364: 2019-02-12 684
```

There's only 364 rows, meaning only 364 dates which indicates a missing date. Let's create a sequence of dates from 1 Jul 2018 to 30 Jun 2019 and use this to create a chart of number of transactions over time to find the missing date

Create a sequence of dates and join this the count of transactions by date Create a column of dates that includes every day from 1 Jul 2018 to 30 Jun 2019, and join it onto the data to fill in the missing day

```
#### Sequence of date
all_dates <- data.table(seq(as.Date("2018-07-01"), as.Date("2019-06-30"), by = "day"))
setnames(all_dates,"date")
all_dates
```

```
##            date
##   1: 2018-07-01
##   2: 2018-07-02
##   3: 2018-07-03
##   4: 2018-07-04
##   5: 2018-07-05
##  ---
## 361: 2019-06-26
## 362: 2019-06-27
## 363: 2019-06-28
## 364: 2019-06-29
## 365: 2019-06-30
```

```
#### Join squence of date and new_transaction date
transaction_by_day <- merge(data.table(all_dates),new_transaction_dt [, .N, by = date], all = TRUE)
transaction_by_day
```

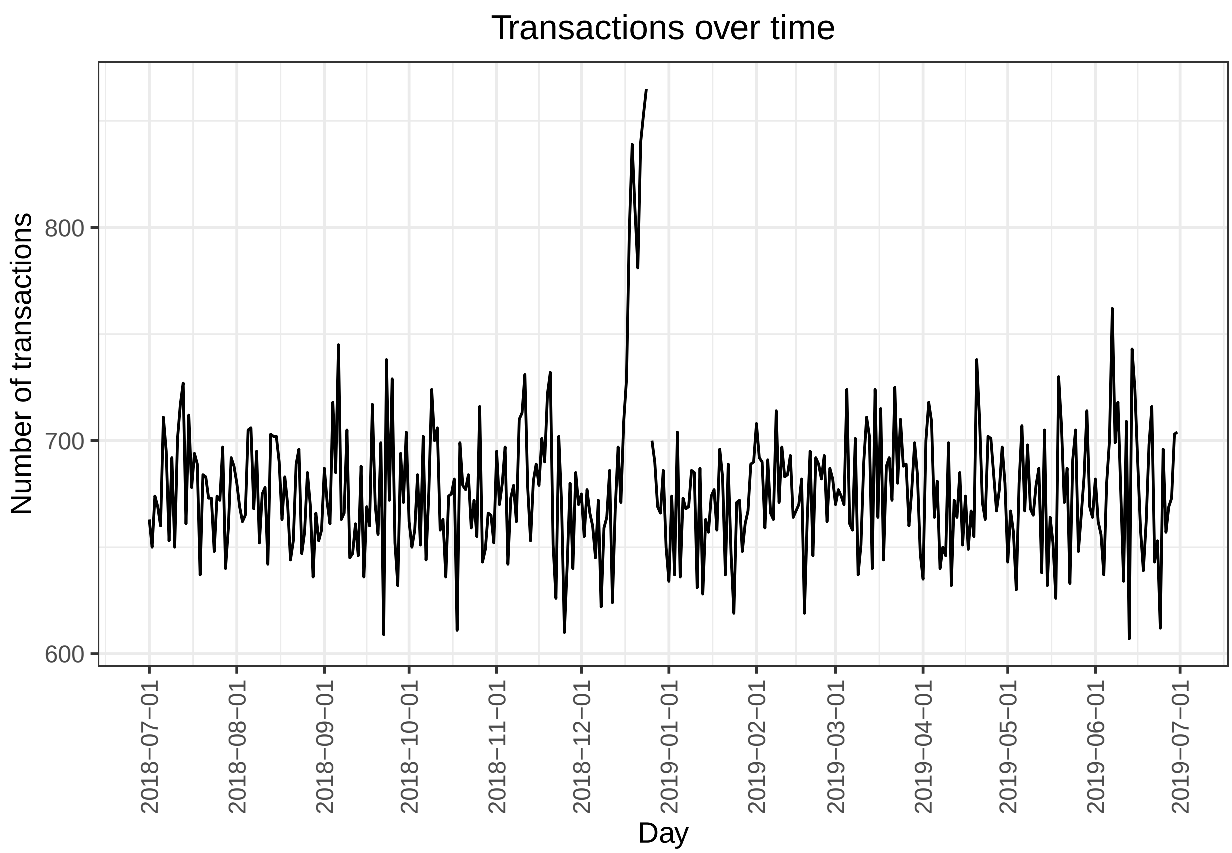
```
##            date   N
##   1: 2018-07-01 663
##   2: 2018-07-02 650
##   3: 2018-07-03 674
##   4: 2018-07-04 669
##   5: 2018-07-05 660
##  ---
## 361: 2019-06-26 657
## 362: 2019-06-27 669
## 363: 2019-06-28 673
## 364: 2019-06-29 703
## 365: 2019-06-30 704
```

```
theme_set(theme_bw())
theme_update(plot.title = element_text(hjust = 0.5))
```

Setting plot themes to format graphs

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

Plot transactions over time

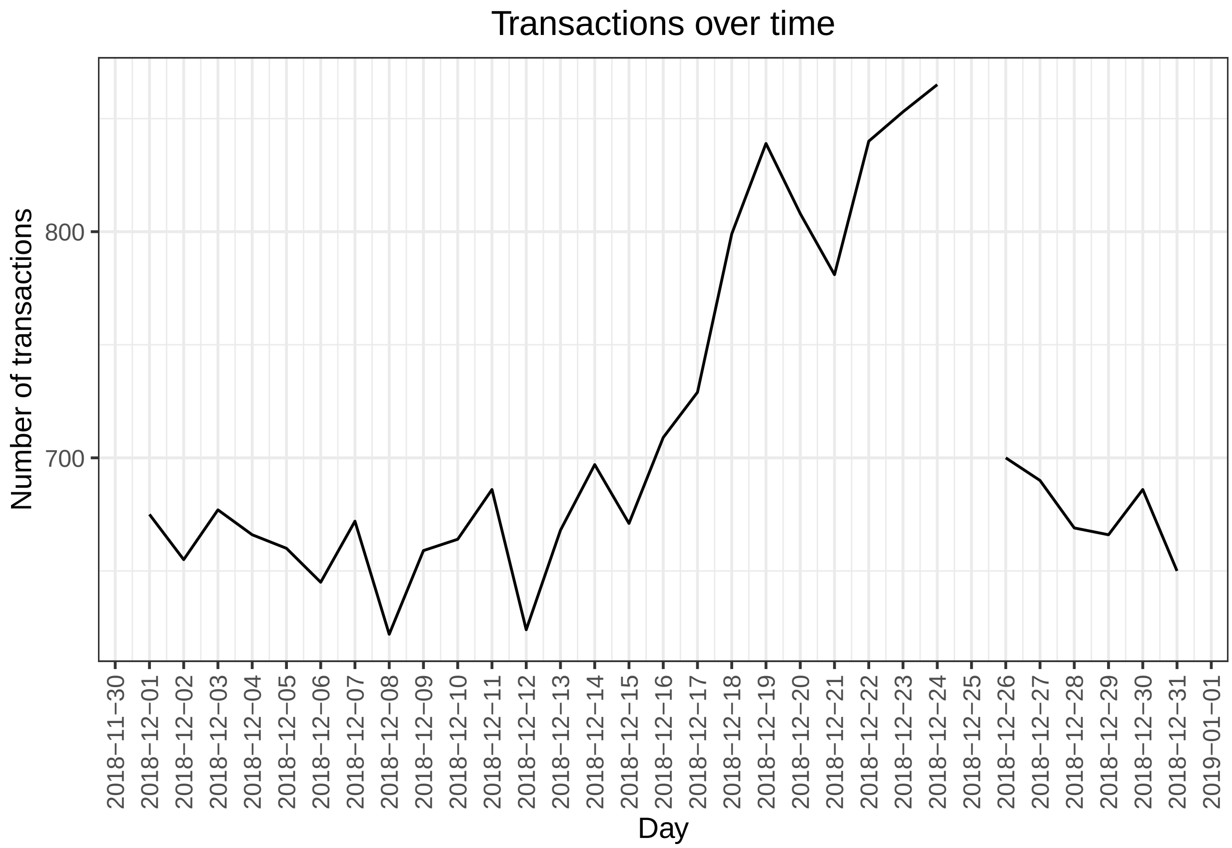


There is an increase in purchases in December and a break in late December.

```
december_data <- subset(transaction_by_day, format(date, "%m") == "12")

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

Filter to December and look at individual days



We can see that the increase in sales occurs in the lead-up to Christmas and that there are zero sales on Christmas day itself. This is due to shops being closed on Christmas day.

Now that we are satisfied that the data no longer has outliers, we can move on to creating other features such as brand of chips or pack size from PROD_NAME. We will start with pack size. #### Pack size We can work this out by taking the digits that are in prod_name

```
new_transaction_dt[, pack_size := parse_number(prod_name)]
new_transaction_dt
```

```
##                 date store_nbr lylty_card_nbr txn_id prod_nbr
##      1: 2018-10-17         1           1000      1        5
##      2: 2019-05-14         1           1307    348       66
##      3: 2019-05-20         1           1343    383       61
##      4: 2018-08-17         2           2373    974       69
##      5: 2018-08-18         2           2426   1038      108
##     ---
## 246736: 2019-03-09       272         272319 270088       89
## 246737: 2018-08-13       272         272358 270154       74
## 246738: 2018-11-06       272         272379 270187       51
## 246739: 2018-12-27       272         272379 270188       42
## 246740: 2018-09-22       272         272380 270189       74
##                                        prod_name prod_qty tot_sales pack_size
##      1:   Natural Chip        Compny SeaSalt175g        2       6.0       175
##      2:                   CCs Nacho Cheese    175g        3       6.3       175
##      3:   Smiths Crinkle Cut  Chips Chicken 170g        2       2.9       170
```

```
##      4:   Smiths Chip Thinly  S/Cream&Onion 175g        5      15.0       175
##      5: Kettle Tortilla ChpsHny&Jlpno Chili 150g        3      13.8       150
##     ---
## 246736:  Kettle Sweet Chilli And Sour Cream 175g        2      10.8       175
## 246737:            Tostitos Splash Of  Lime 175g        1       4.4       175
## 246738:                 Doritos Mexicana    170g        2       8.8       170
## 246739:  Doritos Corn Chip Mexican Jalapeno 150g        2       7.8       150
## 246740:            Tostitos Splash Of  Lime 175g        2       8.8       175
```

```
transaction_pack_size <- new_transaction_dt[, .N,pack_size][order(pack_size)]
transaction_pack_size
```

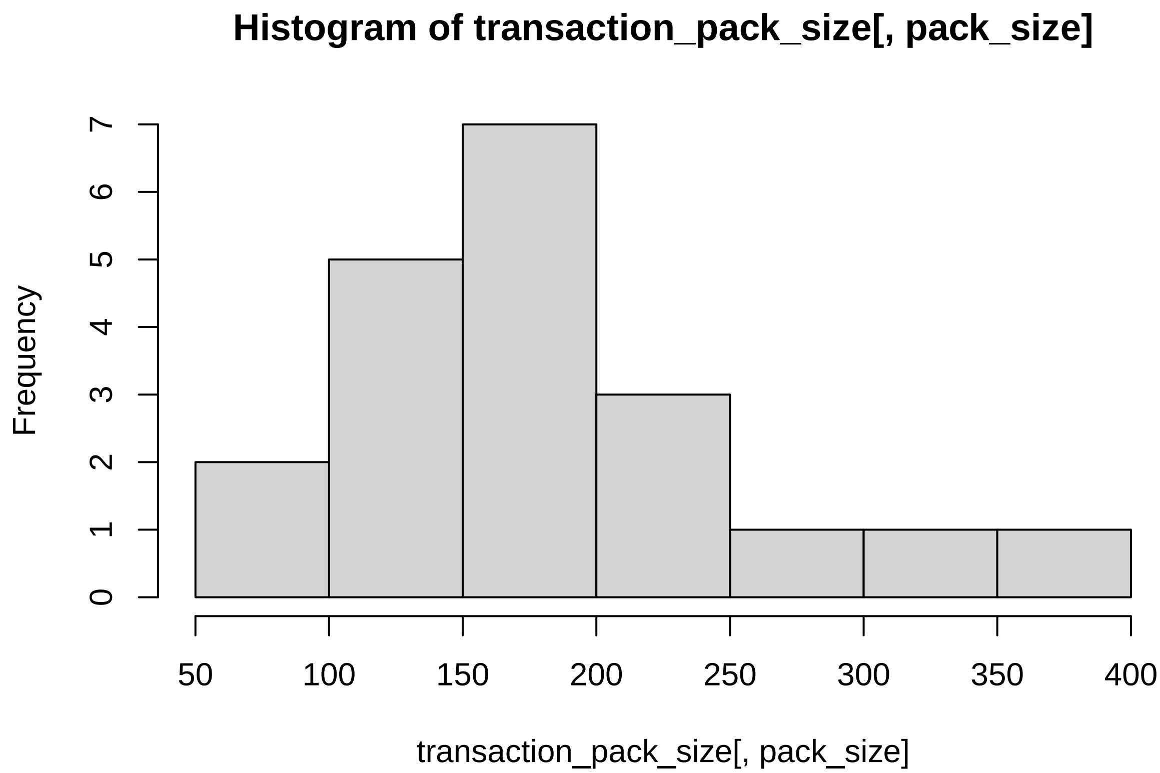
Check if the pack sizes look sensible

```
##     pack_size     N
##  1:        70  1507
##  2:        90  3008
##  3:       110 22387
##  4:       125  1454
##  5:       134 25102
##  6:       135  3257
##  7:       150 40203
##  8:       160  2970
##  9:       165 15297
## 10:       170 19983
## 11:       175 66390
## 12:       180  1468
## 13:       190  2995
## 14:       200  4473
## 15:       210  6272
## 16:       220  1564
## 17:       250  3169
## 18:       270  6285
## 19:       330 12540
## 20:       380  6416
```

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

```
hist(transaction_pack_size[,pack_size])
```

Plot a histogram showing the number of transactions by pack size



Create brands Create a column which contains the brand of the product, by extracting it from the product name

```
brand_transaction <- new_transaction_dt %>%
  mutate(brand = toupper (str_extract(prod_name, "[a-zA-Z]+")))
brand_transaction%>% select(brand)%>%group_by(brand)
```

```
## # A tibble: 246,740 x 1
## # Groups:   brand [28]
##    brand
##    <chr>
##  1 NATURAL
##  2 CCS
##  3 SMITHS
##  4 SMITHS
##  5 KETTLE
##  6 SMITHS
##  7 GRAIN
##  8 DORITOS
##  9 GRAIN
## 10 SMITHS
## # i 246,730 more rows
```

Some of the brand names look like they are of the same brands - such as RED and RRD, which are both Red Rock Deli chips

```r
brand_transaction[brand== "RED",brand := "RRD"]
brand_transaction[brand== "SNBTS",brand := "SUNBITES"]
brand_transaction[brand== "INFZNS",brand := "INFUZIONS"]
brand_transaction[brand== "WW",brand := "WOOLWORTHS"]
brand_transaction[brand== "SMITH",brand := "SMITHS"]
brand_transaction[brand== "NCC",brand := "NATURAL"]
brand_transaction[brand== "DORITO",brand := "DORITOS"]
brand_transaction[brand== "GRAIN",brand := "GRNWVES"]
fix_transaction <- brand_transaction
fix_transaction
```

Clean brand names

```
##               date store_nbr lylty_card_nbr txn_id prod_nbr
##      1: 2018-10-17         1           1000      1        5
##      2: 2019-05-14         1           1307    348       66
##      3: 2019-05-20         1           1343    383       61
##      4: 2018-08-17         2           2373    974       69
##      5: 2018-08-18         2           2426   1038      108
##     ---
## 246736: 2019-03-09       272         272319 270088       89
## 246737: 2018-08-13       272         272358 270154       74
## 246738: 2018-11-06       272         272379 270187       51
## 246739: 2018-12-27       272         272379 270188       42
## 246740: 2018-09-22       272         272380 270189       74
##                                        prod_name prod_qty tot_sales pack_size
##      1:   Natural Chip        Compny SeaSalt175g        2       6.0       175
##      2:                 CCs Nacho Cheese    175g        3       6.3       175
##      3:   Smiths Crinkle Cut  Chips Chicken 170g        2       2.9       170
##      4:   Smiths Chip Thinly  S/Cream&Onion 175g        5      15.0       175
##      5: Kettle Tortilla ChpsHny&Jlpno Chili 150g        3      13.8       150
##     ---
## 246736:  Kettle Sweet Chilli And Sour Cream 175g        2      10.8       175
## 246737:            Tostitos Splash Of  Lime 175g        1       4.4       175
## 246738:                 Doritos Mexicana    170g        2       8.8       170
## 246739:  Doritos Corn Chip Mexican Jalapeno 150g        2       7.8       150
## 246740:            Tostitos Splash Of  Lime 175g        2       8.8       175
##            brand
##      1:  NATURAL
##      2:      CCS
##      3:   SMITHS
##      4:   SMITHS
##      5:   KETTLE
##     ---
## 246736:   KETTLE
## 246737: TOSTITOS
## 246738:  DORITOS
## 246739:  DORITOS
## 246740: TOSTITOS
```

Examining Customer Data

```r
purchase_df <- clean_names(purchase_behaviour)
summary(purchase_df)
```

```
##  lylty_card_nbr     lifestage         premium_customer
##  Min.   :   1000   Length:72637       Length:72637
##  1st Qu.:  66202   Class :character   Class :character
##  Median : 134040   Mode  :character   Mode  :character
##  Mean   : 136186
##  3rd Qu.: 203375
##  Max.   :2373711
```

```r
all_data <- merge(fix_transaction, purchase_df, all.x = TRUE)
all_data
```

Merge transaction data to customer data

```
##         lylty_card_nbr       date store_nbr txn_id prod_nbr
##      1:           1000 2018-10-17         1      1        5
##      2:           1002 2018-09-16         1      2       58
##      3:           1003 2019-03-07         1      3       52
##      4:           1003 2019-03-08         1      4      106
##      5:           1004 2018-11-02         1      5       96
##     ---
## 246736:        2370651 2018-08-03        88 240350        4
## 246737:        2370701 2018-12-08        88 240378       24
## 246738:        2370751 2018-10-01        88 240394       60
## 246739:        2370961 2018-10-24        88 240480       70
## 246740:        2373711 2018-12-14        88 241815       16
##                                       prod_name prod_qty tot_sales pack_size
##      1:   Natural Chip        Compny SeaSalt175g        2       6.0       175
##      2:    Red Rock Deli Chikn&Garlic Aioli 150g        1       2.7       150
##      3:    Grain Waves Sour     Cream&Chives 210G        1       3.6       210
##      4:   Natural ChipCo      Hony Soy Chckn175g        1       3.0       175
##      5:           WW Original Stacked Chips 160g        1       1.9       160
##     ---
## 246736:         Dorito Corn Chp     Supreme 380g        2      13.0       380
## 246737:    Grain Waves         Sweet Chilli 210g        2       7.2       210
## 246738:     Kettle Tortilla ChpsFeta&Garlic 150g        2       9.2       150
## 246739:  Tyrrells Crisps     Lightly Salted 165g        2       8.4       165
## 246740: Smiths Crinkle Chips Salt & Vinegar 330g        2      11.4       330
##              brand              lifestage premium_customer
##      1:    NATURAL  YOUNG SINGLES/COUPLES          Premium
##      2:        RRD  YOUNG SINGLES/COUPLES       Mainstream
##      3:    GRNWVES         YOUNG FAMILIES           Budget
##      4:    NATURAL         YOUNG FAMILIES           Budget
##      5: WOOLWORTHS  OLDER SINGLES/COUPLES       Mainstream
##     ---
```

```
## 246736:    DORITOS MIDAGE SINGLES/COUPLES       Mainstream
## 246737:    GRNWVES         YOUNG FAMILIES       Mainstream
## 246738:     KETTLE         YOUNG FAMILIES          Premium
## 246739:   TYRRELLS         OLDER FAMILIES           Budget
## 246740:     SMITHS  YOUNG SINGLES/COUPLES       Mainstream
```

As the number of rows in all_data is the same as that of clean_transaction, we can be sure that no duplicates were created. This is because we created all_data by setting all.x = TRUE (in other words, a left join) which means take all the rows in clean_transaction and find rows with matching values in shared columns and then joining the details in these rows to the x or the first mentioned table.

```r
skim_without_charts(all_data)
```

See if any transactions did not have a matched customer

Table 4: Data summary

Name	all_data
Number of rows	246740
Number of columns	12
Key	lylty_card_nbr
Column type frequency:	
character	4
Date	1
numeric	7
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
prod_name	0	1	17	40	0	105	0
brand	0	1	3	10	0	20	0
lifestage	0	1	8	22	0	7	0
premium_customer	0	1	6	10	0	3	0

Variable type: Date

skim_variable	n_missing	complete_rate	min	max	median	n_unique
date	0	1	2018-07-01	2019-06-30	2018-12-30	364

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
lylty_card_nbr	0	1	135530.25	80715.20	1000.0	70015.00	130367.0	203083.2	2373711.0
store_nbr	0	1	135.05	76.79	1.0	70.00	130.0	203.0	272.0
txn_id	0	1	135130.36	78147.60	1.0	67568.75	135181.5	202652.2	2415841.0
prod_nbr	0	1	56.35	33.70	1.0	26.00	53.0	87.0	114.0
prod_qty	0	1	1.91	0.34	1.0	2.00	2.0	2.0	5.0
tot_sales	0	1	7.32	2.47	1.7	5.80	7.4	8.8	29.5
pack_size	0	1	175.58	59.43	70.0	150.00	170.0	175.0	380.0

There are no nulls. So all our customers in the transaction data has been accounted for in the customer dataset

```r
write.csv(all_data, "all_data.csv",row.names = FALSE)
```

Data exploration is now complete

Data Analysis in Customer Segments

Now that the data is ready for analysis, we can define some metrics of interest to the client: * Who spends the most on chips (total sales), describing customers by lifestage and how premium their general purchasing behaviour is * How many customers are in each segment * How many chips are bought per customer by segment * What's the average chip price by customer segment

We could also ask our data team for more information. Examples are: * The customer's total spend over the period and total spend for each transaction to understand what proportion of their grocery spend is on chips * Proportion of customers in each customer segment overall to compare against the mix of customers who purchase chips

Let's start with calculating total sales by LIFESTAGE and PREMIUM_CUSTOMER and plotting the split by these segments to describe which customer segment contribute most to chip sales. #### Total sales by lifestage and premium_customer

```r
all_data %>%
  group_by(lifestage, premium_customer) %>%
  summarize(total_sales = sum(tot_sales)) %>%
  arrange(desc(total_sales))
```

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

```
## # A tibble: 21 x 3
## # Groups:   lifestage [7]
##    lifestage              premium_customer total_sales
##    <chr>                  <chr>                  <dbl>
##  1 OLDER FAMILIES         Budget               156864.
##  2 YOUNG SINGLES/COUPLES Mainstream           147582.
##  3 RETIREES               Mainstream           145169.
##  4 YOUNG FAMILIES         Budget               129718.
##  5 OLDER SINGLES/COUPLES Budget               127834.
##  6 OLDER SINGLES/COUPLES Mainstream           124648.
##  7 OLDER SINGLES/COUPLES Premium              123538.
```

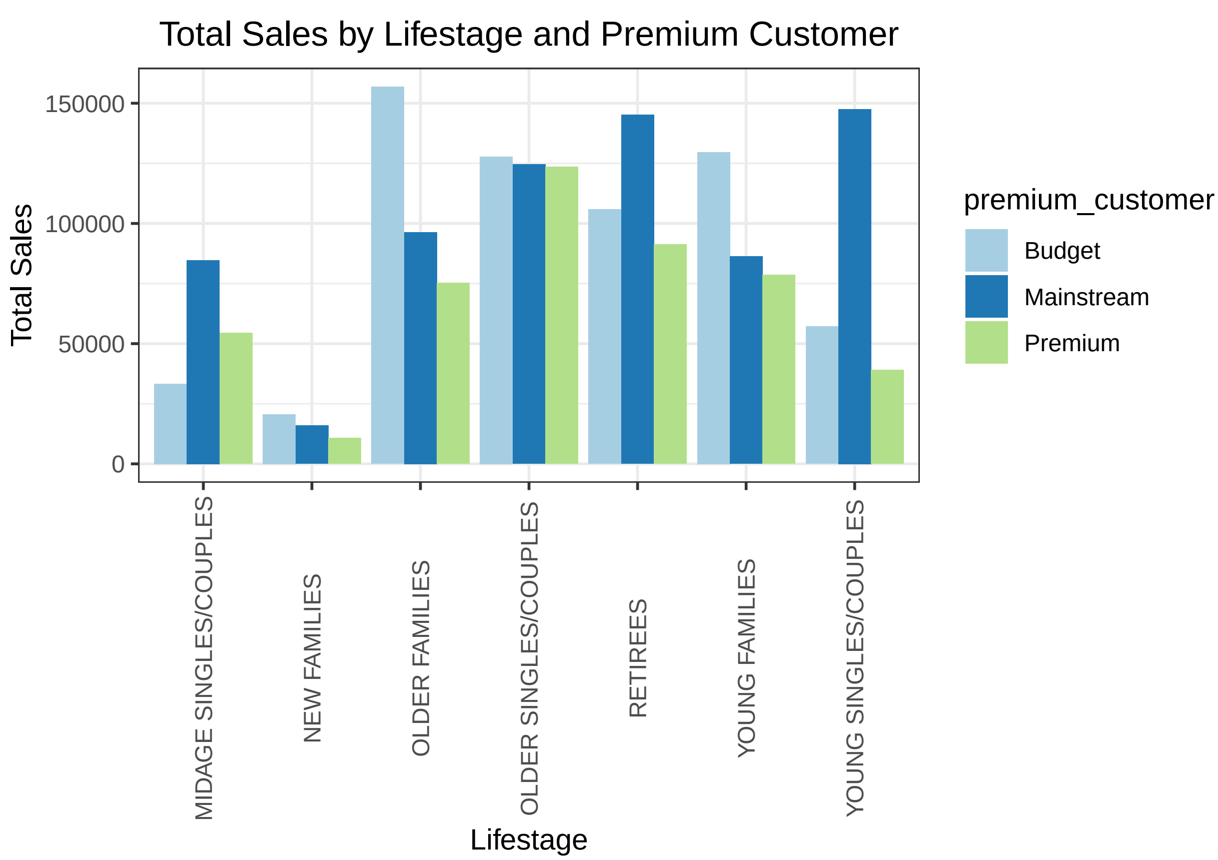
```
## 8 RETIREES           Budget          105916.
## 9 OLDER FAMILIES     Mainstream       96414.
## 10 RETIREES          Premium          91297.
## # i 11 more rows
```

```
total_sales_by_segment <- all_data %>%
  group_by(lifestage, premium_customer) %>%
  summarize(total_sales = sum(tot_sales))
```

Create a plot

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

```
# Plotting the results
ggplot(total_sales_by_segment, aes(x = lifestage, y = total_sales, fill = premium_customer)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Total Sales by Lifestage and Premium Customer",
       x = "Lifestage",
       y = "Total Sales") +
  theme(axis.text.x= element_text(angle=90,vjust=0.5))+
  scale_fill_brewer(palette = "Paired")
```



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

```r
all_data %>%
  group_by(lifestage, premium_customer) %>%
  summarize(total_customer = n_distinct(lylty_card_nbr),
            total_customer = sum(lylty_card_nbr)) %>%
  arrange(desc(total_customer))
```

Number of customers by lifestage and premium_customer

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

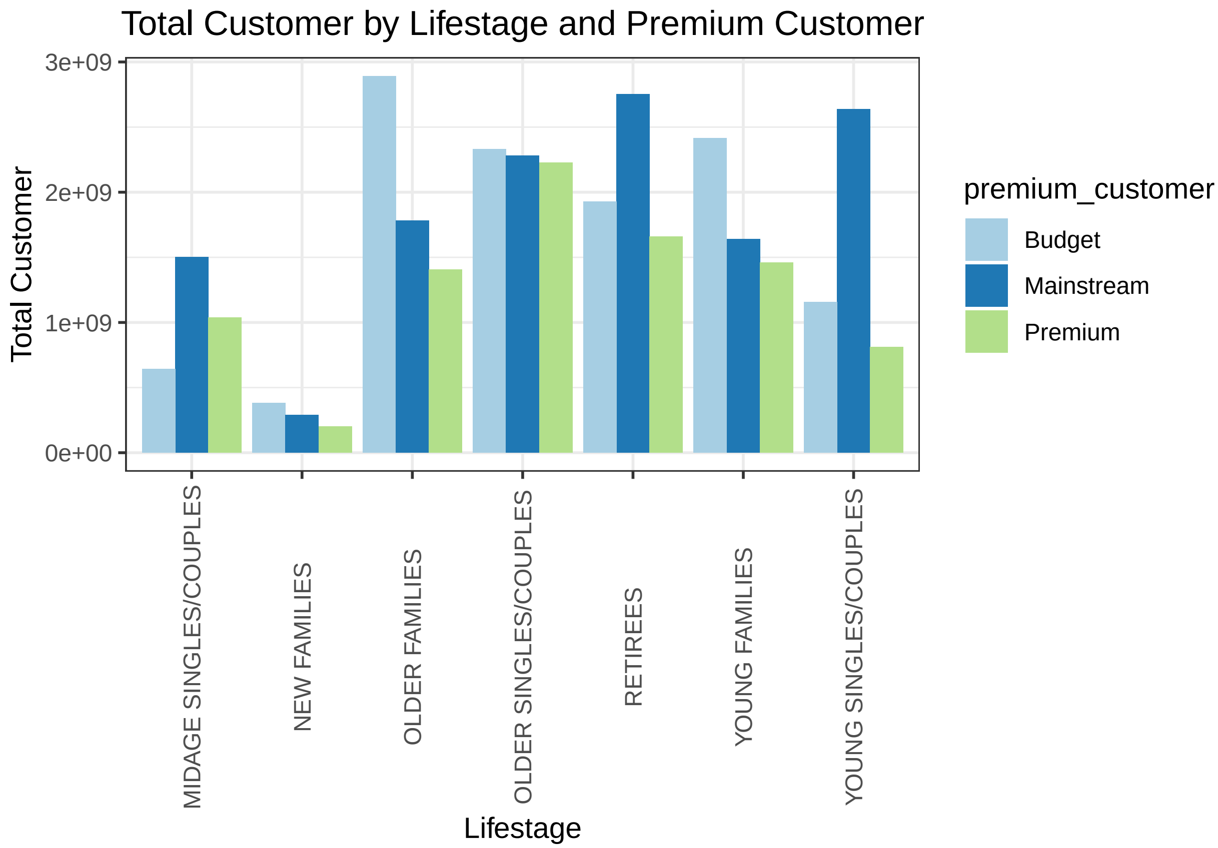
```
## # A tibble: 21 x 3
## # Groups:   lifestage [7]
##    lifestage              premium_customer total_customer
##    <chr>                  <chr>                     <dbl>
##  1 OLDER FAMILIES         Budget               2891942530
##  2 RETIREES               Mainstream           2753153856
##  3 YOUNG SINGLES/COUPLES  Mainstream           2637061979
##  4 YOUNG FAMILIES         Budget               2415761554
##  5 OLDER SINGLES/COUPLES  Budget               2332495098
##  6 OLDER SINGLES/COUPLES  Mainstream           2279764274
##  7 OLDER SINGLES/COUPLES  Premium              2228223157
##  8 RETIREES               Budget               1927702126
##  9 OLDER FAMILIES         Mainstream           1782766792
## 10 RETIREES               Premium              1660094379
## # i 11 more rows
```

```r
total_customer_by_segment <- all_data %>%
  group_by(lifestage, premium_customer) %>%
  summarize(total_customer = n_distinct(lylty_card_nbr),
            total_customer = sum(lylty_card_nbr)) %>%
  arrange(desc(total_customer))
```

Create a plot

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

```r
# Plotting the results
ggplot(total_customer_by_segment, aes(x = lifestage, y = total_customer, fill = premium_customer)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Total Customer by Lifestage and Premium Customer",
       x = "Lifestage",
       y = "Total Customer") +
  theme(axis.text.x= element_text(angle=90,vjust=0.5))+
  scale_fill_brewer(palette = "Paired")
```



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

Higher sales may also be driven by more units of chips being bought per customer.

```r
all_data %>%
  group_by(lifestage, premium_customer) %>%
  summarize(avg_unit = sum(prod_qty)/n_distinct(lylty_card_nbr) ) %>%
  arrange(desc(avg_unit))
```

Average number of units per customer by lifestage and premium_customer

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

```
## # A tibble: 21 x 3
## # Groups:   lifestage [7]
##    lifestage              premium_customer avg_unit
##    <chr>                  <chr>               <dbl>
##  1 OLDER FAMILIES         Mainstream           9.26
##  2 OLDER FAMILIES         Budget               9.08
##  3 OLDER FAMILIES         Premium              9.07
##  4 YOUNG FAMILIES         Budget               8.72
```

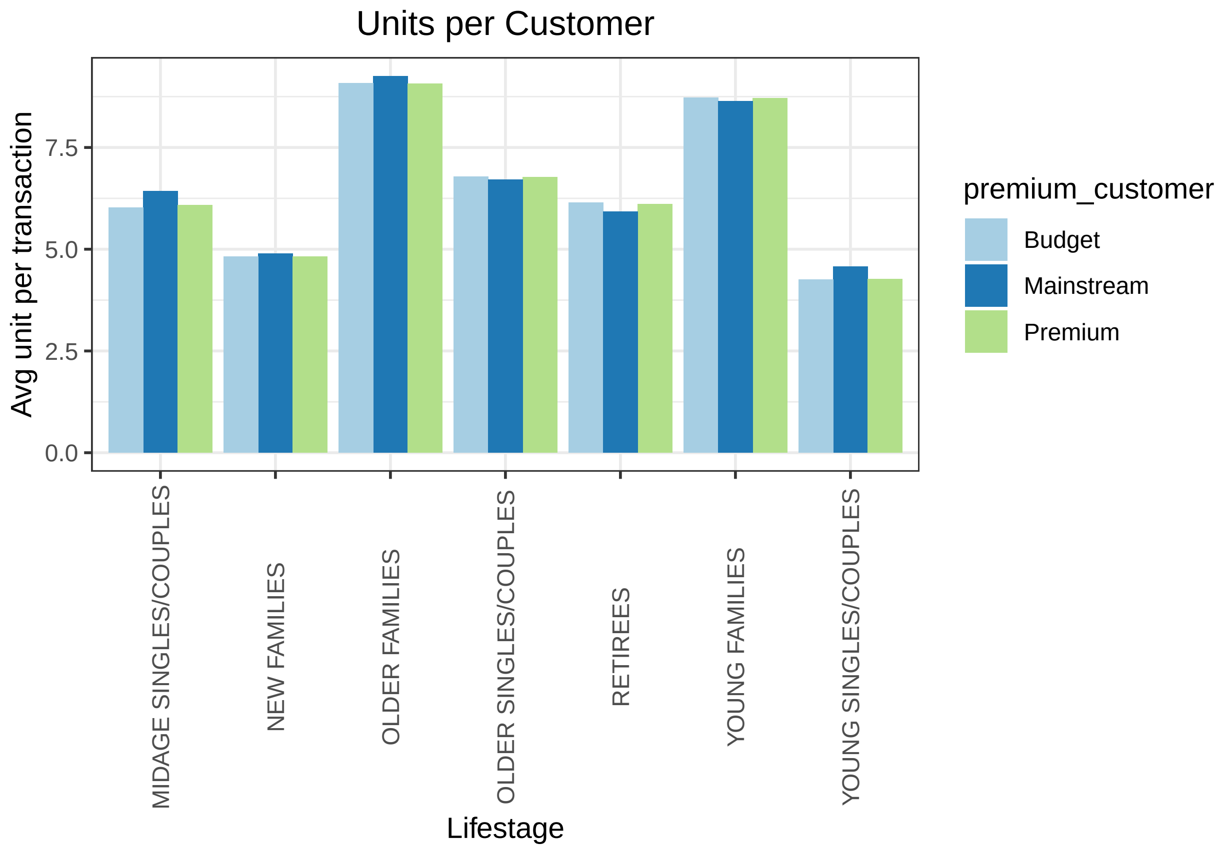
```
##   5 YOUNG FAMILIES         Premium              8.72
##   6 YOUNG FAMILIES         Mainstream           8.64
##   7 OLDER SINGLES/COUPLES  Budget               6.78
##   8 OLDER SINGLES/COUPLES  Premium              6.77
##   9 OLDER SINGLES/COUPLES  Mainstream           6.71
##  10 MIDAGE SINGLES/COUPLES Mainstream           6.43
## # i 11 more rows
```

```r
avg_unit_by_segment <- all_data %>%
  group_by(lifestage, premium_customer) %>%
  summarize(avg_unit = sum(prod_qty)/n_distinct(lylty_card_nbr) ) %>%
  arrange(desc(avg_unit))
```

Create a plot

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

```r
# Plotting the results
ggplot(avg_unit_by_segment, aes(x = lifestage, y = avg_unit, fill = premium_customer)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Units per Customer",
       x = "Lifestage",
       y = "Avg unit per transaction") +
  theme(axis.text.x= element_text(angle=90,vjust=0.5))+
  scale_fill_brewer(palette = "Paired")
```



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

Investigate the average price per unit chips bought for each customer segment as this is also a driver of total sales. #### Average price per unit by lifestage and premium_customer

```
all_data %>%
  group_by(lifestage, premium_customer) %>%
  summarize(avg_price = sum(tot_sales)/sum(prod_qty)) %>%
  arrange(desc(avg_price))
```

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

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

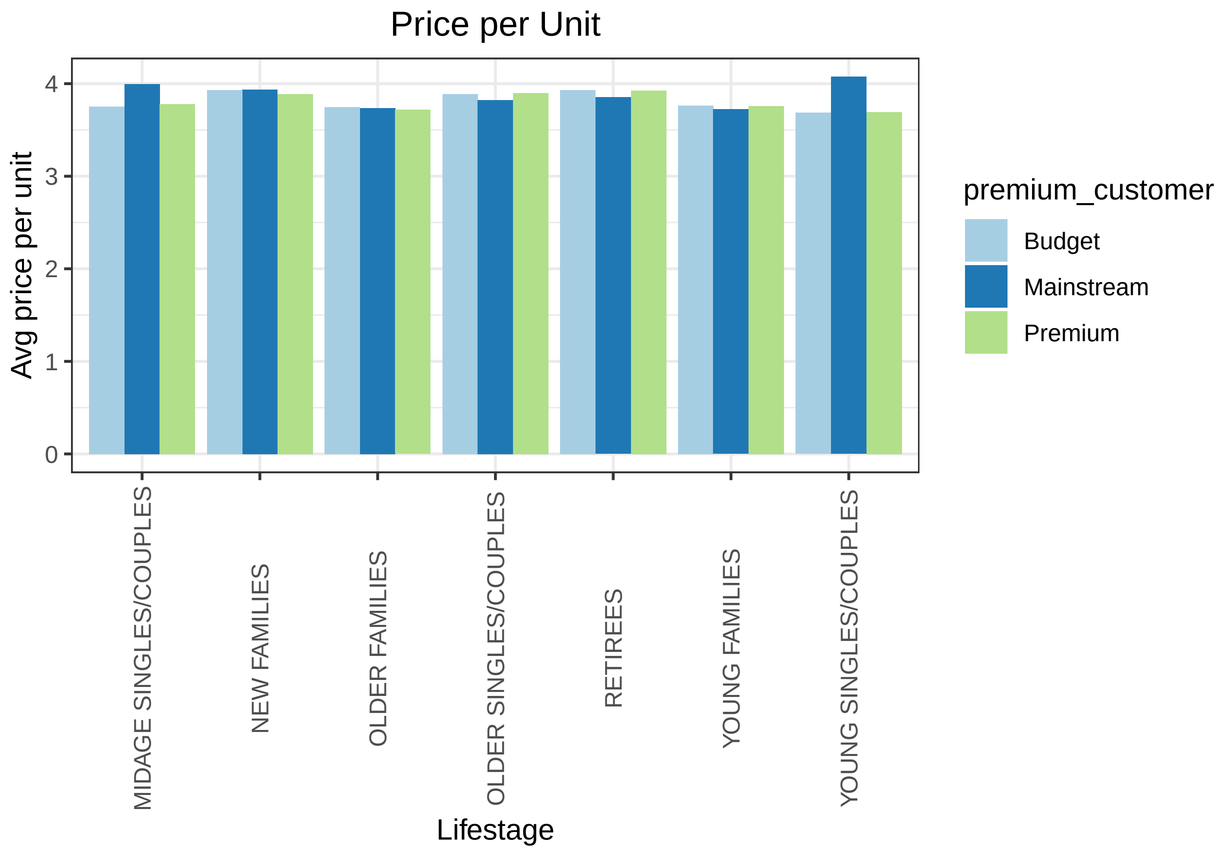
```
## # i 11 more rows
```

```r
avg_price_by_segment <- all_data %>%
  group_by(lifestage, premium_customer) %>%
  summarize(avg_price = sum(tot_sales)/sum(prod_qty)) %>%
  arrange(desc(avg_price))
```

Create a plot

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

```r
# Plotting the results
ggplot(avg_price_by_segment, aes(x = lifestage, y = avg_price, fill = premium_customer)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Price per Unit",
       x = "Lifestage",
       y = "Avg price per unit") +
  theme(axis.text.x= element_text(angle=90,vjust=0.5))+
  scale_fill_brewer(palette = "Paired")
```



Mainstream midage and young singles and couples are more willing to pay more per packet of chips compared to their budget and premium counterparts. This may be due to premium shoppers being more likely to buy healthy snacks and when they buy chips, this is mainly for entertainment purposes rather than their own consumption. This is also supported by there being fewer premium midage and young singles and couples buying chips compared to their mainstream counterparts.

As the difference in average price per unit isn't large, we can check if this difference is statistically different. #### Perform an independent t-test between mainstream vs premium and budget midage and young singles and couples

```
setDT(all_data)
price_per_unit <- all_data[, price := tot_sales/prod_qty]
t.test(all_data[lifestage %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES") & premium_customer =
, all_data[lifestage %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES") & premium_customer != "Ma
, alternative = "greater")
```

```
##
##  Welch Two Sample t-test
##
## data:  all_data[lifestage %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES") & premium_custome
## t = 37.624, df = 54791, p-value < 2.2e-16
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  0.3187234       Inf
## sample estimates:
## mean of x mean of y
##  4.039786  3.706491
```

The t-test results in a p-value of 2.2e-16, i.e. the unit price for mainstream, young and mid-age singles and couples ARE significantly higher than that of budget or premium, young and midage singles and couples.

Deep dive into specific customer segments for insights

We might want to target customer segments that contribute the most to sales to retain them or further increase sales. Let's look at Mainstream - young singles/couples. For instance, let's find out if they tend to buy a particular brand of chips.

```
segment_1 <- all_data[all_data$lifestage == "YOUNG SINGLES/COUPLES" & all_data$premium_customer =="Mains
other <- all_data[!(all_data$lifestage == "YOUNG SINGLES/COUPLES" & all_data$premium_customer =="Mainstr
```

Deep dive into Mainstream, young singles/couples

```
# Calculate total quantities
quantity_segment1 <- segment_1[, sum(segment_1$prod_qty)]
quantity_other <- other[, sum(other$prod_qty)]

# Calculate brand proportions for each segment
quantity_segment1_by_brand <- segment_1[, .(target_segment = sum(prod_qty)/quantity_segment1), by = bran
quantity_other_by_brand <- other[, .(other = sum(prod_qty)/quantity_other), by = brand]
```

```r
# Merge brand proportions
brand_proportions <- merge(quantity_segment1_by_brand,quantity_other_by_brand)[, affinityToBrand := targ

# Order by affinityToBrand
brand_proportions[order(-affinityToBrand)]
```

Brand affinity compared to the rest of the population

```
##          brand target_segment       other affinityToBrand
##  1:   TYRRELLS    0.031552795 0.025692464       1.2280953
##  2:   TWISTIES    0.046183575 0.037876520       1.2193194
##  3:    DORITOS    0.122760524 0.101074684       1.2145526
##  4:     KETTLE    0.197984817 0.165553442       1.1958967
##  5:   TOSTITOS    0.045410628 0.037977861       1.1957131
##  6:   PRINGLES    0.119420290 0.100634769       1.1866703
##  7:       COBS    0.044637681 0.039048861       1.1431238
##  8:  INFUZIONS    0.064679089 0.057064679       1.1334347
##  9:      THINS    0.060372671 0.056986370       1.0594230
## 10:    GRNWVES    0.032712215 0.031187957       1.0488733
## 11:   CHEEZELS    0.017971014 0.018646902       0.9637534
## 12:     SMITHS    0.096369910 0.124583692       0.7735355
## 13:     FRENCH    0.003947550 0.005758060       0.6855694
## 14:    CHEETOS    0.008033126 0.012066591       0.6657329
## 15:        RRD    0.043809524 0.067493678       0.6490908
## 16:    NATURAL    0.019599724 0.030853989       0.6352412
## 17:        CCS    0.011180124 0.018895650       0.5916771
## 18:   SUNBITES    0.006349206 0.012580210       0.5046980
## 19: WOOLWORTHS    0.024099379 0.049427188       0.4875733
## 20:     BURGER    0.002926156 0.006596434       0.4435967
```

We can see that : * Mainstream young singles/couples are 25% more likely to purchase Tyrrells chips compared to the rest of the population * Mainstream young singles/couples are 65% less likely to purchase Burger Rings compared to the rest of the population