QVI_RetailAnalytics

Bena

2023-05-29

Setting up

Load required packages and libraries

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

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

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

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

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

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

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

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

```
library(tidyverse)
```

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

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

Importing data

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

Exploratory Data Analysis

A. Examining transaction data

```
str(transactionData)
```

```
## Classes 'data.table' and 'data.frame':   264839 obs. of  8 variables:
##  $ DATE          : chr  "10/17/2018" "5/14/2019" "5/20/2019" "8/17/2018" ...
##  $ STORE_NBR     : int  1 1 1 2 2 4 4 4 5 7 ...
##  $ LYLTY_CARD_NBR: int  1000 1307 1343 2373 2426 4074 4149 4196 5026 7150 ...
##  $ TXN_ID        : int  1 348 383 974 1038 2982 3333 3539 4525 6900 ...
##  $ PROD_NBR      : int  5 66 61 69 108 57 16 24 42 52 ...
##  $ PROD_NAME     : chr  "Natural Chip        Compny SeaSalt175g" "CCs Nacho Cheese    175g" "Smiths C
rinkle Cut  Chips Chicken 170g" "Smiths Chip Thinly  S/Cream&Onion 175g" ...
##  $ PROD_QTY      : int  2 3 2 5 3 1 1 1 1 2 ...
##  $ TOT_SALES     : num  6 6.3 2.9 15 13.8 5.1 5.7 3.6 3.9 7.2 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

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

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

Variable type: character

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

Variable type: numeric

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

```
colnames(transactionData)
```

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

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

1. Amend data type for DATE column from chr

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

2. Examine the PROD_NAME column

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

Alternatively

```
# product_summary <- transactionData[, .N, PROD_NAME]
```

The data includes other products which aren't chips % we'd like to exclude them from analysis i.e. the "Old El Paso Salsa" products

```
pattern <- "(?i)old el paso"
transactionData <- transactionData %>%
  mutate(CHIPS = str_detect(PROD_NAME,pattern)) %>%
  filter(CHIPS == "FALSE")
```

3. Create size column from product name

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

Remove numbers and special characters from product names

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

4. Summary to check data types, outliers, etc

```
summary(transactionData)
```

```
##       DATE              STORE_NBR     LYLTY_CARD_NBR       TXN_ID
##  Min.   :2018-07-01   Min.   :  1.0   Min.   :   1000   Min.   :      1
##  1st Qu.:2018-09-30   1st Qu.: 70.0   1st Qu.:  70031   1st Qu.:  67669
##  Median :2018-12-30   Median :130.0   Median : 130354   Median : 135124
##  Mean   :2018-12-30   Mean   :135.1   Mean   : 135539   Mean   : 135149
##  3rd Qu.:2019-03-31   3rd Qu.:203.0   3rd Qu.: 203078   3rd Qu.: 202629
##  Max.   :2019-06-30   Max.   :272.0   Max.   :2373711   Max.   :2415841
##  NA's   :1            NA's   :3       NA's   :3         NA's   :3
##     PROD_NBR       PROD_NAME            PROD_QTY         TOT_SALES
##  Min.   :  1.00   Length:255515      Min.   :  1.000   Min.   :  1.500
##  1st Qu.: 28.00   Class :character   1st Qu.:  2.000   1st Qu.:  5.400
##  Median : 53.00   Mode  :character   Median :  2.000   Median :  7.400
##  Mean   : 56.45                      Mean   :  1.907   Mean   :  7.215
##  3rd Qu.: 86.00                      3rd Qu.:  2.000   3rd Qu.:  8.800
##  Max.   :114.00                      Max.   :200.000   Max.   :650.000
##  NA's   :3                           NA's   :3         NA's   :3
##    CHIPS              SIZE
##  Mode :logical   Min.   : 70.0
##  FALSE:255515    1st Qu.:150.0
##                  Median :170.0
##                  Mean   :178.1
##                  3rd Qu.:175.0
##                  Max.   :380.0
##                  NA's   :3
```

`PROD_QTY` & `TOT_SALES` seem to have outliers. The Max is way above the Mean

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

# or transactionData[PROD_QTY == 200, ]
```

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

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

# or transactionData[LYLTY_CARD_NBR == 226000, ]
```

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

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

5. We can get brand names from product names Standardize them first by making them all lower case

```
transactionData$PROD_NAME <- tolower(transactionData$PROD_NAME)
```

```
transactionData <- transactionData %>%
  mutate(BRAND_CHR = str_sub(regexpr(pattern = ' ', PROD_NAME)-1)) %>%
  mutate(BRAND = str_sub(PROD_NAME, 1, BRAND_CHR))
```

Some brand names seem to be repeated using different words. There's still some cleaning up required eg natural chips company appears as natural = ncc Red Rock Deli appears as rrd = red doritos = dorito smiths = smith infuzions = infzns ww = woolworths grain = grnwves snbts = sunbites

```
transactionData <- transactionData %>%
  mutate(BRAND = case_when(
    BRAND == "ncc" ~ "natural",
    BRAND == "red" ~ "rrd",
    BRAND == "dorito" ~ "doritos",
    BRAND == "smith" ~ "smiths",
    BRAND == "infzns" ~ "infuzions",
    BRAND == "ww" ~ "woolworths",
    BRAND == "grain" ~ "grnwves",
    BRAND == "snbts" ~ "sunbites",
    .default = as.character(BRAND)
    ))
```

How many brands are in the data?

```
transactionData[, .N, by = BRAND][order(BRAND)]
```

```
##          BRAND     N
##  1:     burger  1564
##  2:        ccs  4551
##  3:    cheetos  2927
##  4:   cheezels  4603
##  5:       cobs  9693
##  6:    doritos 28145
##  7:     french  1418
##  8:    grnwves  7740
##  9:  infuzions 14201
## 10:     kettle 41288
## 11:    natural  7469
## 12:   pringles 25102
## 13:        rrd 17779
## 14:     smiths 31823
## 15:   sunbites  3008
## 16:      thins 14075
## 17:   tostitos  9471
## 18:   twisties  9454
## 19:   tyrrells  6442
## 20: woolworths 14757
```

```
# or brand_count <- transactionData %>%
#group_by(BRAND) %>%
#summarize(number = n())
```

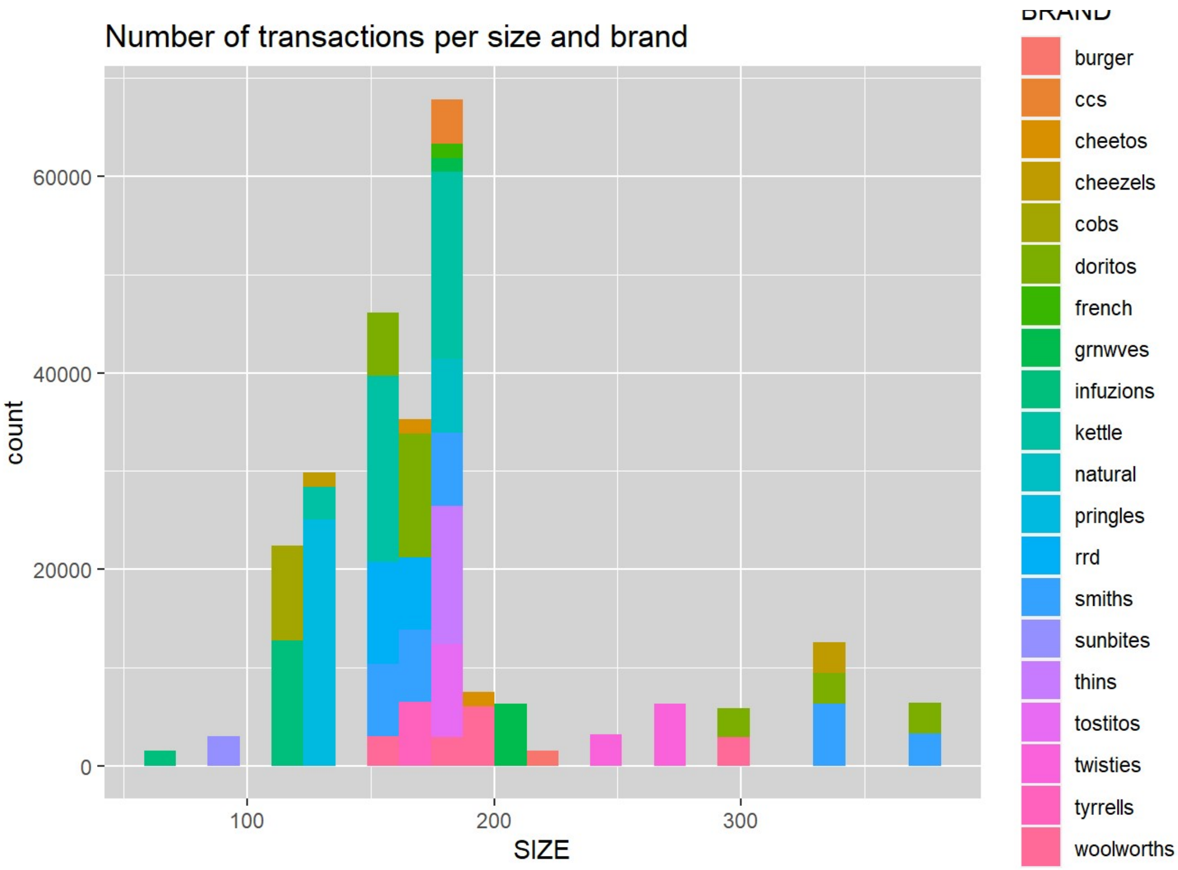
6. Let's look into `SIZE`

```
size_count <- transactionData %>%
  group_by(SIZE) %>%
  summarise(number = n())

# or transactionData[, .N, by = SIZE][order(SIZE)]
```

The package sizes look reasonable, thus ok to proceed We shall visualize the number bought for each size & brand

```
transactionData %>%
  ggplot(aes(x = SIZE, fill = BRAND)) +
  geom_histogram(bins = 25) +
  labs(title = "Number of transactions per size and brand") +
  theme(panel.background = element_rect(fill = "lightgrey"))
```



Alternatively, the tabular summary is as follows:

```
size_count2 <- transactionData %>%
  group_by(SIZE,BRAND) %>%
  summarise(number = n()) %>%
  arrange(-number)
```

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

The top3 brands & sizes are pringles 134g, kettle 175g & kettle 150g

7. Check out `TOT_SALES`

Find the cost per unit

```
transactionData <- transactionData %>%
  mutate(UNIT_COST = TOT_SALES/PROD_QTY)
```

Revisit brand summary

```
brand_summary <- transactionData %>%
  group_by(BRAND) %>%
  summarize(number_bought = n(),
            avg_qnty = mean(PROD_QTY),
            min_qnty = min(PROD_QTY),
            max_qnty = max(PROD_QTY),
            avg_sales = mean(UNIT_COST),
            total_sales = sum(TOT_SALES)) %>%
  arrange(-avg_qnty)
```

Minimum quantity for each brand is 1 and maximum quantity ranges between 3-5. Top 3 brands with the most total_sales are kettle, doritos & smiths Top 3 brands with the most avg_sales are kettle, cheezels & twisties Top 3 brands with the most avg_qnty are twisties, cobs & tostitos

8. About the dates

The transactions are for a full year.

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

# or date_summary <- transactionData[, .N, by = DATE] %>%
# arrange(DATE)
```

But there are 364 days only in our data. We’ll create a sequence for all calendar dates then merge with date_summary

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

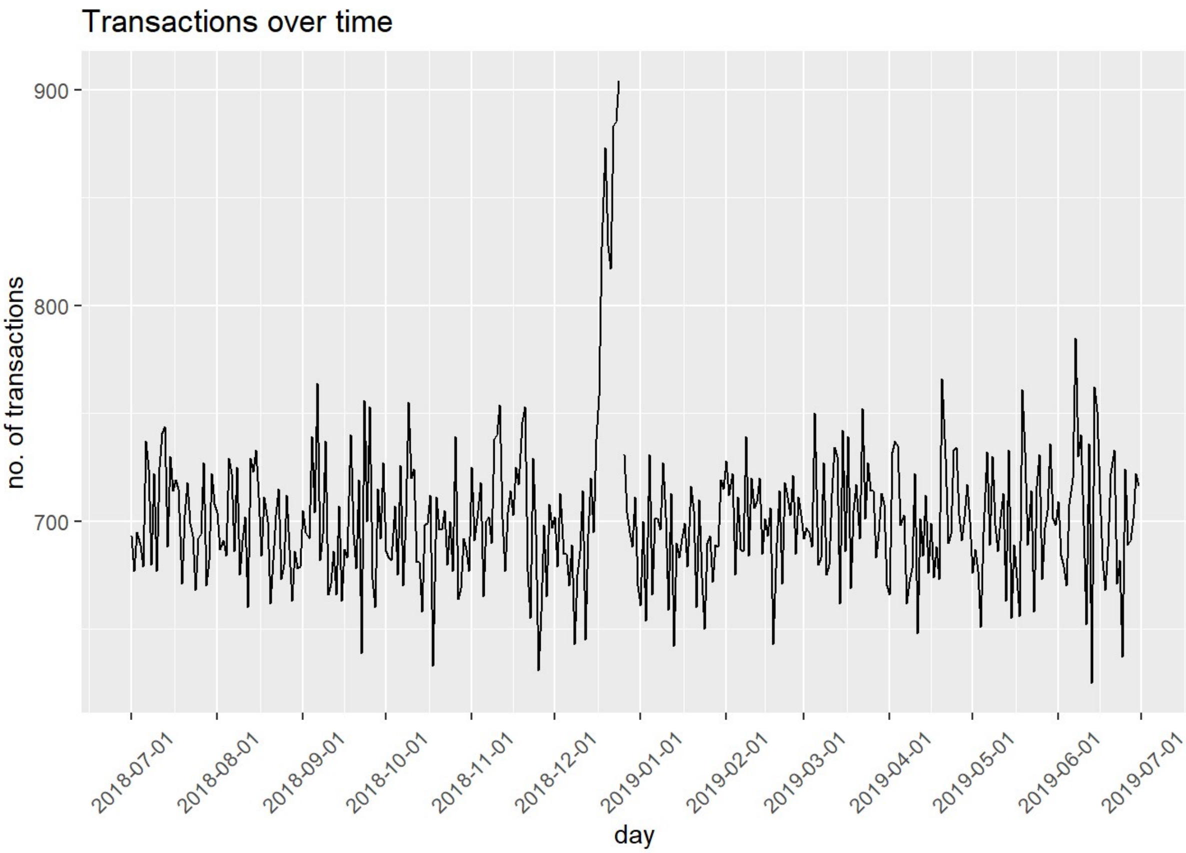
```
transactions_by_day <- alldates %>%
  left_join(date_summary, by = "DATE")

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

Trend Analysis

Plot the dates

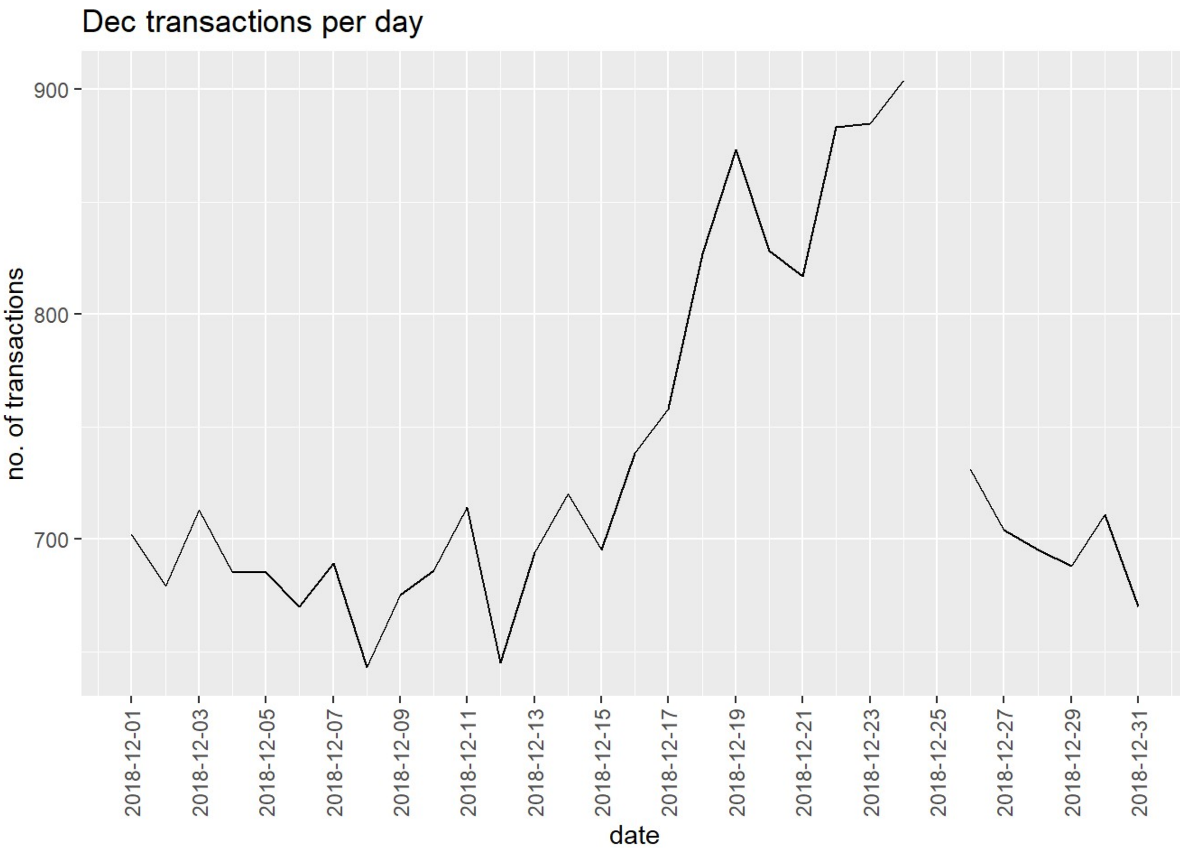
```
ggplot(data = transactions_by_day, aes(x = DATE, y=tranx_per_day)) +
  geom_line() +
  labs(x = "day", y = "no. of transactions", title = "Transactions over time") +
  scale_x_date(breaks = "1 month") +
  theme(axis.text.x = element_text(angle = 45, vjust = 0.5))
```



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

a. More about Dec data

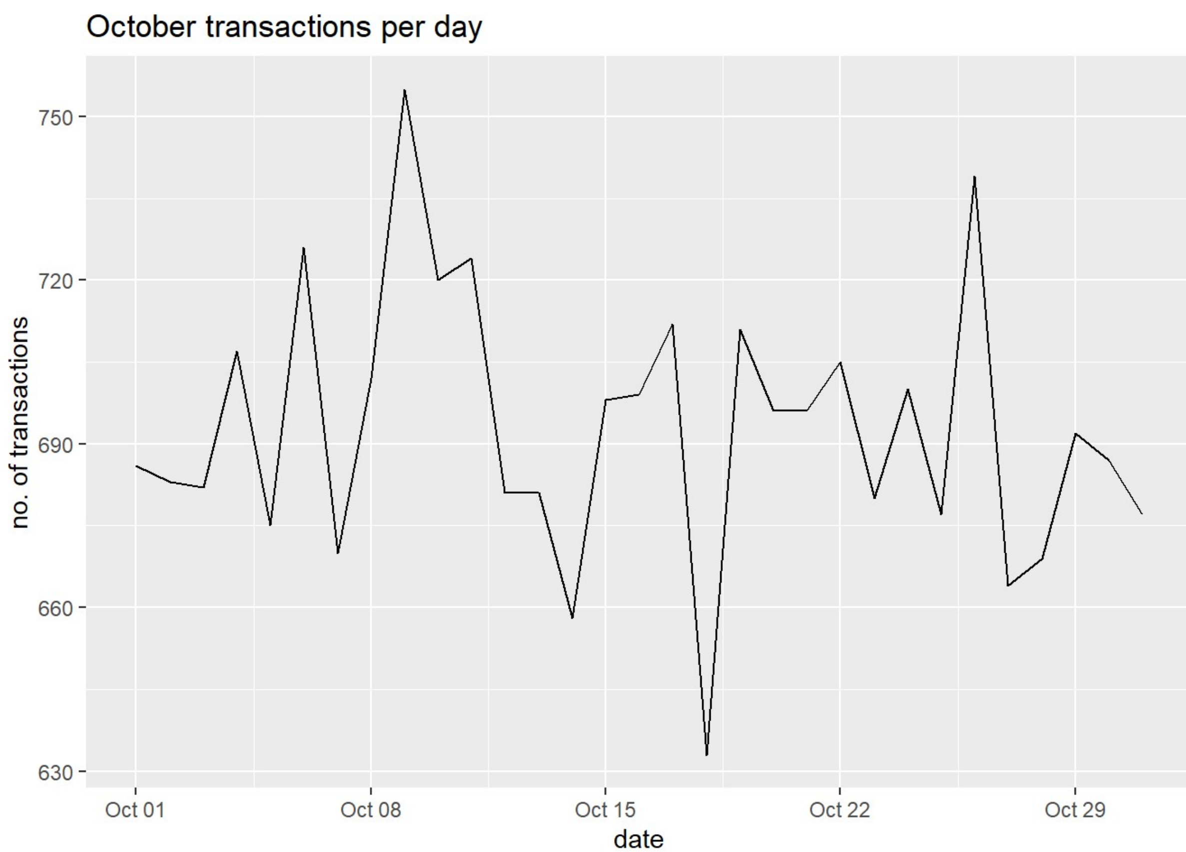
```
transactions_by_day %>%
  filter(DATE >= '2018-12-01' & DATE <= '2018-12-31') %>%
  ggplot(aes(x = DATE, y = tranx_per_day)) +
  geom_line() +
  labs(x = "date", y = "no. of transactions", title = "Dec transactions per day") +
  scale_x_date(breaks = "2 day") +
  theme(axis.text.x = element_text(angle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



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

b. More about Oct data

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



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

Exploratory Data Analysis

B. Examining customer data

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

Data summary

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

Variable type: character

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

Variable type: numeric

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

```
str(customerData)
```

```
## Classes 'data.table' and 'data.frame':   72637 obs. of  3 variables:
##  $ LYLTY_CARD_NBR  : int  1000 1002 1003 1004 1005 1007 1009 1010 1011 1012 ...
##  $ LIFESTAGE       : chr  "YOUNG SINGLES/COUPLES" "YOUNG SINGLES/COUPLES" "YOUNG FAMILIES" "OLDER SIN
GLES/COUPLES" ...
##  $ PREMIUM_CUSTOMER: chr  "Premium" "Mainstream" "Budget" "Mainstream" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

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

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

print(life_stage)
```

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

A summary of `LIFESTAGE`

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

print(life_stage)
```

```
## # A tibble: 7 × 2
##   LIFESTAGE              number
##   <chr>                   <int>
## 1 RETIREES                14805
## 2 OLDER SINGLES/COUPLES   14609
## 3 YOUNG SINGLES/COUPLES   14441
## 4 OLDER FAMILIES           9780
## 5 YOUNG FAMILIES           9178
## 6 MIDAGE SINGLES/COUPLES   7275
## 7 NEW FAMILIES             2549
```

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

Display unique entries in `PREMIUM_CUSTOMER`

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

print(customer_category)
```

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

A summary of `PREMIUM_CUSTOMER`

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

print(customer_category)
```

```
## # A tibble: 3 × 2
##   PREMIUM_CUSTOMER number
##   <chr>             <int>
## 1 Mainstream        29245
## 2 Budget            24470
## 3 Premium           18922
```

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

Customer summary according to `LYLTY_CARD_NBR`

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

There's only transaction per card. We can confirm using

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

Merge customerData & transactionData

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

```
combined_data <- transactionData %>%
  left_join(customerData, by = "LYLTY_CARD_NBR")

# or combined_data <- merge(transactionData, customerData, all.x = TRUE)
```

1. Delete unnecessary columns

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

2. Standardize column names

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

3. Rename columns so they're easier to remember

```
names(combined_data)[1] <- "loyalty_card_number"    #from lylty_card_nbr
names(combined_data)[5] <- "product_number"      #from prod_nbr
names(combined_data)[6] <- "product_name"      #from prod_name
names(combined_data)[7] <- "product_quantity"      #from prod_qty
names(combined_data)[8] <- "total_sales"      #from tot_sales
names(combined_data)[9] <- "package_size"      #from size
names(combined_data)[13] <- "segment"      #from premium_customer
```

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

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

or

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

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

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

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

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

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

Data preparation is over now, time for data analysis on customer segments

Define some metrics

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

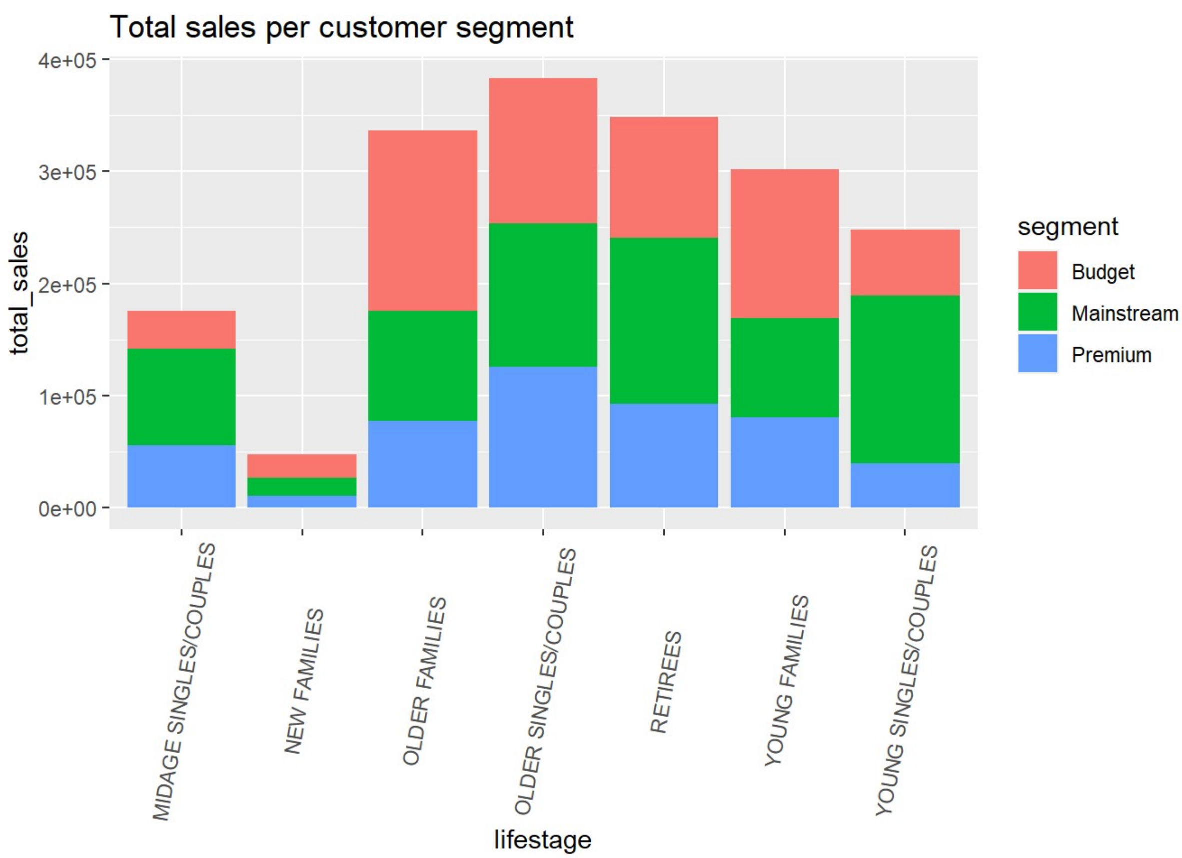
```
total_sales_lifestage <- combined_data %>%
  group_by(lifestage, segment) %>%
  summarise(total_sales = sum(total_sales)) %>%
  arrange(-total_sales)
```

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

```
# or total_sales_lifestage <- combined_data[, .(sales = sum(total_sales)), .(lifestage, segment)][order
(-sales)]
```

A visualization for the same

```
combined_data %>%
  ggplot(aes(x = lifestage, y = total_sales, fill = segment)) +
  geom_col() +
  labs(y = "total_sales", title = "Total sales per customer segment") +     #exact figures to show on y-
axis**
  theme(axis.text.x = element_text(angle = 80, vjust = 0.5))
```

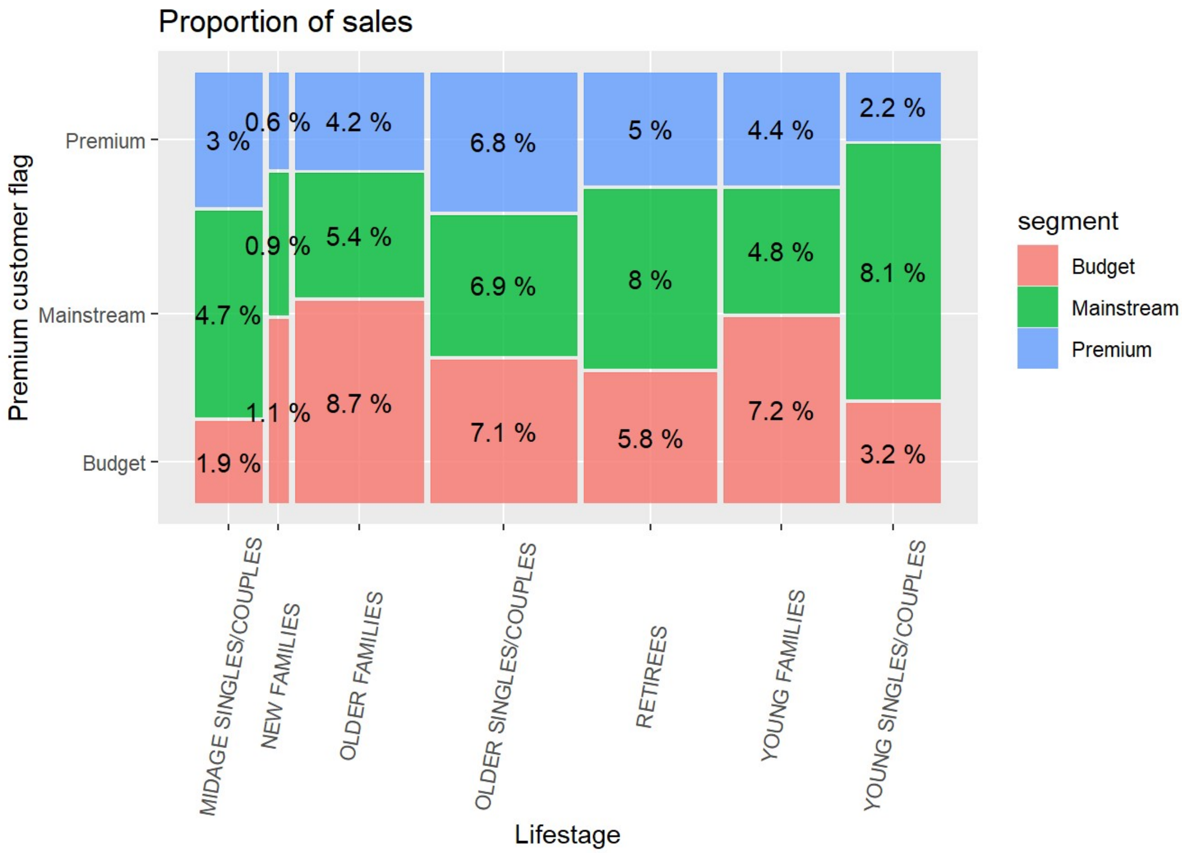


The customer category with the most sales are Budget - OLDER FAMILIES, Mainstream - YOUNG SINGLES/COUPLES and Mainstream - RETIREES Alternatively, plot using

```
p <- ggplot(data = total_sales_lifestage) +
  geom_mosaic(aes(weight = total_sales, x = product(segment, lifestage), fill = segment)) +
  labs(x = "Lifestage", y = "Premium customer flag", title = "Proportion of sales") +
  theme(axis.text.x = element_text(angle = 80, vjust = 0.5))

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

```
## Warning: `unite_()` was deprecated in tidyr 1.2.0.
## ℹ Please use `unite()` instead.
## ℹ The deprecated feature was likely used in the ggmosaic package.
##   Please report the issue at <]8;;https://github.com/haleyjeppson/ggmosaichttps://github.com/haleyjep
pson/ggmosaic]8;;>.
```



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

b. Number of customers by lifestage and segment

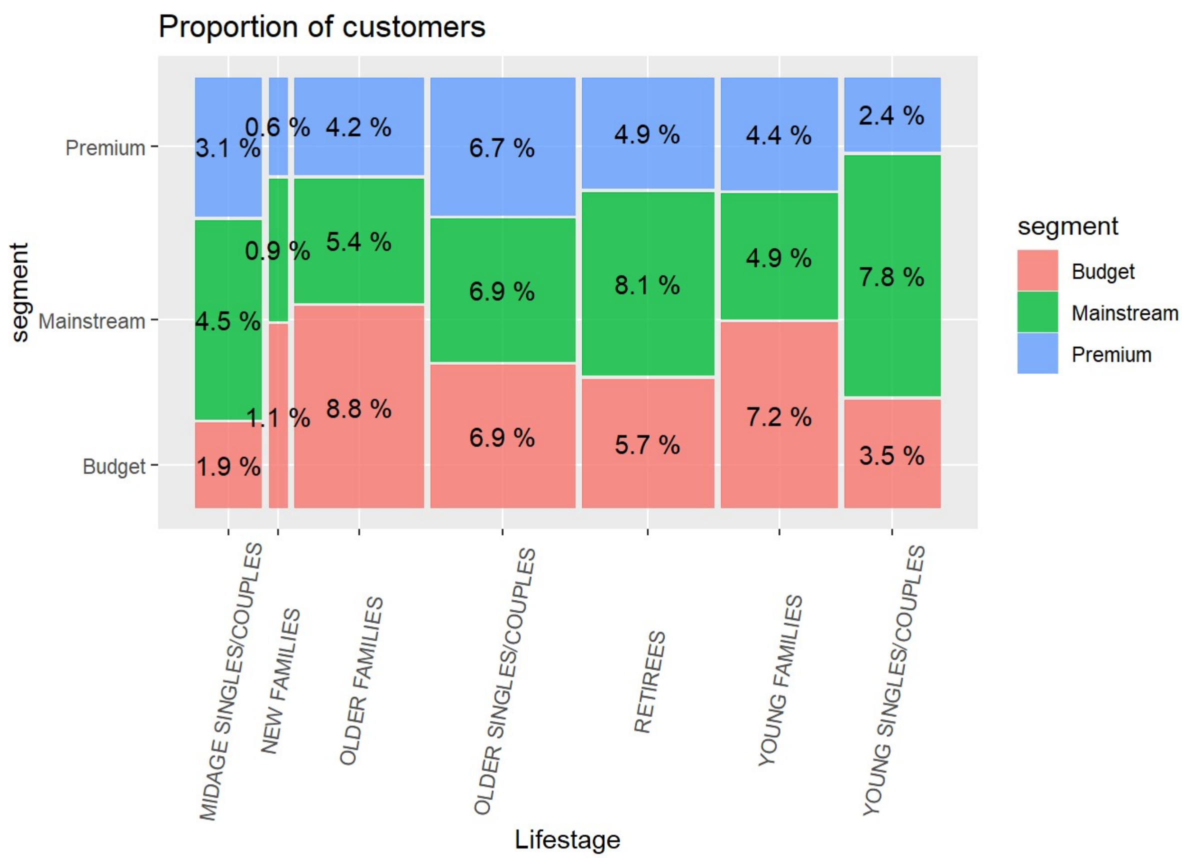
```
segment_tranxns <- combined_data %>%
  group_by(lifestage, segment)%>%
  summarise(customers = n()) %>%
  arrange(-customers)
```

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

Visualize using

```
q <- ggplot(data = segment_tranxns) +
  geom_mosaic(aes(weight = customers, x = product(segment,lifestage), fill = segment)) +
  labs(x = "Lifestage", y = "segment", title = "Proportion of customers") +
  theme(axis.text.x = element_text(angle = 80, vjust = 0.5))

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



There are more mainstream - YOUNG SINGLES/COUPLES & mainstream RETIREES who buy chips. This contributes to more sales in these 2 categories but doesn't seem to be the main driver of sales in Budget - OLDER FAMILIES. This implies it's not about having more customers who buy chips. If not, then let's consider quantity bought.

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

```
segment_qnty <- combined_data %>%
  group_by(lifestage, segment)%>%
  summarise(avg_qnty = sum(product_quantity)/uniqueN(loyalty_card_number)) %>%
  arrange(-avg_qnty)
```

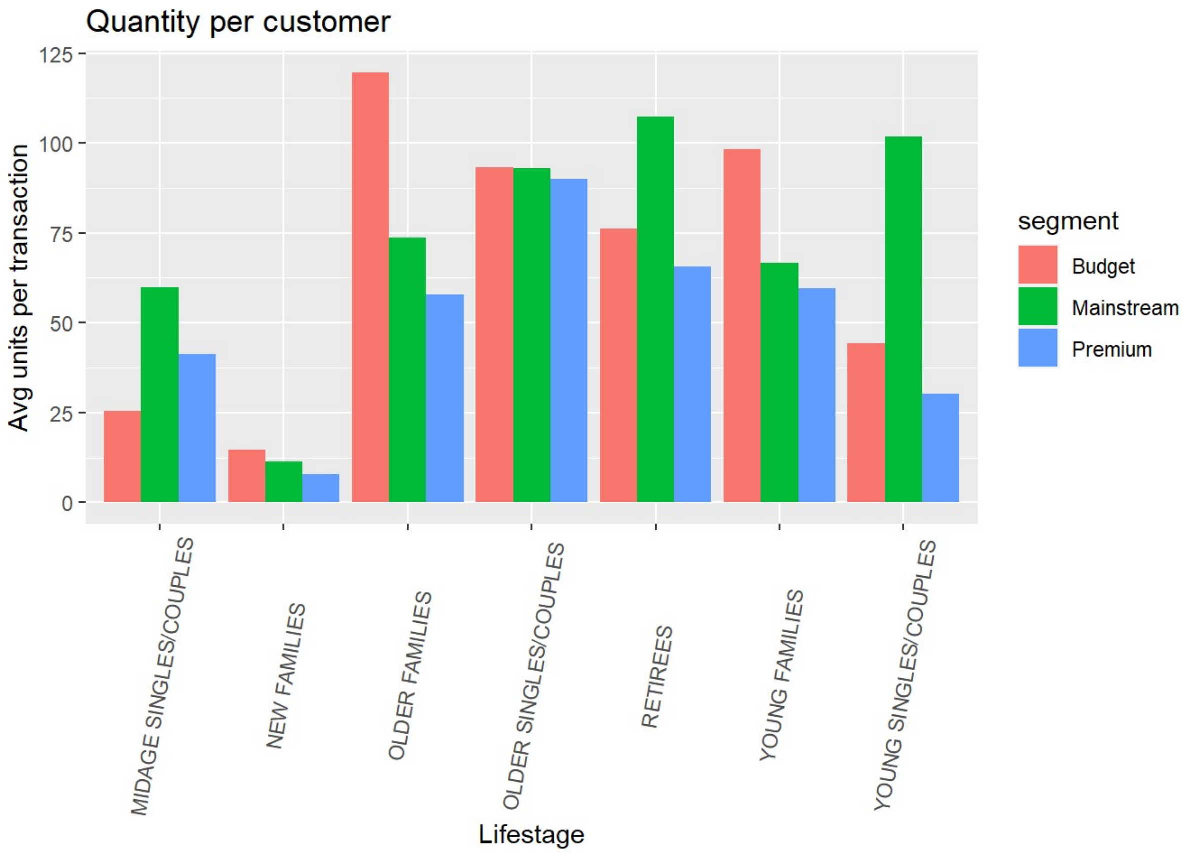
```
## `summarise()` has grouped output by 'lifestage'. You can override using the
## `.groups` argument.
```

or

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

Visualization of the same

```
ggplot(data = segment_qnty, aes(weight = avg_qnty, x = lifestage, fill = segment)) +
  geom_bar(position = position_dodge()) +
  labs(x = "Lifestage", y = "Avg units per transaction", title = "Quantity per customer") +
  theme(axis.text.x = element_text(angle = 80, vjust = 0.5))
```



Generally, Older & Young families buy more quantities on average

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

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

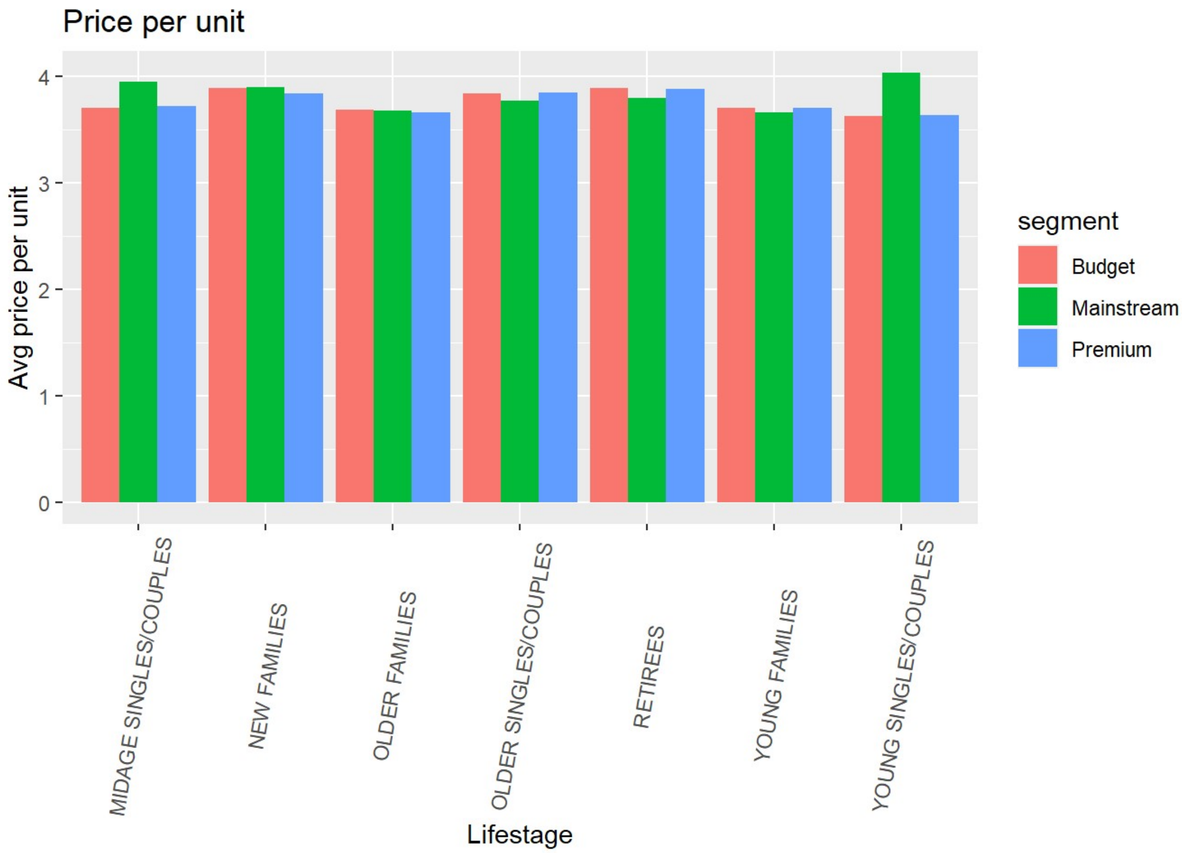
```
## `summarise()` has grouped output by 'lifestage'. You can override using the
## `.groups` argument.
```

or

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

Visualize

```
ggplot(data = segment_price_avg,
       aes(weight = avg_price, x = lifestage, fill = segment)) +
  geom_bar(position = position_dodge()) +
  labs(x = "Lifestage", y = "Avg price per unit", title = "Price per unit") +
  theme(axis.text.x = element_text(angle = 80, vjust = 0.5))
```



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

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

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

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

Statistical analysis

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

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

```
##
##  Welch Two Sample t-test
##
## data:  combined_data[lifestage %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES") & segment ==
"Mainstream", unit_cost] and combined_data[lifestage %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COU
PLES") & segment != "Mainstream", unit_cost]
## t = 39.922, df = 57088, p-value < 2.2e-16
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  0.3408767       Inf
## sample estimates:
## mean of x mean of y
##  3.999687  3.644162
```

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

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

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

a. Which brands do they tend to buy most?

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

We can see that kettle tops the list followed by doritos We can also use the affinity analysis or a-priori analysis to find out their most preferred brand

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

```
## [1] 37025
```

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

```
## [1] 449894
```

```
quantity_MYSC_by_brand <- mainstream_YSC[, .(MYSC = sum(product_quantity)/quantity_MYSC), by = brand]
quantity_other_by_brand <- other_segments[, .(other = sum(product_quantity)/quantity_others), by = bran
d]
```

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

```
##          brand        MYSC       other affinityToBrand
##  1:   tyrrells 0.030871033 0.024794729       1.2450643
##  2:   twisties 0.045185685 0.036553055       1.2361671
##  3:     kettle 0.193706955 0.159768746       1.2124208
##  4:   tostitos 0.044429440 0.036650856       1.2122347
##  5:   pringles 0.116839973 0.097118432       1.2030669
##  6:    doritos 0.128210668 0.109067914       1.1755122
##  7:       cobs 0.043673194 0.037684432       1.1589187
##  8:   infuzions 0.063281567 0.055070750       1.1490958
##  9:      thins 0.059068197 0.054995177       1.0740614
## 10:    grnwves 0.032005402 0.030098201       1.0633659
## 11:   cheezels 0.017582714 0.017995350       0.9770699
## 12:     smiths 0.097474679 0.126096369       0.7730173
## 13:     french 0.003862255 0.005556865       0.6950422
## 14:    cheetos 0.007859554 0.011644965       0.6749315
## 15:        rrd 0.047346388 0.070890032       0.6678850
## 16:    natural 0.019176232 0.029775903       0.6440185
## 17:        ccs 0.010938555 0.018235407       0.5998525
## 18:   sunbites 0.006212019 0.012140638       0.5116716
## 19: woolworths 0.029412559 0.059496237       0.4943600
## 20:     burger 0.002862930 0.006365944       0.4497260
```

We see that: - Mainstream YOUNG SINGLES/COUPLES are 24% more likely to purchase tyrrells Chips compared to the other segments - Mainstream YOUNG SINGLES/COUPLES are 56% less likely to purchase burger Chips compared to the other segments

b. Which package_size do they tend to buy most?

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

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

Conlcusion

In summary, we've noted the following: i) Larger proportions of sales are from the Budget - OLDER FAMILIES, Mainstream - YOUNG SINGLES/COUPLES, and Mainstream - RETIREES customers. ii) The high sales proportion by Mainstream - YOUNG SINGLES/COUPLES is due to there being more of them compared to other buyers. iii) Mainstream - YOUNG SINGLES/COUPLES are also likely to pay more per packet of chips compared to other customer categories. This suggests that there could be more impulsive buying among clients in this category. iv) Mainstream - YOUNG SINGLES/COUPLES are 24% more likely to buy Tyrrells Chips compared to the rest of the population.

Recommendation & next steps:

i. The category manager may strategically place the Tyrells Chips near shelves that are most frequented by Mainstream - YOUNG SINGLES/COUPLES. It's only packaged in 165g. They could add a few Kettle Chips packaged in 175g & 150g

ii. Quantium can help the Category Manager with recommendations of where these shelves are and further help them with measuring the impact of the changed placement.