Retail Analytics

Author : Sirawit N.

source : Quantium

```
#### Load required libraries
library(data.table)
library(ggplot2)
library(ggmosaic)
library(readr)
library(glue)
library(tidyverse)
library(readxl)
library(stringr)
library(stringi)
library(lubridate)
library(glue)
library(patchwork)
library(hrbrthemes)
library(RColorBrewer)
library(viridis)
```

Load required libraries Load the data to R

df_xlsx <- read_excel("filename.xlsx")

df_csv <- read_csv("filename.csv")

Exploratory data analysis EDA is the first step in any analysis to first understand the data. Let's take a look at each of the datasets provided.

Observe the transaction data

```
transactionData %>% head(10)
```

```
## # A tibble: 10 x 8
##     DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR PROD_NAME      PROD_~1 TOT_S~2
##    <dbl>     <dbl>          <dbl>  <dbl>    <dbl> <chr>           <dbl>   <dbl>
##  1 43390         1           1000      1        5 Natural Chip ~       2     6
##  2 43599         1           1307    348       66 CCs Nacho Che~       3     6.3
##  3 43605         1           1343    383       61 Smiths Crinkl~       2     2.9
##  4 43329         2           2373    974       69 Smiths Chip T~       5    15
##  5 43330         2           2426   1038      108 Kettle Tortil~       3    13.8
##  6 43604         4           4074   2982       57 Old El Paso S~       1     5.1
```

```
##  7 43601         4           4149   3333       16 Smiths Crinkl~       1     5.7
##  8 43601         4           4196   3539       24 Grain Waves  ~       1     3.6
##  9 43332         5           5026   4525       42 Doritos Corn ~       1     3.9
## 10 43330         7           7150   6900       52 Grain Waves S~       2     7.2
## # ... with abbreviated variable names 1: PROD_QTY, 2: TOT_SALES
```

```
colnames(transactionData)
```

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

```
str(transactionData)
```

```
## 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 Cheese    175g
##  $ 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 ...
```

Observe the purchase behaviour data

```
df_purc %>% head(10)
```

```
## # A tibble: 10 x 3
##    LYLTY_CARD_NBR LIFESTAGE              PREMIUM_CUSTOMER
##             <dbl> <chr>                  <chr>
##  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
##  7           1009 NEW FAMILIES           Premium
##  8           1010 YOUNG SINGLES/COUPLES  Mainstream
##  9           1011 OLDER SINGLES/COUPLES  Mainstream
## 10           1012 OLDER FAMILIES         Mainstream
```

```
colnames(df_purc)
```

```
## [1] "LYLTY_CARD_NBR"   "LIFESTAGE"        "PREMIUM_CUSTOMER"
```

```
str(df_purc)
```

```
## spec_tbl_df [72,637 x 3] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ LYLTY_CARD_NBR  : num [1:72637] 1000 1002 1003 1004 1005 ...
##  $ LIFESTAGE       : chr [1:72637] "YOUNG SINGLES/COUPLES" "YOUNG SINGLES/COUPLES" "YOUNG FAMILIES"
```

```
##  $ PREMIUM_CUSTOMER: chr [1:72637] "Premium" "Mainstream" "Budget" "Mainstream" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   LYLTY_CARD_NBR = col_double(),
##   ..   LIFESTAGE = col_character(),
##   ..   PREMIUM_CUSTOMER = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Expect to be numeric are in numeric form and date columns are in date format. #### Examine transaction data

```
#### Convert DATE column to a date format
#### CSV and Excel integer dates begin on 30 Dec 1899
transactionData$DATE <- as.Date(transactionData$DATE, origin = "1899-12-30")
transactionData %>% head(5)
```

```
## # A tibble: 5 x 8
##   DATE       STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR PROD_NAME  PROD_~1 TOT_S~2
##   <date>         <dbl>          <dbl>  <dbl>    <dbl> <chr>        <dbl>   <dbl>
## 1 2018-10-17         1           1000      1        5 Natural C~       2     6
## 2 2019-05-14         1           1307    348       66 CCs Nacho~       3     6.3
## 3 2019-05-20         1           1343    383       61 Smiths Cr~       2     2.9
## 4 2018-08-17         2           2373    974       69 Smiths Ch~       5    15
## 5 2018-08-18         2           2426   1038      108 Kettle To~       3    13.8
## # ... with abbreviated variable names 1: PROD_QTY, 2: TOT_SALES
```

Examine PROD_NAME text analysis by summarising the product name

```
transactionData %>% group_by(PROD_NAME) %>% summarise(COUNT = n()) %>% arrange(desc(COUNT))%>% head(5)
```

```
## # A tibble: 5 x 2
##   PROD_NAME                                COUNT
##   <chr>                                    <int>
## 1 Kettle Mozzarella   Basil & Pesto 175g    3304
## 2 Kettle Tortilla ChpsHny&Jlpno Chili 150g  3296
## 3 Cobs Popd Swt/Chlli &Sr/Cream Chips 110g  3269
## 4 Tyrrells Crisps     Ched & Chives 165g    3268
## 5 Cobs Popd Sea Salt  Chips 110g            3265
```

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

```
summary(transactionData$PROD_NAME)
```

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

```
tdf <- transactionData %>% group_by(PROD_NAME) %>% summarise(n =n()) %>% arrange(desc(n))

tdf <- tdf[1]
tdf%>% head(5)
```

```
## # A tibble: 5 x 1
##   PROD_NAME
##   <chr>
## 1 Kettle Mozzarella   Basil & Pesto 175g
## 2 Kettle Tortilla ChpsHny&Jlpno Chili 150g
## 3 Cobs Popd Swt/Chlli &Sr/Cream Chips 110g
## 4 Tyrrells Crisps     Ched & Chives 165g
## 5 Cobs Popd Sea Salt  Chips 110g
```

```
productWords <- strsplit(tdf$PROD_NAME, " ")
productWords_df<-data.table(productWords)
setnames(productWords_df, 'words')

productWords_df %>% head(5)
```

```
##                                      words
## 1:          Kettle,Mozzarella,,,Basil,&,...
## 2: Kettle,Tortilla,ChpsHny&Jlpno,Chili,150g
## 3: Cobs,Popd,Swt/Chlli,&Sr/Cream,Chips,110g
## 4:                  Tyrrells,Crisps,,,,,...
## 5:            Cobs,Popd,Sea,Salt,,Chips,...
```

```
#removing special characters
productWords_df$words <-  str_replace_all(productWords_df$words,"[[:punct:]]"," ")
```

```
#removing digit
productWords_df$words <- str_replace_all(productWords_df$words,"[0-9]"," ")
#### Removing special characters
productWords_df$words <- str_replace_all(productWords_df$words,"[gG]"," ")
```

```
wordsplit <- strsplit(productWords_df$words," ")
# check data type
typeof(wordsplit)
```

```
## [1] "list"
```

```
### since the variable is list we have to unlist to get the result as we want
```

```
word_n <- as.data.frame(table(unlist(wordsplit)))
#### sorting them by this frequency in order of highest to lowest frequency
word_n <- word_n %>% rename(Word = Var1, n = Freq ) %>% arrange(desc(n))
word_n %>% head(10)
```

```
##       Word    n
## 1          3200
## 2        c  114
## 3    Chips   21
## 4   Smiths   16
## 5  Crinkle   14
## 6      Cut   14
## 7   Kettle   13
## 8   Cheese   12
## 9     Salt   12
## 10     Ori   11
```

use regular expression with grepl to filter the name There are salsa products in the dataset but we are only interested in the chips category, so let's remove these.

```
#### Remove salsa products
t1 <- transactionData %>% mutate(salsa = grepl("salsa", transactionData$PROD_NAME, ignore.case =T))
colnames(t1)
```

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

```
#filter salsa
t1 <- t1 %>% filter(salsa == 'FALSE')
summary(t1)
```

```
##       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
##    salsa
##  Mode :logical
##  FALSE:246742
##
##
##
##
```

```
# now drop the salsa column to original dataframe
t1 <- t1[1:8]
colnames(t1)
```

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

```
#write a function to check missing value in each column
check_na <- function(col){sum(is.na(col))}
# check na through column
count_na <- apply(t1, MARGIN = 2, check_na)
count_na
```

Summarise the data to check for nulls and possible outliers

```
##            DATE      STORE_NBR LYLTY_CARD_NBR         TXN_ID       PROD_NBR
##               0              0              0              0              0
##      PROD_NAME       PROD_QTY      TOT_SALES
##               0              0              0
```

```
# There is no missing value in the dataframe now check outlier
# using summary to observe the outlier
summary(t1)
```

```
##       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
```

Notice that from the summary the maximun value in the PROD_QTY column is 200 where as its Mean and 3rdQu is 1.9 and 2 respectively so this might suspect to be the outliers.

```
#chect the outlier
t1 %>% select(DATE,LYLTY_CARD_NBR,PROD_NAME,PROD_QTY,TOT_SALES) %>% filter(PROD_QTY >= 50)
```

```
## # A tibble: 2 x 5
##   DATE       LYLTY_CARD_NBR PROD_NAME                          PROD_QTY TOT_SALES
##   <date>              <dbl> <chr>                                 <dbl>     <dbl>
## 1 2018-08-19         226000 Dorito Corn Chp     Supreme 380g      200       650
## 2 2019-05-20         226000 Dorito Corn Chp     Supreme 380g      200       650
```

Notice that the transaction form the customer loyalty card number 226000 had purchased "Dorito Corn Chp Supreme" with the quantity of 200 in one transaction on Date 2018-08-19 and 2019-05-20 on the same product and quantity.

```
#### Let's see if the customer has had other transactions
t1 %>% filter(LYLTY_CARD_NBR == 226000)
```

```
## # A tibble: 2 x 8
##   DATE       STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR PROD_NAME  PROD_~1 TOT_S~2
##   <date>         <dbl>          <dbl>  <dbl>    <dbl> <chr>        <dbl>   <dbl>
## 1 2018-08-19       226         226000 226201        4 Dorito Co~     200     650
## 2 2019-05-20       226         226000 226210        4 Dorito Co~     200     650
## # ... with abbreviated variable names 1: PROD_QTY, 2: TOT_SALES
```

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

```
## remove customer with loyalty card number 226000 from the dataframe
#### Filter out the customer based on the loyalty card number
t1 <- t1 %>% filter(LYLTY_CARD_NBR != 226000)
```

Re-examine transaction data

Let's look at the number of transaction lines over time to see if there are any obvious data issues such as missing data.

```
#### Count the number of transactions by date
t1_cdate <- t1 %>% group_by(DATE) %>% summarise(n = n())
t1_cdate %>% head(5)
```

```
## # A tibble: 5 x 2
##   DATE           n
##   <date>     <int>
## 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
```

```
t1 %>% group_by(DATE) %>% summarise(n = n()) %>% head(5)
```

```
## # A tibble: 5 x 2
##   DATE           n
##   <date>     <int>
## 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
```

```
n_distinct(t1$DATE)
```

```
## [1] 364
```

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.

```
# find the missing date
date_range <- seq(min(t1d$DATE), max(t1d$DATE), by =1)
date_range[!date_range %in% t1d$DATE]
```

```
## [1] "2018-12-25"
```

```
#[1] "2018-12-25"
t1 %>% filter(DATE == '2018-12-25')
```

```
## # A tibble: 0 x 8
## # ... with 8 variables: DATE <date>, STORE_NBR <dbl>, LYLTY_CARD_NBR <dbl>,
## #   TXN_ID <dbl>, PROD_NBR <dbl>, PROD_NAME <chr>, PROD_QTY <dbl>,
## #   TOT_SALES <dbl>
```

- create a column of dates that includes every day from 1 Jul 2018 to 30 Jun 2019, and join it on to the data to fill in the missing day.

```
#### Create a sequence of dates and join this the count of transactions by date

date_df <-data.frame(DATE = seq(as.Date("2018-07-01"), as.Date("2019-06-30"), by = "day"))
## create new data frame contain missing date
t1_misdt <- date_df %>% left_join(t1)
```

```
## Joining, by = "DATE"
```

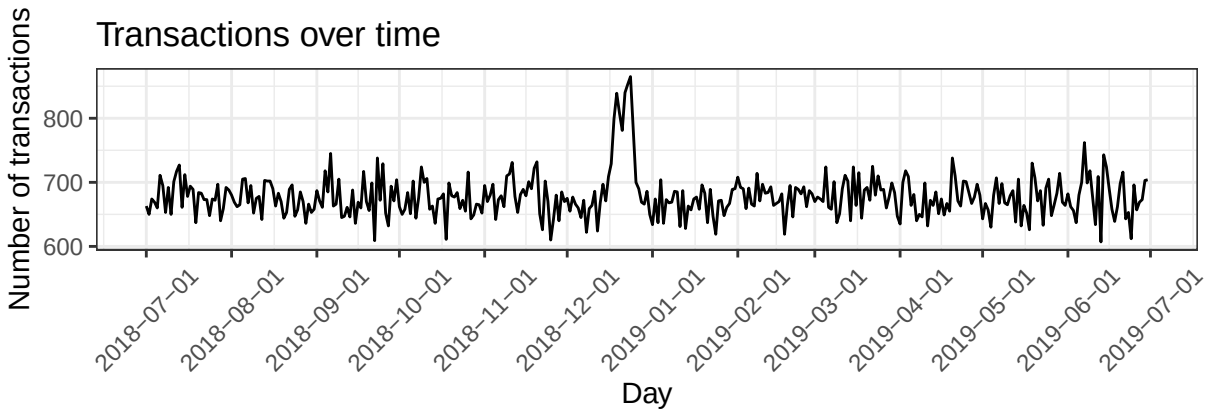
```
## now the missing date is appear in the dataa frame but still contain NA
transactions_by_day <- t1_misdt %>% group_by(DATE) %>% summarise(n = n()) %>% arrange(DATE)
transactions_by_day %>% head(5)
```

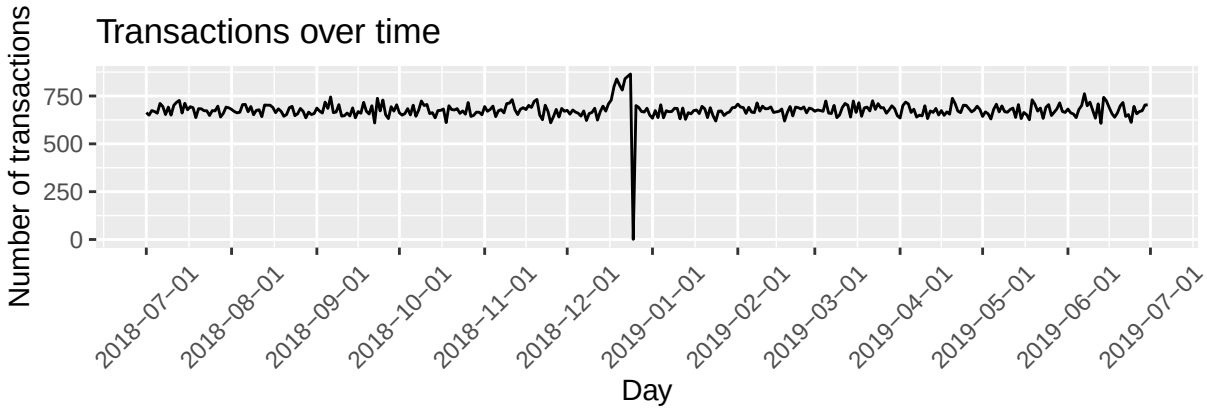
```
## # A tibble: 5 x 2
##   DATE           n
##   <date>     <int>
## 1 2018-07-01   663
## 2 2018-07-02   650
## 3 2018-07-03   674
## 4 2018-07-04   669
## 5 2018-07-05   660
```

Setting plot themes to format graphs theme_set(theme_bw()) theme_update(plot.title = element_text(hjust = 0.5))

```
## plot include missing date
p2<-ggplot(transactions_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_set(theme_bw())+theme(axis.text.x = element_text(angle = 45, vjust = 0.5))
```

```
#plot without miissig date
p1<-ggplot(t1_cdate, 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_set(theme_bw())+theme(axis.text.x = element_text(angle = 45, vjust = 0.5))
```





We can see that there is an increase in purchases in December and a break in late December. Let's zoom in on this.

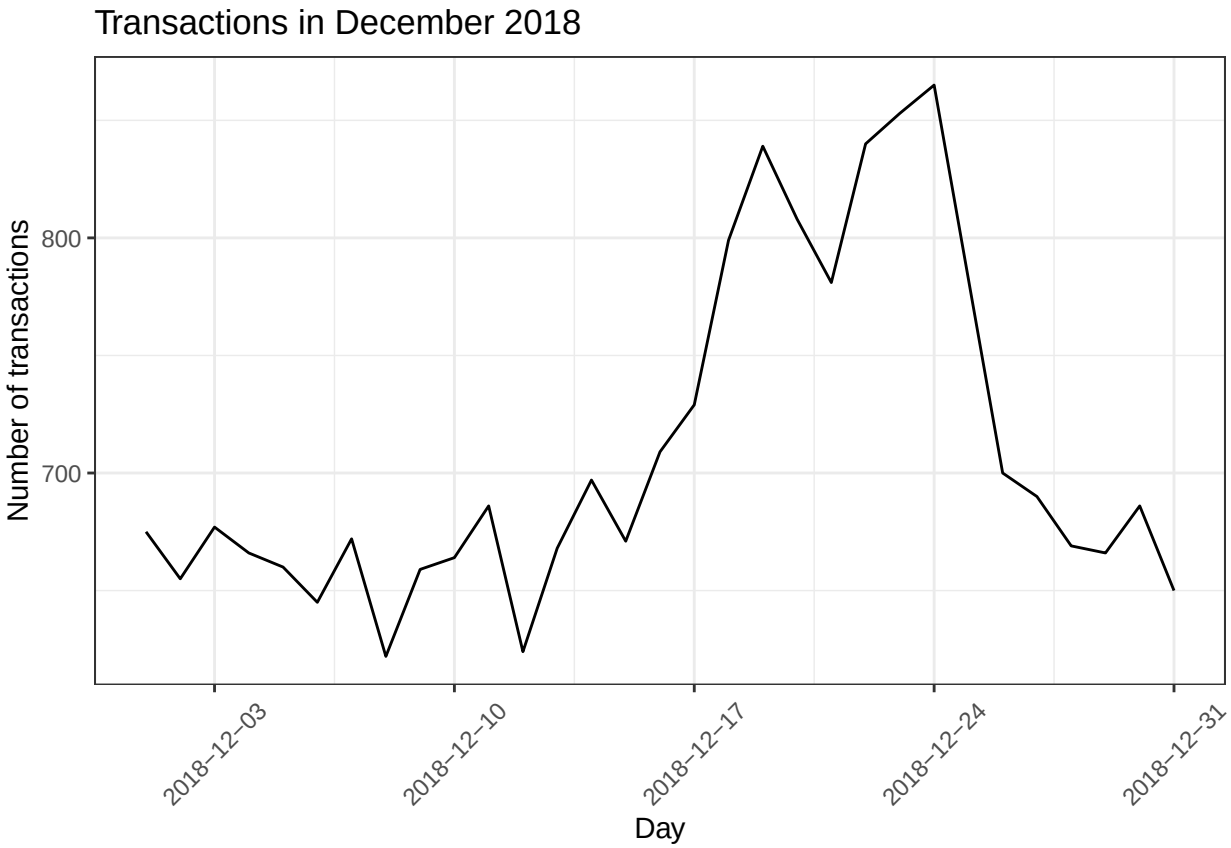
```
dec_transac <- t1_cdate %>%
  filter(between(t1_cdate$DATE, as.Date('2018-12-01'), as.Date('2018-12-31')))

dec_transac %>% head(10)
```

Filter to December and look at individual days

```
## # A tibble: 10 x 3
##    DATE           n month
##    <date>     <int> <dbl>
##  1 2018-12-01   675    12
##  2 2018-12-02   655    12
##  3 2018-12-03   677    12
##  4 2018-12-04   666    12
##  5 2018-12-05   660    12
##  6 2018-12-06   645    12
##  7 2018-12-07   672    12
##  8 2018-12-08   622    12
##  9 2018-12-09   659    12
## 10 2018-12-10   664    12
```

recreate the chart above zoomed in to the relevant dates.

```
ggplot(dec_transac, aes(x = DATE, y = n)) +
 geom_line() +
 labs(x="Day", y="Number of transactions", title="Transactions in December 2018") +
 scale_x_date(breaks = "1 week") +
 theme_set(theme_bw())+
 theme(axis.text.x = element_text(angle = 45, vjust = 0.5))
```

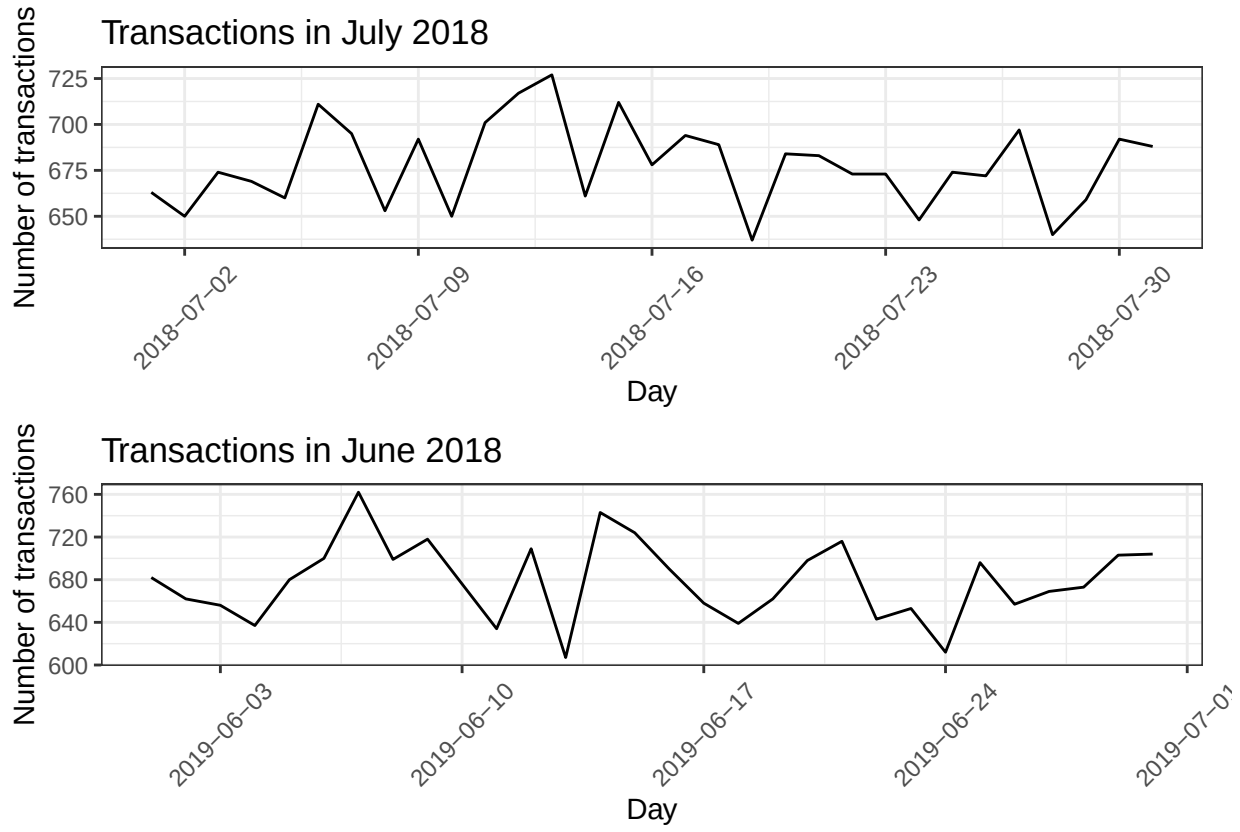


We can see that the increase in sales occurs in the lead-up to Christmas and that there are zero sales on Christmas day itself. This is due to shops being closed on Christmas day.

```
jun_transac <- t1_cdate %>% filter(month == 6)

jul_transac <- t1_cdate %>% filter(month == 7)

ju1<- ggplot(jun_transac, aes(x = DATE, y = n)) +
     geom_line() +
     labs(x="Day", y="Number of transactions", title="Transactions in June 2018") +
     scale_x_date(breaks = "1 week") +
     theme_set(theme_bw())+
     theme_update(plot.title = element_text(hjust = 0.5))+
     theme(axis.text.x = element_text(angle = 45, vjust = 0.5))
```

```
jul1 <-  ggplot(jul_transac, aes(x = DATE, y = n)) +
     geom_line() +
     labs(x="Day", y="Number of transactions", title="Transactions in July 2018") +
     scale_x_date(breaks = "1 week") +
     theme_set(theme_bw())+
     theme_update(plot.title = element_text(hjust = 0.5))+
     theme(axis.text.x = element_text(angle = 45, vjust = 0.5))
```

Lets compare the transection on Jul vs Jun Transaction July 2018 vs June 2018



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.

Packaging size

```
t_pk <- transactionData %>% mutate(PACK_SIZE = parse_number(transactionData$PROD_NAME))
#### Let's check if the pack sizes look sensible
summary(t_pk$PACK_SIZE)
```

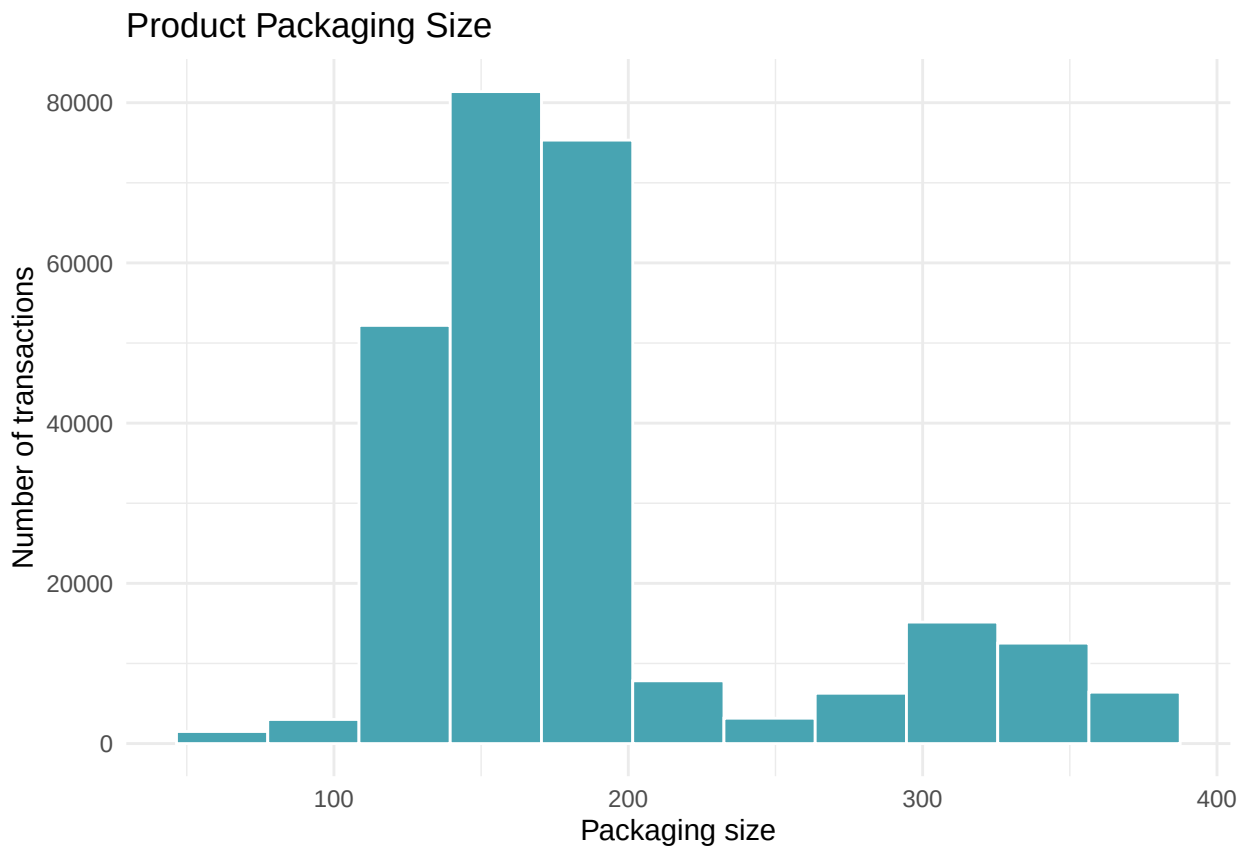
We can work this out by taking the digits that are in PROD_NAME

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    70.0   150.0   170.0   182.4   175.0   380.0
```

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

Let's plot a histogram of PACK_SIZE since we know that it is a categorical variable and not a continuous variable even though it is numeric.

```
 #histogram showing the number of transactions by pack size
ggplot(t_pk,aes(PACK_SIZE))+geom_histogram(bins = 11,fill="#48a4b2",color = 'white')+
  labs(x="Packaging size", y="Number of transactions", title="Product Packaging Size")+
  theme_minimal()
```



From the histogram the packaging sizes created look reasonable.

Now to create brands, we can use the first word in PROD_NAME to work out the brand name. Create a column which contains the brand of the product, by extracting it from the product name.

```
transactionData$Brand <- gsub(pattern="([A-Z]+).*", "\\1",
                              transactionData$PROD_NAME,
                              ignore.case = TRUE)
transactionData %>% group_by(Brand) %>% summarise(n = n()) %>% head(5)
```

```
## # A tibble: 5 x 2
##   Brand        n
##   <chr>    <int>
## 1 Burger    1564
## 2 CCs       4551
## 3 Cheetos   2927
## 4 Cheezels  4603
## 5 Cobs      9693
```

Some of the brand names look like they are of the same brands - such as RED and RRD, which are both Red Rock Deli chips. Let's combine these together.

```
#### Clean brand names
trD <- transactionData
trD$Brand[trD$Brand == "RRD"] <-'RED'
```

other additional brand adjustments that required to change brand's name.

```
trD$Brand[trD$Brand == "WW"] <-'Woolworths'
trD$Brand[trD$Brand == "Dorito"] <-'Doritos'
trD$Brand[trD$Brand == "Infzns"] <-'Infuzions'
trD$Brand[trD$Brand == "Smith"] <-'Smiths'
trD$Brand[trD$Brand == "Snbts"] <-'Sunbites'
trD$Brand[trD$Brand == "GrnWves"] <-'Grain'
trD$Brand[trD$Brand == "Red"] <-'RED'

trD %>%  group_by(Brand) %>% summarise(n = n()) %>% head(5)
```

```
## # A tibble: 5 x 2
##   Brand        n
##   <chr>    <int>
## 1 Burger    1564
## 2 CCs       4551
## 3 Cheetos   2927
## 4 Cheezels  4603
## 5 Cobs      9693
```

Examining customer data

let's have a look at the customer dataset.

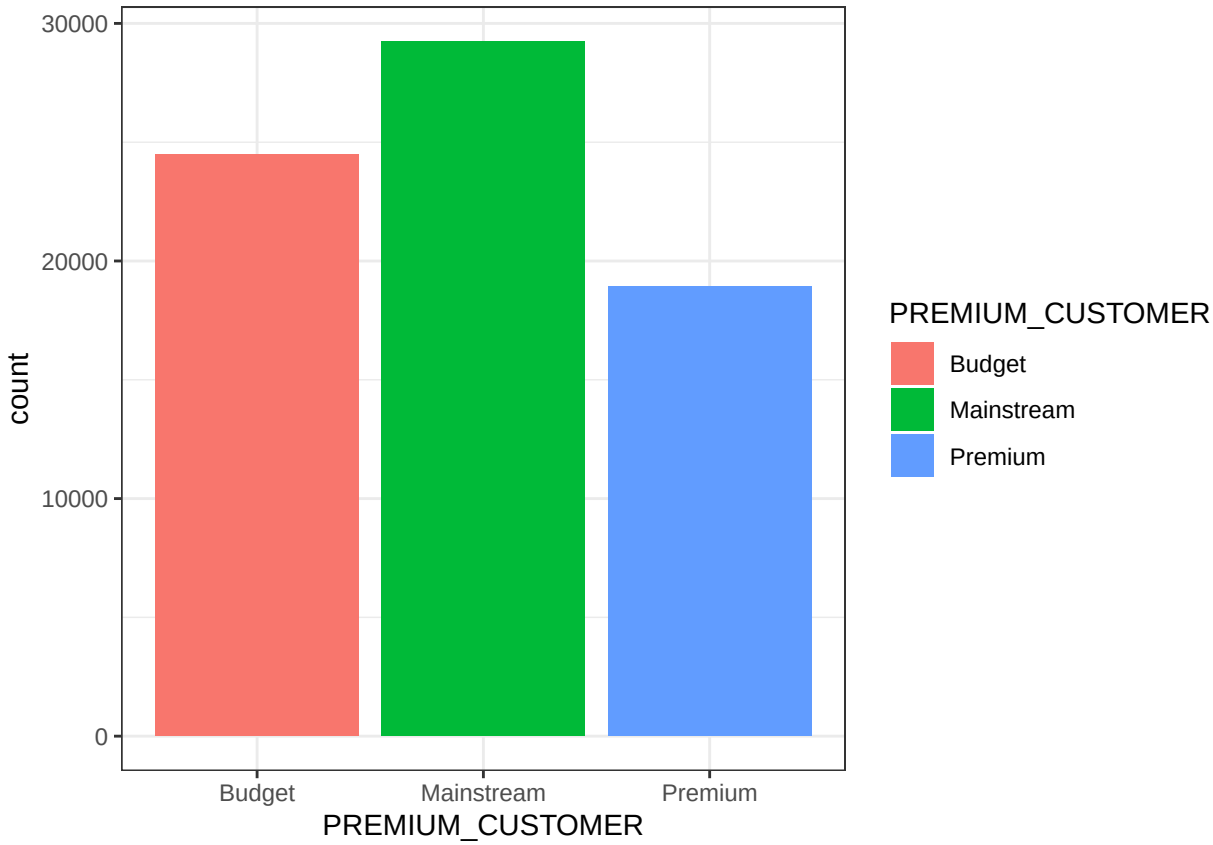
```
#### Examining customer data, summaries of the dataset

colnames(df_purc)
```

```
## [1] "LYLTY_CARD_NBR"   "LIFESTAGE"        "PREMIUM_CUSTOMER"
```

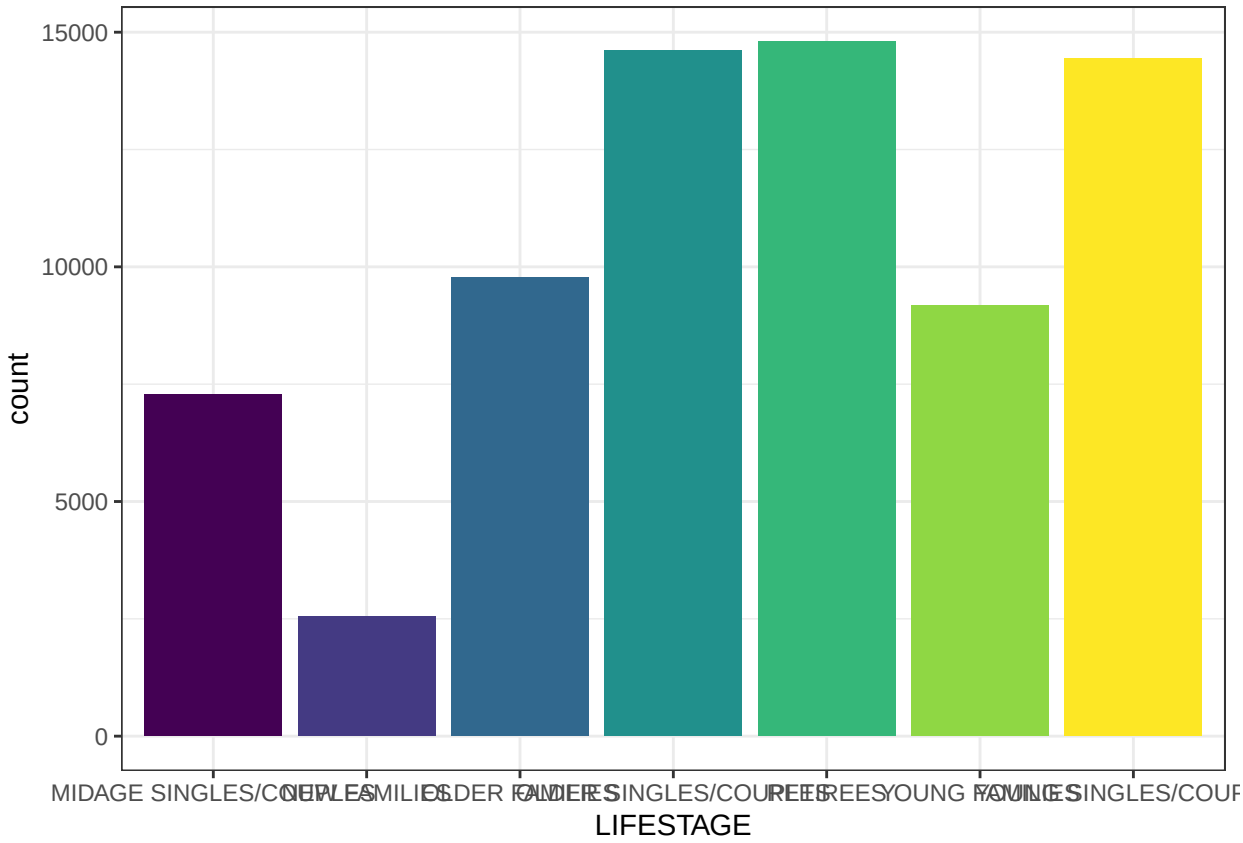
```
df_purc %>% group_by(PREMIUM_CUSTOMER) %>% summarise(n = n()) %>% arrange(desc(n))
```

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

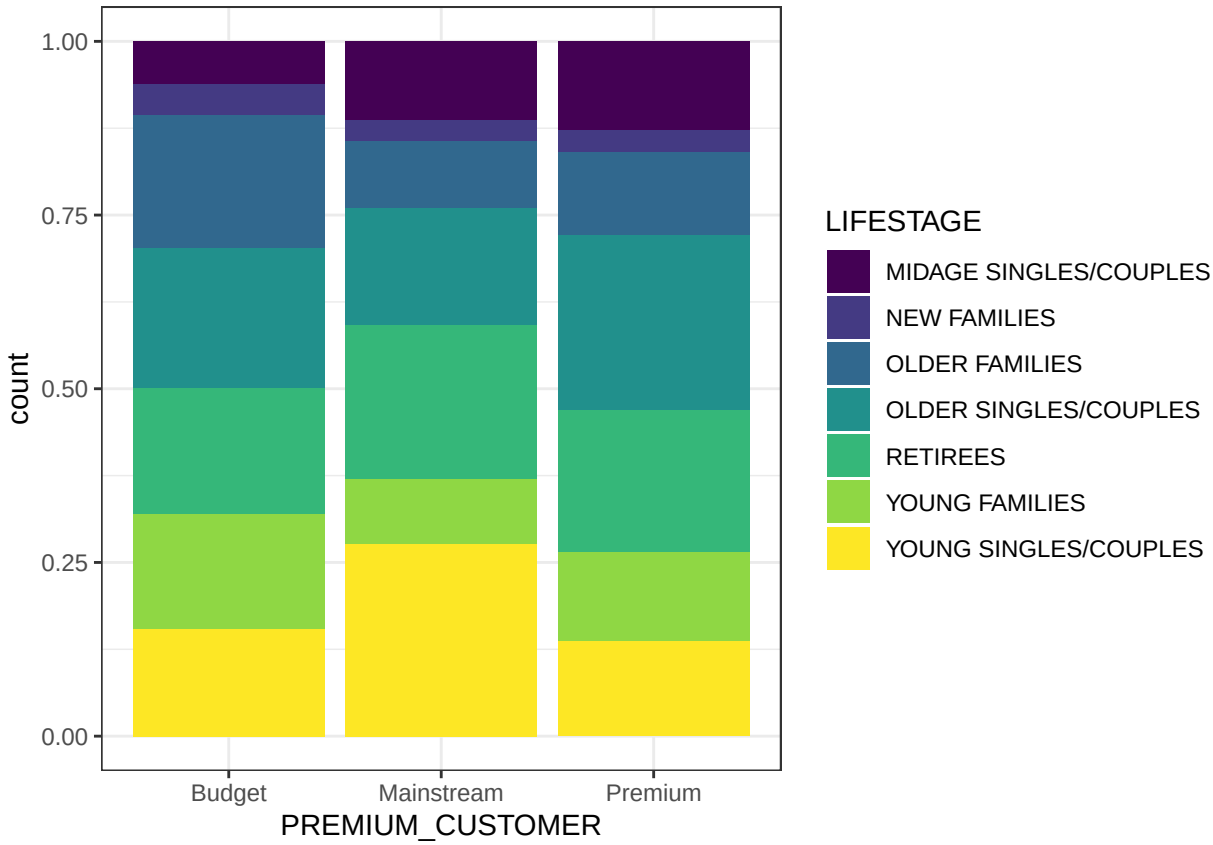
```
ggplot(df_purc, aes(PREMIUM_CUSTOMER, fill = PREMIUM_CUSTOMER))+geom_bar()
```



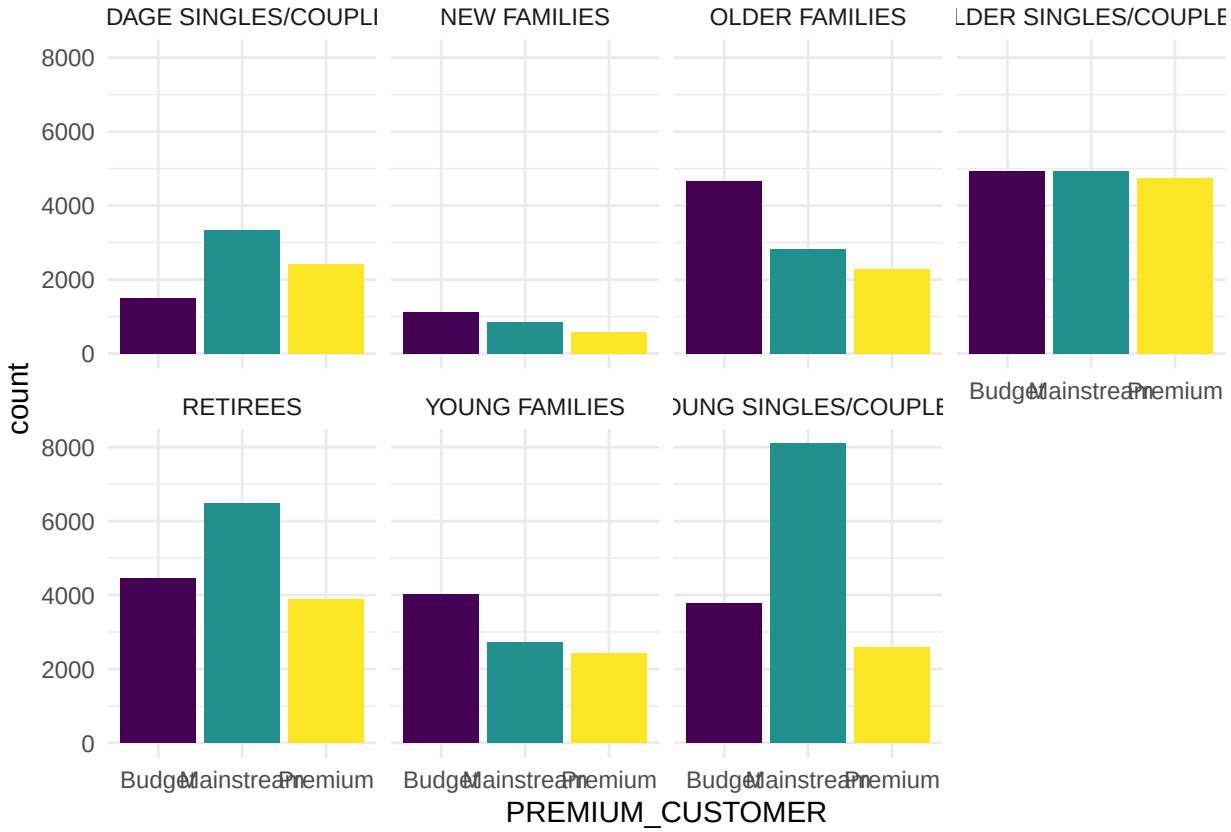

```
df_purc %>% group_by(LIFESTAGE) %>% ggplot(aes(LIFESTAGE, fill =LIFESTAGE))+
  geom_bar()+scale_fill_viridis(discrete=TRUE, guide=FALSE, option="D")
```




```
ggplot(df_purc, aes(PREMIUM_CUSTOMER, fill = LIFESTAGE))+
  geom_bar(position = 'fill')+scale_fill_viridis(discrete=TRUE, option="D")
```




```
ggplot(df_purc, aes(PREMIUM_CUSTOMER, fill = PREMIUM_CUSTOMER))+
  geom_bar()+facet_wrap(~LIFESTAGE, ncol = 4)+theme_minimal()+
  scale_fill_viridis(discrete=TRUE, guide=FALSE, option="D")
```




```
#### Merge transaction data to customer data
data <- merge(transactionData, df_purc, all.x = TRUE)
data %>% head(5)
```

```
##   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
##                                PROD_NAME PROD_QTY TOT_SALES   Brand
## 1 Natural Chip        Compny SeaSalt175g        2       6.0 Natural
## 2  Red Rock Deli Chikn&Garlic Aioli 150g        1       2.7     Red
## 3  Grain Waves Sour    Cream&Chives 210G        1       3.6   Grain
## 4 Natural ChipCo      Hony Soy Chckn175g        1       3.0 Natural
## 5         WW Original Stacked Chips 160g        1       1.9      WW
##               LIFESTAGE PREMIUM_CUSTOMER
## 1 YOUNG SINGLES/COUPLES          Premium
## 2 YOUNG SINGLES/COUPLES       Mainstream
## 3        YOUNG FAMILIES           Budget
## 4        YOUNG FAMILIES           Budget
## 5 OLDER SINGLES/COUPLES       Mainstream
```

As the number of rows in `data` is the same as that of `transactionData`, we can be sure that no duplicates were created. This is because we created `data` by setting `all.x = TRUE` (in other words, a left join) which

means take all the rows in `transactionData` and find rows with matching values in shared columns and then joining the details in these rows to the `x` or the first mentioned table.

Let's also check if some customers were not matched on by checking for nulls.

```
# See if any transactions did not have a matched customer.
apply(data, MARGIN = 2, check_na)
```

```
##   LYLTY_CARD_NBR             DATE        STORE_NBR           TXN_ID
##                0                0                0                0
##         PROD_NBR        PROD_NAME         PROD_QTY        TOT_SALES
##                0                0                0                0
##            Brand        LIFESTAGE PREMIUM_CUSTOMER
##                0                0                0
```

Great, there are no nulls! So all our customers in the transaction data has been accounted for in the customer dataset.

write this dataset into a csv file

```
write.csv(data,"QVI_data_clean.csv")
```

Data analysis on 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

```
data %>% select(LYLTY_CARD_NBR, PROD_QTY,PROD_NAME, TOT_SALES, Brand, LIFESTAGE,PREMIUM_CUSTOMER) %>% h
```

```
##   LYLTY_CARD_NBR PROD_QTY                               PROD_NAME TOT_SALES
## 1           1000        2 Natural Chip        Compny SeaSalt175g       6.0
## 2           1002        1  Red Rock Deli Chikn&Garlic Aioli 150g       2.7
## 3           1003        1  Grain Waves Sour    Cream&Chives 210G       3.6
## 4           1003        1 Natural ChipCo      Hony Soy Chckn175g       3.0
## 5           1004        1         WW Original Stacked Chips 160g       1.9
##     Brand             LIFESTAGE PREMIUM_CUSTOMER
## 1 Natural YOUNG SINGLES/COUPLES          Premium
## 2     Red YOUNG SINGLES/COUPLES       Mainstream
## 3   Grain        YOUNG FAMILIES           Budget
## 4 Natural        YOUNG FAMILIES           Budget
## 5      WW OLDER SINGLES/COUPLES       Mainstream
```

```
chip <- data %>% select(LYLTY_CARD_NBR, PROD_QTY,PROD_NAME,
                        TOT_SALES, Brand, LIFESTAGE,PREMIUM_CUSTOMER) %>%
                mutate(chips = grepl("chips*", data$PROD_NAME, ignore.case =TRUE)) %>%
                filter(chips == TRUE)
```

```
chip %>% select(card_no = LYLTY_CARD_NBR, TOT_SALES, l_sta = LIFESTAGE,
                cus_sta = PREMIUM_CUSTOMER) %>%
        group_by(card_no) %>%
        mutate(chip_sales = sum(TOT_SALES)) %>%
        summarise(l_sta, cus_sta, n_transc = n() ,chip_sales,
                  avg_bill = round(chip_sales/n_transc,digits = 1)) %>%
```

```
## # A tibble: 5 x 6
## # Groups:   card_no [5]
##   card_no l_sta          cus_sta n_transc chip_sales avg_bill
##     <dbl> <chr>          <chr>      <int>      <dbl>    <dbl>
## 1   69154 OLDER FAMILIES Budget         9       79.6      8.8
## 2   32060 YOUNG FAMILIES Budget         9       68.6      7.6
## 3  212185 OLDER FAMILIES Budget        10       67.8      6.8
## 4   72150 YOUNG FAMILIES Budget         7       61.4      8.8
## 5  157091 OLDER FAMILIES Premium        7       59.2      8.5
```

- Who spends the most on chips (total sales), describing customers by lifestage and how premium their general purchasing behaviour is:

The top 5 spenders on chip products are shown in the table above, and the most spender is a customer card no. 69154 from the older families section which had a total spending amount of 79.6 USD and 9 transactions with an average bill for chip products of 8.8 USD throughout the period we can summarise that budget customer favor purchasing chip products and only 1 customer that is a premium customer out of 5 highest spenders and all of them are in the families lifestage which 3 are older families and 2 young families.

```
data %>% select(card_no = LYLTY_CARD_NBR, TOT_SALES, l_sta = LIFESTAGE,
                cus_sta = PREMIUM_CUSTOMER) %>%
        group_by(card_no) %>%
        mutate(Total_sales = sum(TOT_SALES)) %>%
        summarise(l_sta, cus_sta, n_transc = n() ,Total_sales,
                  avg_bill = round(Total_sales/n_transc,digits = 1)) %>%
```

```
## # A tibble: 5 x 6
## # Groups:   card_no [5]
##   card_no l_sta                cus_sta    n_transc Total_sales avg_bill
##     <dbl> <chr>                <chr>         <int>       <dbl>    <dbl>
## 1  230078 OLDER FAMILIES       Budget           17        139.      8.2
## 2   63197 OLDER FAMILIES       Budget           15        133.      8.9
## 3  259009 OLDER SINGLES/COUPLES Mainstream       15        127.      8.5
## 4  162039 OLDER FAMILIES       Mainstream       18        127.      7
## 5   58361 YOUNG FAMILIES       Budget           14        125.      8.9
```

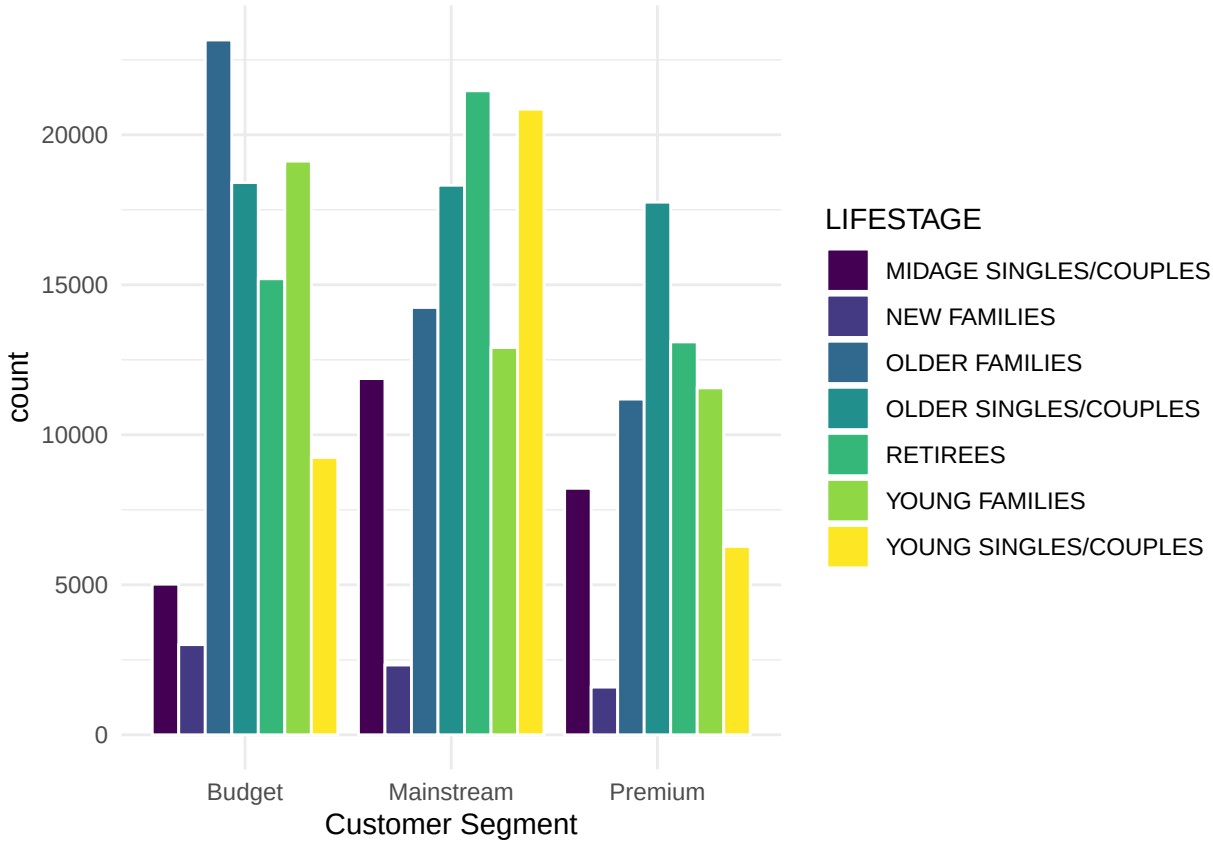
The most spenders from all product categories is in budget with older families customer section with the total bill of 138.6 USD and the number of transaction throughout the year is 17 with the average per bill of 8.2 USD.

- How many customers are in each segment

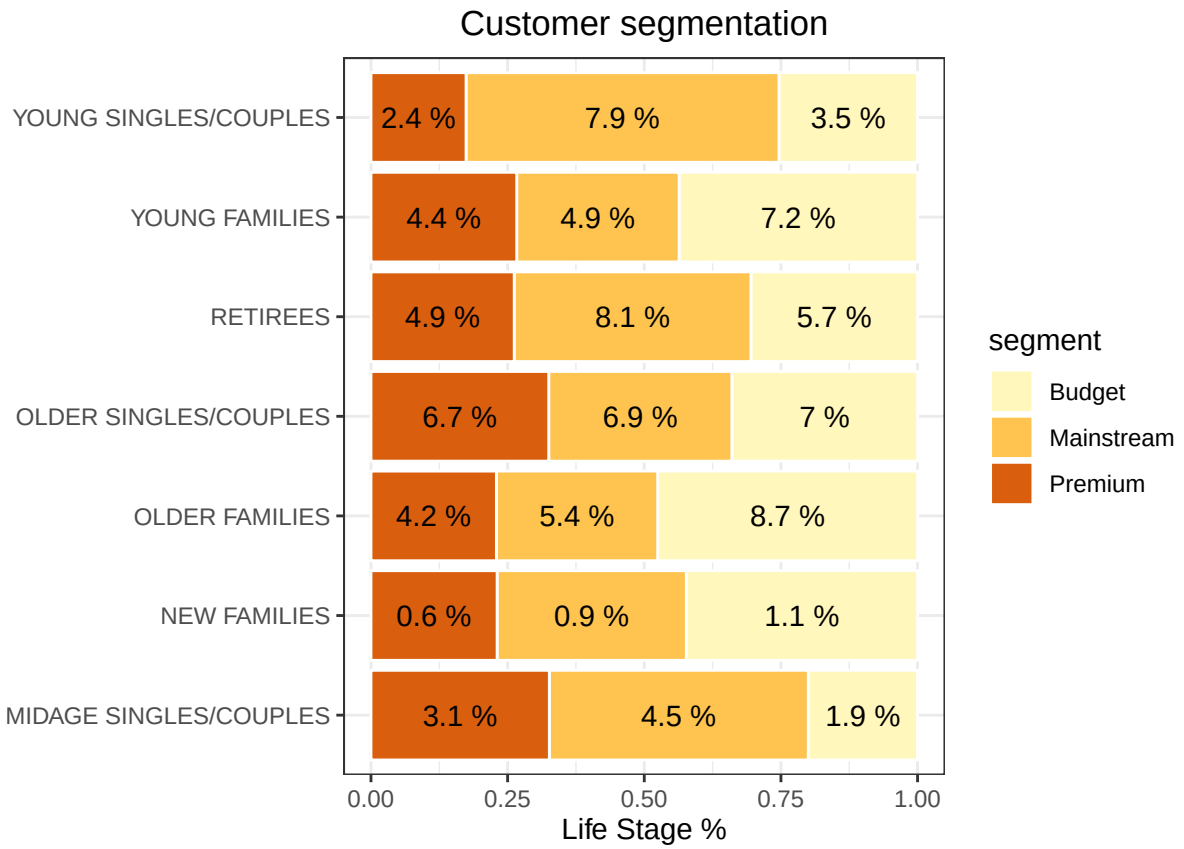
```
data %>% group_by(LIFESTAGE,segment = PREMIUM_CUSTOMER) %>%
         summarise(n = n()) %>% arrange(segment, LIFESTAGE,desc(n))%>% head(5)
```

```
## # A tibble: 5 x 3
## # Groups:   LIFESTAGE [5]
##   LIFESTAGE              segment     n
##   <chr>                  <chr>   <int>
## 1 MIDAGE SINGLES/COUPLES Budget   5020
## 2 NEW FAMILIES           Budget   3005
## 3 OLDER FAMILIES         Budget  23160
## 4 OLDER SINGLES/COUPLES  Budget  18407
## 5 RETIREES               Budget  15201
```

```
ggplot(data, aes(PREMIUM_CUSTOMER, fill = LIFESTAGE))+
  geom_bar(position = 'dodge',color ='white')+ theme_minimal() +
  scale_fill_viridis(discrete=TRUE, option="D")+xlab('Customer Segment')
```




```
pctt %>% ggplot(aes(y=LIFESTAGE, x=pctt[[5]], fill = segment))+
  geom_col(position="fill", stat="identity", color = 'white')+
  theme(axis.text.x = element_text(angle = 0, vjust = 0.7))+
  labs(y='', x ='Life Stage %', title  = "Customer segmentation")+
  geom_text(aes(label=paste(pctt[[5]],'%')),position = position_fill(vjust = 0.5))+
  scale_fill_brewer(palette = "YlOrBr")
```



- How many chips are bought per customer by segment

```
chip %>% group_by(LIFESTAGE,segment = PREMIUM_CUSTOMER) %>%
         summarise(n = n()) %>% arrange(segment,LIFESTAGE, desc(n))%>% head(5)
```

```
## # A tibble: 5 x 3
## # Groups:   LIFESTAGE [5]
##   LIFESTAGE              segment     n
##   <chr>                  <chr>   <int>
## 1 MIDAGE SINGLES/COUPLES Budget   1473
## 2 NEW FAMILIES           Budget    840
## 3 OLDER FAMILIES         Budget   6539
## 4 OLDER SINGLES/COUPLES  Budget   5172
## 5 RETIREES               Budget   4305
```

- What's the average chip price by customer segment

```
#- What's the average chip price by customer segment
chip %>% group_by(Customer_segment = PREMIUM_CUSTOMER) %>%
  summarise(avg_bill = round(mean(TOT_SALES),digit = 1),
            spends = round(sum(TOT_SALES),digit = 1)) %>%
  arrange(desc(spends))
```

```
## # A tibble: 3 x 3
```

```
##   Customer_segment avg_bill  spends
##   <chr>               <dbl>   <dbl>
## 1 Mainstream            6.9 197981.
## 2 Budget                6.8 180185.
## 3 Premium               6.8 133423.
```

The most sales contribution is from the mainstream segment with the average bill if 6.9 and total of 197980.7 USD

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

```
#avg totl avg by cus
sale_g <- data %>% summarise(avg_grocery_spend = round(mean(TOT_SALES), digits = 1),
                   sum_grocery = sum(TOT_SALES),1)

sale_c <- chip %>% summarise(avg_chip_spending = round(mean(TOT_SALES), digit = 1),
                   sum_chip = round(sum(TOT_SALES), digits = 1),1)
```

```
gro_sp_avg <- sale_pp[[1]]
gro_sp <- sale_pp[[2]]
pp_per <- round(sale_pp[[5]]/sale_pp[[2]]*100,digits = 1)
cp_sal <- sale_pp[[5]]
cp_avg <- sale_pp[[4]]


glue('The total spend on the grocery :{gro_sp} USD, average : {gro_sp_avg} USD
     The total chip sale : {cp_sal} USD, average : {cp_avg} USD
     with {pp_per}% proportion of their grocery spend is on chips')
```

```
## The total spend on the grocery :1933115 USD, average : 7.3 USD
## The total chip sale : 511588.3 USD, average : 6.9 USD
## with 26.5% 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

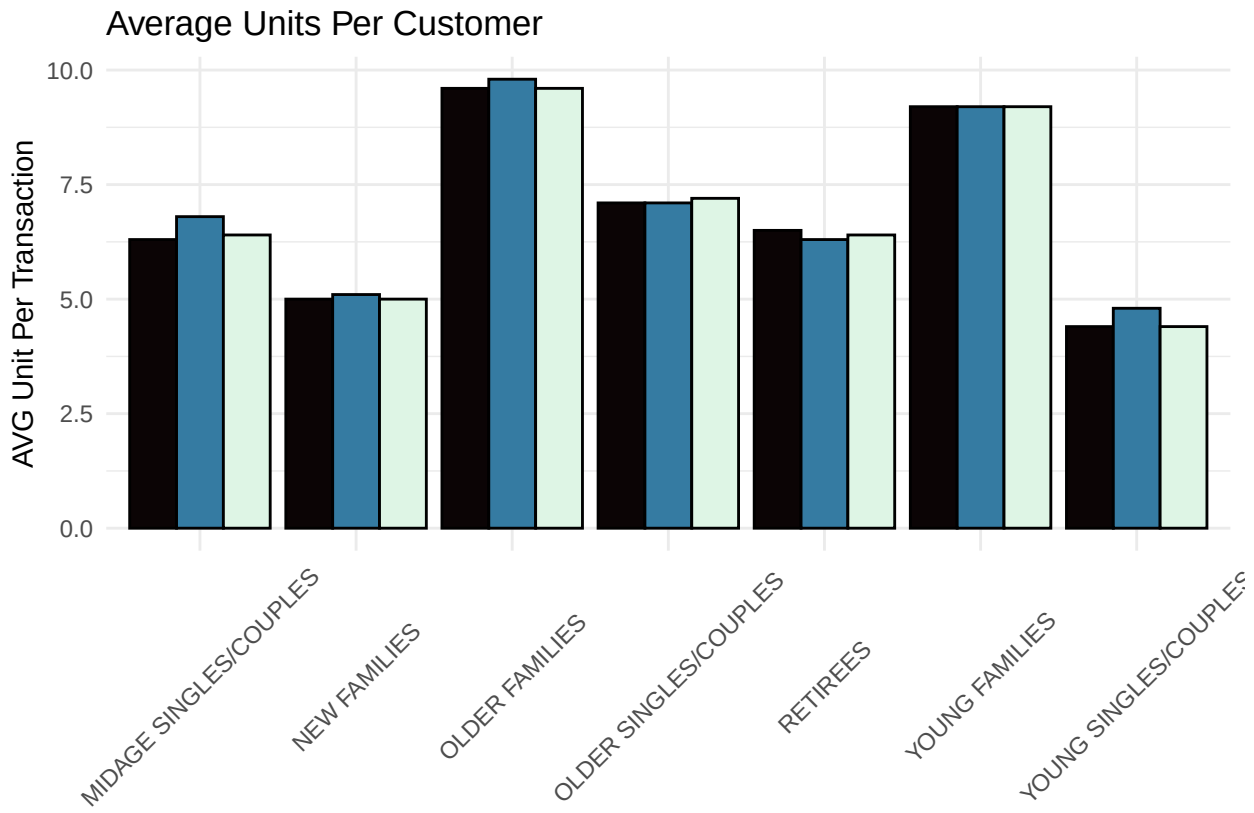
```
c_cust <-chip %>% distinct(LYLTY_CARD_NBR) %>% summarise(purchase_chips = n())
groc_cust <- data %>% distinct(LYLTY_CARD_NBR) %>% summarise(purchase_grocery =n())

prop_c_g <- round((c_cust[[1]]/groc_cust[[1]])*100,digit=1)
glue('The number of customer that purchase chip is {c_cust} and over all number of customer is {groc_cus
```

```
## The number of customer that purchase chip is 43625 and over all number of customer is 72636 and the p
```

Average number of units per customer by LIFESTAGE and PREMIUM_CUSTOMER Calculate and plot the average number of units per customer by those two dimensions.

```
unt_p_cust <- data %>% group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
    summarise(unit_per_cust = round(sum(PROD_QTY)/uniqueN(LYLTY_CARD_NBR),digits = 1))
```

```
ggplot(unt_p_cust,aes(weight = unit_per_cust, x =LIFESTAGE,
                      fill = PREMIUM_CUSTOMER))+
     geom_bar(position = 'dodge',color = 'black')+theme_minimal()+
     theme(axis.text.x = element_text(angle = 45, vjust = 0.5))+
     scale_fill_viridis(discrete=TRUE, option="G",guide="none")+
     labs(y ="AVG Unit Per Transaction",x = "", title = "Average Units Per Customer")
```




```
# (average sale price) by those two customer dimensions
sle_p_unt <- data %>% group_by(LIFESTAGE, PREMIUM_CUSTOMER) %>%
    summarise(sl_per_unt = round(sum(TOT_SALES)/sum(PROD_QTY),digits = 1))
sle_p_unt %>% head(5)
```

Average price per unit by LIFESTAGE and PREMIUM_CUSTOMER

```
## # A tibble: 5 x 3
## # Groups:   LIFESTAGE [2]
##   LIFESTAGE              PREMIUM_CUSTOMER sl_per_unt
##   <chr>                  <chr>                <dbl>
## 1 MIDAGE SINGLES/COUPLES Budget                 3.7
## 2 MIDAGE SINGLES/COUPLES Mainstream             4
## 3 MIDAGE SINGLES/COUPLES Premium                3.8
## 4 NEW FAMILIES           Budget                 3.9
## 5 NEW FAMILIES           Mainstream             3.9
```

```
ggplot(sle_p_unt,aes(weight = sl_per_unt, x =LIFESTAGE,
                     fill = PREMIUM_CUSTOMER))+
     geom_bar(position = 'dodge',color = 'black')+theme_minimal()+
     theme(axis.text.x = element_text(angle = 45, vjust = 0.5))+
     scale_fill_brewer(palette = "YlGnBu")+
     labs(y ="AVG Price Per Unit",x = "", title = "Average Sale Price Per Unit")
```

Sirawit N. credit : Quantium