Quantium Virtual Internship Task- 1

Talib

installing package

```
#install.packages("tidyverse")
#install.packages("readxl")
#install.packages("skimr")
#install.packages("janitor")
```

loading packages

```
library(tidyverse)
```

```
## Warning: package ’tidyverse’ was built under R version 4.3.2

## Warning: package ’ggplot2’ was built under R version 4.3.2

## Warning: package ’tidyr’ was built under R version 4.3.2

## Warning: package ’stringr’ was built under R version 4.3.2

## Warning: package ’lubridate’ was built under R version 4.3.2

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## 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     1.0.1
## -- 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 error
```

```
library(readr)
library(skimr)
```

```
## Warning: package ’skimr’ was built under R version 4.3.2
```

```r
library(janitor)
```

```
## Warning: package 'janitor' was built under R version 4.3.2
```

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

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

```
## Warning: package 'data.table' was built under R version 4.3.2
```

```
##
## 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
```

importing transaction file

(checked in excel there is only 1 sheet in this workbook & converted to csv) And purchase behavior file(it has also 1 sheet)

```r
cust_beh1 <- fread("C:\\Users\\dexter\\Documents\\tlb docs\\Data_Analytics\\Forage\\QVI_purchase_behavio

#fread to get glimpse of first 5 & last5 rows
trns_1 <- fread("C:\\Users\\dexter\\Documents\\tlb docs\\Data_Analytics\\Forage\\QVI_transaction_data.cs

head(trns_1)
```

```
##     DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
## 1: 43390         1           1000      1        5
## 2: 43599         1           1307    348       66
## 3: 43605         1           1343    383       61
## 4: 43329         2           2373    974       69
## 5: 43330         2           2426   1038      108
## 6: 43604         4           4074   2982       57
##                                    PROD_NAME PROD_QTY TOT_SALES
## 1:   Natural Chip        Compny SeaSalt175g        2       6.0
## 2:                 CCs Nacho Cheese    175g        3       6.3
## 3:   Smiths Crinkle Cut  Chips Chicken 170g        2       2.9
## 4:   Smiths Chip Thinly  S/Cream&Onion 175g        5      15.0
## 5: Kettle Tortilla ChpsHny&Jlpno Chili 150g        3      13.8
## 6: Old El Paso Salsa   Dip Tomato Mild 300g        1       5.1
```

```
head(cust_beh1)
```

```
##    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
```

data type

Transaction table

```
str(trns_1)
```

```
## Classes 'data.table' and 'data.frame':   264836 obs. of  8 variables:
##  $ DATE          : int  43390 43599 43605 43329 43330 43604 43601 43601 43332 43330 ...
##  $ 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
##  $ 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>
```

```
trns_1 %>%
    summarize_all(class) %>%
    gather(variable, class)
```

```
##         variable     class
## 1           DATE   integer
## 2      STORE_NBR   integer
## 3 LYLTY_CARD_NBR   integer
## 4         TXN_ID   integer
## 5       PROD_NBR   integer
## 6      PROD_NAME character
## 7       PROD_QTY   integer
## 8      TOT_SALES   numeric
```

data type of customer behavior table

```
str(cust_beh1)
```

```
## 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 SI
##  $ PREMIUM_CUSTOMER: chr  "Premium" "Mainstream" "Budget" "Mainstream" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
cust_beh1 %>%
    summarize_all(class) %>%
    gather(variable, class)
```

```
##           variable     class
## 1   LYLTY_CARD_NBR   integer
## 2        LIFESTAGE character
## 3 PREMIUM_CUSTOMER character
```

Formatting data type

Of DATE column and assigning to new variable

```
trns_1Date <- trns_1
trns_1$DATE <- as.Date(trns_1$DATE,origin="1899-12-30")
```

summary of tables

looking at summary of tables

```
skim_without_charts(trns_1)
```

Table 1: Data summary

Name	trns_1
Number of rows	264836
Number of columns	8
Key	NULL
Column type frequency:	
character	1
Date	1
numeric	6
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: 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
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
skim_without_charts(cust_beh1)
```

Table 5: Data summary

Name	cust_beh1
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

This transaction data-set has 264,836 rows and 8 cols(1 character,7 Numeric) and Customer behavior has(72,637 rows & 3cols 2character,1 Numeric) **Mismatch of number of rows in tables**

Common column-CARD_NBR

For 2 quantitative columns of transaction table will be logical to get statistical description Mean is 2 but 4th quartile is 200 same for sales column mean is 7.4 but 4th quartile is 650 which is a huge difference from mean

In DATE column the date difference is 364 days. 01,July 2018 to 30,June 2019

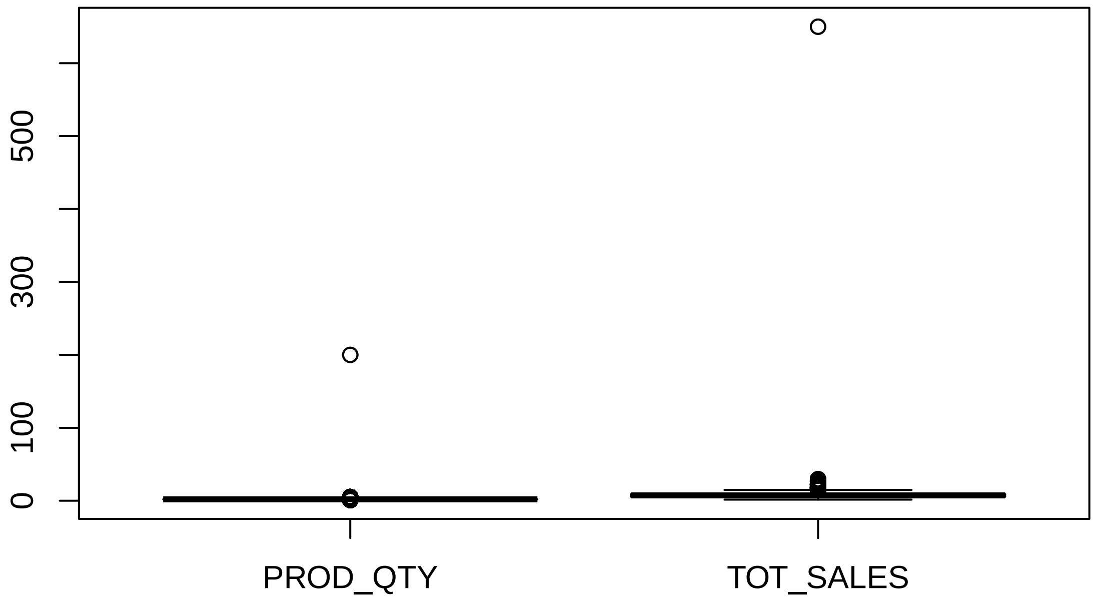
In customer table there are 7 unique values in LIFESTAGE and 3 in PREMIUM CUSTOMER column.

And there are no missing values in trns & customer behavior table

Finding outliers

Plotting box-whisker for Product quantity and sales

```
trns_1 %>% select(PROD_QTY,TOT_SALES) %>% boxplot()
```



Identifying outliers

```
trns_1%>% filter(TOT_SALES>50 & PROD_QTY>50)
```

```
##          DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
## 1: 2018-08-19       226         226000 226201        4
## 2: 2019-05-20       226         226000 226210        4
##                           PROD_NAME PROD_QTY TOT_SALES
## 1: Dorito Corn Chp     Supreme 380g      200       650
## 2: Dorito Corn Chp     Supreme 380g      200       650
```

Removing these extreme outliers

```
trns_out <- trns_1 %>% filter(PROD_QTY<50)

nrow(trns_1)
```

```
## [1] 264836
```

```
noquote("Rows after outliers removed")
```

```
## [1] Rows after outliers removed
```

```
nrow(trns_out)
```

```
## [1] 264834
```

removed 2 rows that were extreme outliers

```
summary(trns_out$PROD_QTY)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   1.000   2.000   2.000   1.906   2.000   5.000
```

```
summary(trns_out$TOT_SALES)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   1.500   5.400   7.400   7.299   9.200  29.500
```

checking this max 29.5 sales

```
trns_out %>% filter(TOT_SALES==29.5) %>% select(PROD_QTY,TOT_SALES)
```

```
##    PROD_QTY TOT_SALES
## 1:        5      29.5
## 2:        5      29.5
## 3:        5      29.5
## 4:        5      29.5
## 5:        5      29.5
## 6:        5      29.5
## 7:        5      29.5
```

As Product quantity is 5 this can be taken into consideration for analysis.

Exploring product name col

```
trns_out[,.N,PROD_NAME]
```

```
##                                    PROD_NAME    N
##   1:   Natural Chip        Compny SeaSalt175g 1468
##   2:                   CCs Nacho Cheese    175g 1498
##   3:   Smiths Crinkle Cut  Chips Chicken 170g 1484
##   4:   Smiths Chip Thinly  S/Cream&Onion 175g 1473
##   5: Kettle Tortilla ChpsHny&Jlpno Chili 150g 3296
##  ---
## 110:    Red Rock Deli Chikn&Garlic Aioli 150g 1434
## 111:      RRD SR Slow Rst     Pork Belly 150g 1526
## 112:                   RRD Pc Sea Salt     165g 1431
## 113:       Smith Crinkle Cut   Bolognese 150g 1451
## 114:                   Doritos Salsa Mild  300g 1472
```

As I am asked to analyse "Chips" products only, I will remove all the other products. Checking most common words - by getting unique values > splitting each string into substring > unlisting all the substrings.

```
products <- data.table(unlist(strsplit(unique(trns_out[,PROD_NAME]), " ")))
setnames(products,"common_words")
```

Removing numbers, special characters the PROD_NAME col having g/G from Prod name also removing "/". (grepl to remove the substring/characters)

```
#Removing Numbers
products <- products[grepl("\\d",common_words) == FALSE, ]
#Removing special characters
products <- products[grepl("[:alpha:]", common_words), ]
#Most common_words by counting the number of times a word appears then sort by descending order
products[, .N, common_words][order(N, decreasing = TRUE)]
```

```
##      common_words  N
##   1:        Chips 21
##   2:       Smiths 16
##   3:      Crinkle 14
##   4:       Kettle 13
##   5:       Cheese 12
##  ---
## 127: Chikn&Garlic  1
## 128:        Aioli  1
## 129:         Slow  1
## 130:        Belly  1
## 131:    Bolognese  1
```

Checking Date Col

As we saw earlier 1 date is missing

```
#unique dates
length(unique(trns_out$DATE))
```

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

```
#See all dates with instances.
trns_out[,.N,DATE]
```

```
##            DATE   N
##   1: 2018-10-17 732
##   2: 2019-05-14 758
##   3: 2019-05-20 754
##   4: 2018-08-17 711
##   5: 2018-08-18 737
##  ---
## 360: 2018-11-21 700
## 361: 2019-05-10 710
## 362: 2018-12-08 672
## 363: 2019-01-30 738
## 364: 2019-02-09 718
```

364 DATES, one date is missing, checking which date is missing

```
#new variable for date range
date_range <- seq(min(trns_out$DATE), max(trns_out$DATE), by = 1)
date_range[!date_range %in% trns_out$DATE]
```

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

It's 25th Dec, due to Christmas stores must be closed and therefore no data.

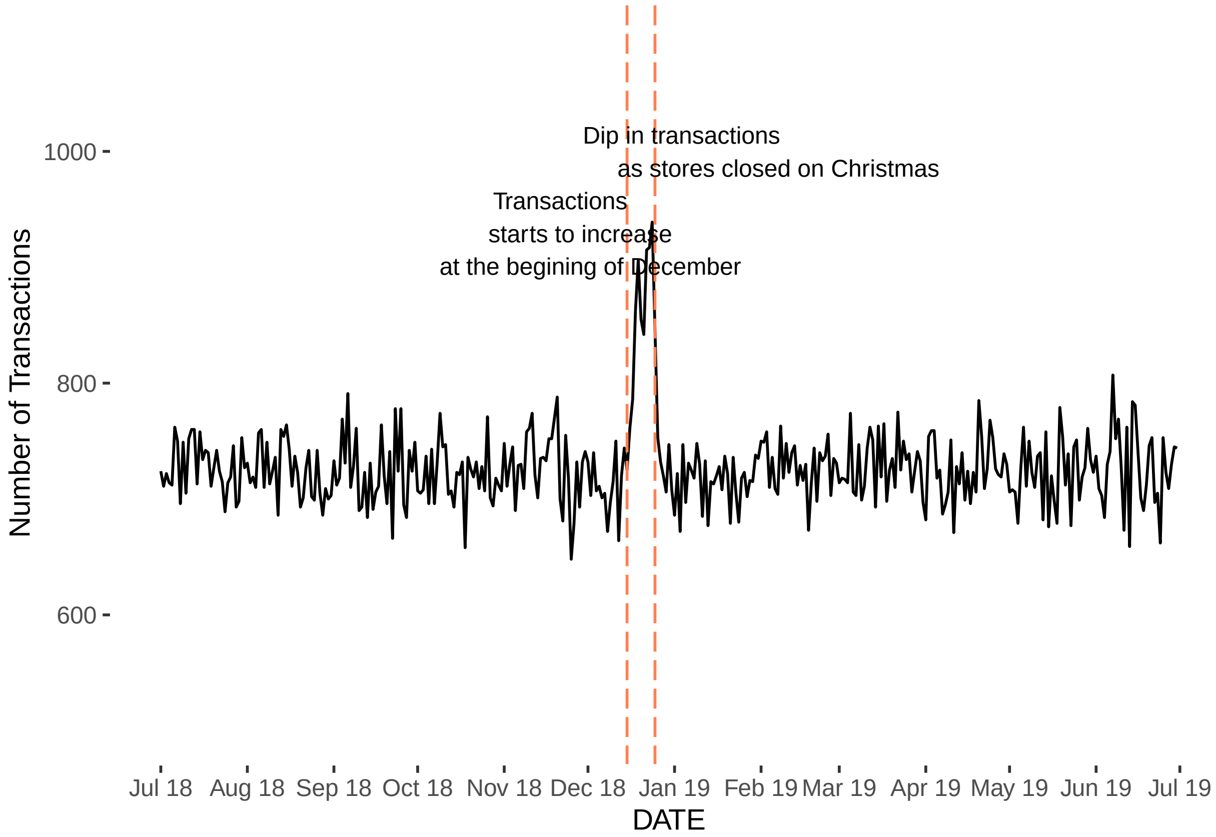
Let's see this visually.

```
transactionByDate <- trns_out[, .N, by = DATE]

transactDate <- tribble(
  ~ DATE, ~ event,
  "2018-12-25", "          Dip in transactions  \n                                       as stores clos
  "2018-12-15", "\n \n \n \n \n Transactions                     \nstarts to increase               \nat
) %>% mutate(DATE = as.Date(DATE))

ggplot(transactionByDate, aes(x=DATE, y=N)) +
  coord_cartesian( ylim = c(500, 1100))+
  geom_line() +
  scale_x_date(date_labels="%b %y",date_breaks  ="1 month")+
  geom_vline(data = transactDate,aes(xintercept=DATE),linetype="longdash",
    color="coral",size=0.5) +
    geom_text(data=transactDate,aes(DATE,label=event),y=1000, size=3)+
    ylab("Number of Transactions") +
  #to remove background and modify axis color
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
panel.background = element_blank(), axis.line = element_line(colour = "white"))
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



After finding out the number of transactions trend, I labeled the graph accordingly to use it in presentation. Now I will check if there are any duplicates in data.

checking duplicates

From Janitor package

```
get_dupes(trns_out )
```

```
## No variable names specified - using all columns.
```

```
##          DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR
## 1: 2018-10-01       107         107024 108462       45
## 2: 2018-10-01       107         107024 108462       45
##                                PROD_NAME PROD_QTY TOT_SALES dupe_count
## 1: Smiths Thinly Cut   Roast Chicken 175g        2         6          2
## 2: Smiths Thinly Cut   Roast Chicken 175g        2         6          2
```

```
get_dupes(cust_beh1)
```

```
## No variable names specified - using all columns.

## No duplicate combinations found of: LYLTY_CARD_NBR, LIFESTAGE, PREMIUM_CUSTOMER

## Empty data.table (0 rows and 4 cols): LYLTY_CARD_NBR,LIFESTAGE,PREMIUM_CUSTOMER,dupe_count
```

1 duplicate found having same values for all columns, no duplicates in customer behavior

Trnasforming table

assigning to new variable & changing the col names

```
trns_out <- trns_out %>%  rename(CARD_NBR = LYLTY_CARD_NBR)
unique_trns <- unique(trns_out)

cust_beh1 <- cust_beh1 %>% rename(CARD_NBR = LYLTY_CARD_NBR )
```

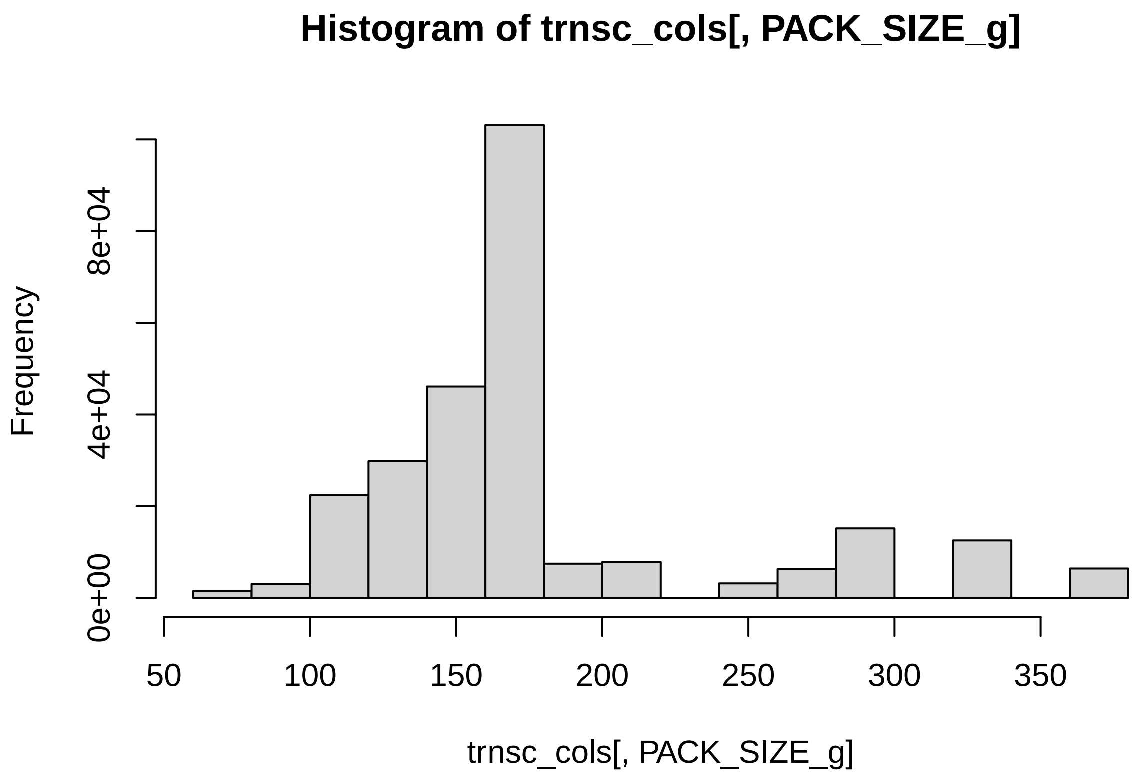
Now total 3 rows removed., creating new col PACK SIZE in trnasaction table (gsub to extract the subsrtings)

```
trnsc_cols <- unique_trns
trnsc_cols[, PACK_SIZE_g :=as.numeric(gsub("\\D","",unique_trns$PROD_NAME))]
trnsc_cols
```

```
##               DATE STORE_NBR 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
##     ---
## 264829: 2019-03-09       272   272319 270088       89
## 264830: 2018-08-13       272   272358 270154       74
## 264831: 2018-11-06       272   272379 270187       51
## 264832: 2018-12-27       272   272379 270188       42
## 264833: 2018-09-22       272   272380 270189       74
##                                         PROD_NAME PROD_QTY TOT_SALES PACK_SIZE_g
##      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
##     ---
## 264829:  Kettle Sweet Chilli And Sour Cream 175g        2      10.8         175
## 264830:            Tostitos Splash Of  Lime 175g        1       4.4         175
## 264831:                 Doritos Mexicana    170g        2       8.8         170
## 264832:  Doritos Corn Chip Mexican Jalapeno 150g        2       7.8         150
## 264833:            Tostitos Splash Of  Lime 175g        2       8.8         175
```

Checking distribution pack size column with histogram

```
hist(trnsc_cols[,PACK_SIZE_g])
```



The size distribution seems fine also it make sense.

Now I will create brand name column(with help of sample solution)

```
#Finding the brand names
trnsc_cols[,.N,substr(PROD_NAME, 1, regexpr(' ',PROD_NAME) - 1)][order(N,decreasing=TRUE)]
```

```
##         substr     N
##  1:     Kettle 41288
##  2:     Smiths 28859
##  3:   Pringles 25102
##  4:    Doritos 24962
##  5:      Thins 14075
##  6:        RRD 11894
##  7:  Infuzions 11057
##  8:         WW 10320
##  9:       Cobs  9693
## 10:   Tostitos  9471
## 11:   Twisties  9454
## 12:        Old  9324
## 13:   Tyrrells  6442
## 14:      Grain  6272
## 15:    Natural  6050
## 16:        Red  5885
## 17:   Cheezels  4603
## 18:        CCs  4551
## 19: Woolworths  4437
## 20:     Dorito  3183
## 21:     Infzns  3144
## 22:      Smith  2963
## 23:    Cheetos  2927
## 24:      Snbts  1576
## 25:     Burger  1564
## 26:    GrnWves  1468
## 27:   Sunbites  1432
## 28:        NCC  1419
## 29:     French  1418
##          substr     N
```

```r
trnsc_cols[, BRAND := toupper(substr(PROD_NAME, 1, regexpr(pattern = ' ',PROD_NAME) - 1))]
#checking names
trnsc_cols[,.N,BRAND][order(-N)]
```

```
##           BRAND     N
##  1:      KETTLE 41288
##  2:      SMITHS 28859
##  3:    PRINGLES 25102
##  4:     DORITOS 24962
##  5:       THINS 14075
##  6:         RRD 11894
##  7:   INFUZIONS 11057
##  8:          WW 10320
##  9:        COBS  9693
## 10:    TOSTITOS  9471
## 11:    TWISTIES  9454
## 12:         OLD  9324
## 13:    TYRRELLS  6442
## 14:       GRAIN  6272
## 15:     NATURAL  6050
## 16:         RED  5885
## 17:    CHEEZELS  4603
## 18:         CCS  4551
## 19:  WOOLWORTHS  4437
## 20:      DORITO  3183
## 21:      INFZNS  3144
## 22:       SMITH  2963
## 23:     CHEETOS  2927
## 24:       SNBTS  1576
## 25:      BURGER  1564
## 26:     GRNWVES  1468
## 27:    SUNBITES  1432
## 28:         NCC  1419
## 29:      FRENCH  1418
##           BRAND     N
```

29 brands total. I took help of sample solution as there were some names repeated, such as RED and RRD,

which are both Red Rock Deli chips.

```
#Cleaning brand name
trnsc_cols[BRAND == "RED",BRAND:="RRD"]
trnsc_cols[BRAND=="WW",BRAND:="WOOLWORTHS"]
trnsc_cols[BRAND=="INFZNS",BRAND:="INFUZIONS"]
```

After cleaning 26 brands are in the dataset.

As we already seen how many unique values are in customer table we will see what are those values. checking unique values in customer behavior

```
unique(cust_beh1$LIFESTAGE)
```

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

```
print("**PREMIUM_CUSTOMER**")
```

```
## [1] "**PREMIUM_CUSTOMER**"
```

```
unique(cust_beh1$PREMIUM_CUSTOMER)
```

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

As there are no issue in this table I will merge both of the tables.

#merging the tables with common records only

```
Merged_Data <-  merge(x=trnsc_cols,y=cust_beh1,by="CARD_NBR")
```

checking rows and cols

```
skim_without_charts(Merged_Data)
```

Table 8: Data summary

Name	Merged_Data
Number of rows	264833
Number of columns	12
Key	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	114	0
BRAND	0	1	3	10	0	26	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
CARD_NBR	0	1	135548.90	80580.03	1000.0	70021.0	130357.0	203094.0	2373711.0
STORE_NBR	0	1	135.08	76.78	1.0	70.0	130.0	203.0	272.0
TXN_ID	0	1	135157.72	78133.05	1.0	67600.0	135137.0	202700.0	2415841.0
PROD_NBR	0	1	56.58	32.83	1.0	28.0	56.0	85.0	114.0
PROD_QTY	0	1	1.91	0.34	1.0	2.0	2.0	2.0	5.0
TOT_SALES	0	1	7.30	2.53	1.5	5.4	7.4	9.2	29.5
PACK_SIZE_g	0	1	182.43	64.33	70.0	150.0	170.0	175.0	380.0

Now the final table has 264833 rows and 12 columns without nulls.

#saving the file in csv format

```
write.csv(Merged_Data,"C:\\Users\\dexter\\Documents\\tlb docs\\Data_Analytics\\Forage\\Merged_Data.csv"
```

Data analysis.

Setting metrics- 1.Customer segment who are spending most, 2.Chips bought per customer by segment, 3.Number of customers in each segment, 4.Avg sales by customer segment.

```
#Numbers of customer in each segment

Merged_Data %>% group_by(LIFESTAGE) %>%  summarise(TOT_SALES = sum(TOT_SALES))
```

```
## # A tibble: 7 x 2
##   LIFESTAGE              TOT_SALES
##   <chr>                      <dbl>
## 1 MIDAGE SINGLES/COUPLES   184751.
## 2 NEW FAMILIES              50433.
## 3 OLDER FAMILIES           352467.
## 4 OLDER SINGLES/COUPLES    402421.
## 5 RETIREES                 366471.
## 6 YOUNG FAMILIES           316160.
## 7 YOUNG SINGLES/COUPLES    260405.
```

```
Merged_Data %>% group_by(PREMIUM_CUSTOMER) %>%  summarise(TOT_SALES = sum(TOT_SALES))
```

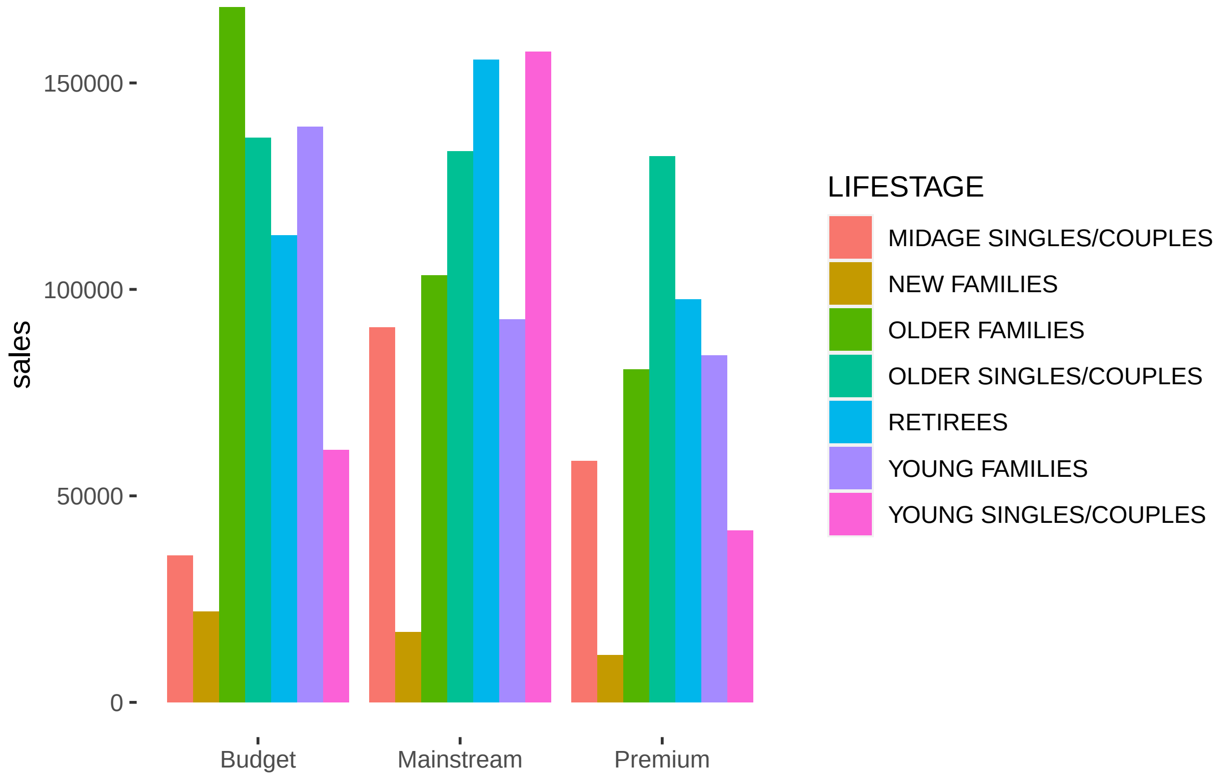
```
## # A tibble: 3 x 2
##   PREMIUM_CUSTOMER TOT_SALES
##   <chr>                <dbl>
## 1 Budget             676212.
## 2 Mainstream         750744.
## 3 Premium            506153.
```

Mainstream customers are spending most in premium_customer and OLDER SINGLES/COUPLES in LIFESTAFE column. visualizing and comparing between these two categorical variables.

```
#creating a variable sum of customers segmented by PREMIUM_CUSTOMER and LIFE_STAGE
sumPC <- Merged_Data  %>% group_by(LIFESTAGE,PREMIUM_CUSTOMER) %>%  summarise(sales=sum(TOT_SALES))
```

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

```
ggplot(data=sumPC) +
  geom_col(mapping=aes(x=PREMIUM_CUSTOMER,y=sales,fill=LIFESTAGE),position='dodge') +
   theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
panel.background = element_blank(), axis.line = element_line(colour = "white"))+
  xlab("")
```

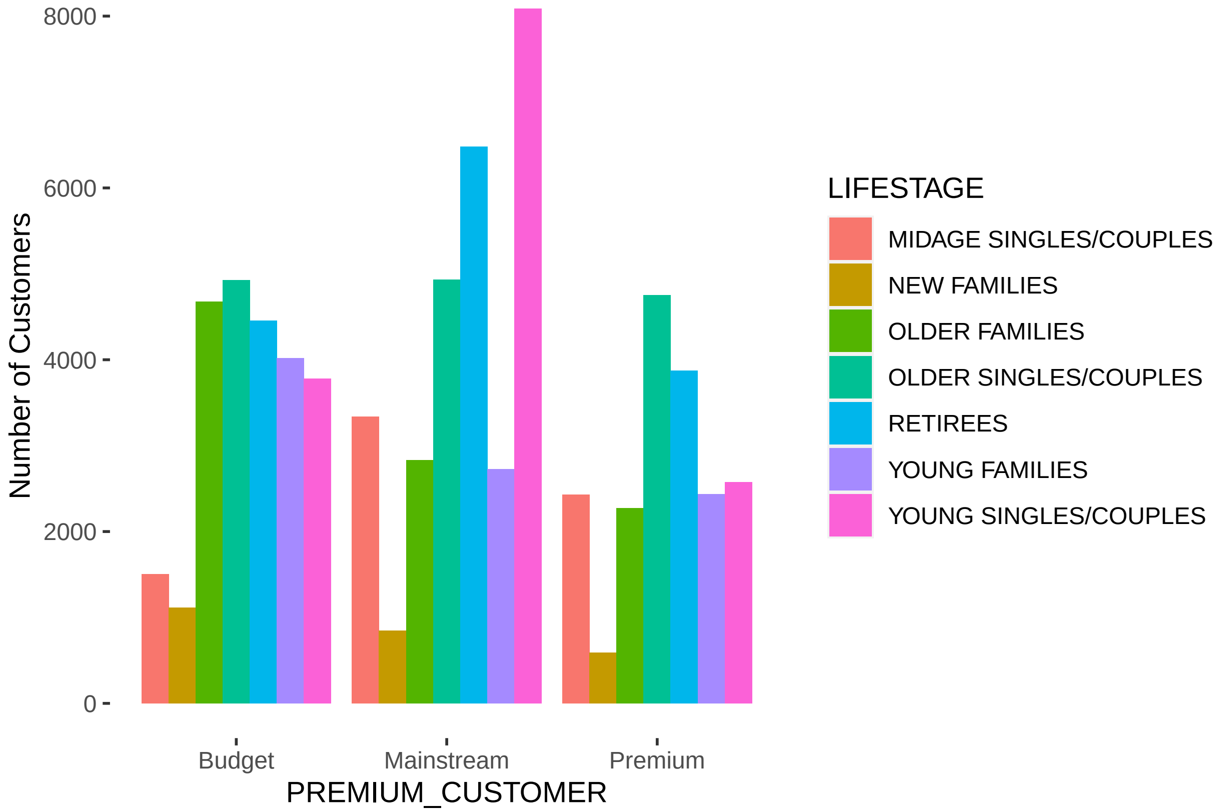


Most sales coming from Budget-Midage single/couples,then Mainstream-young single/couples. Sales are high but is it because there are more customers or few customers buying more chips? Let's find out.

```
custPC <- Merged_Data  %>% group_by(LIFESTAGE,PREMIUM_CUSTOMER) %>%  summarise(CustNum=length(unique(CA
```

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

```
ggplot(data=custPC) +
  geom_col(mapping=aes(x=PREMIUM_CUSTOMER,y=CustNum,fill=LIFESTAGE),position='dodge') +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
panel.background = element_blank(), axis.line = element_line(colour = "white"))+
  ylab("Number of Customers")
```

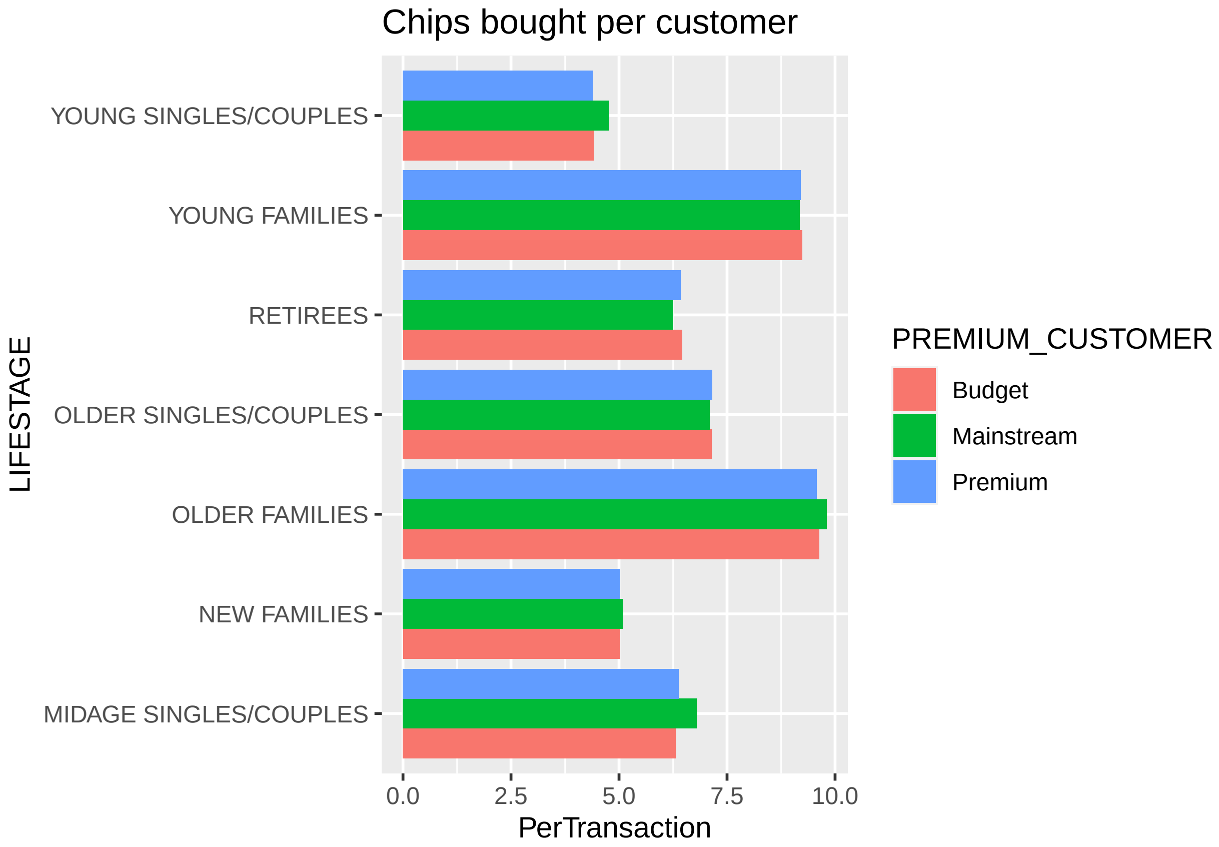


Highest number of customers are in Mainstream-Young Single/Couples segment that is the reason for more sales in this segment. But this is not the case for Budget-midage sement. Now, lets find if more chips are being bought per customer in the above segments or the otherwise.

```
chipsPerCust <- Merged_Data %>% group_by(LIFESTAGE,PREMIUM_CUSTOMER) %>% summarise(PerTransaction=sum(P
```

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

```
ggplot(data=chipsPerCust) +
  geom_col(mapping=aes(x=PerTransaction,y=LIFESTAGE,fill=PREMIUM_CUSTOMER), position='dodge') +
  labs(title='Chips bought per customer')
```

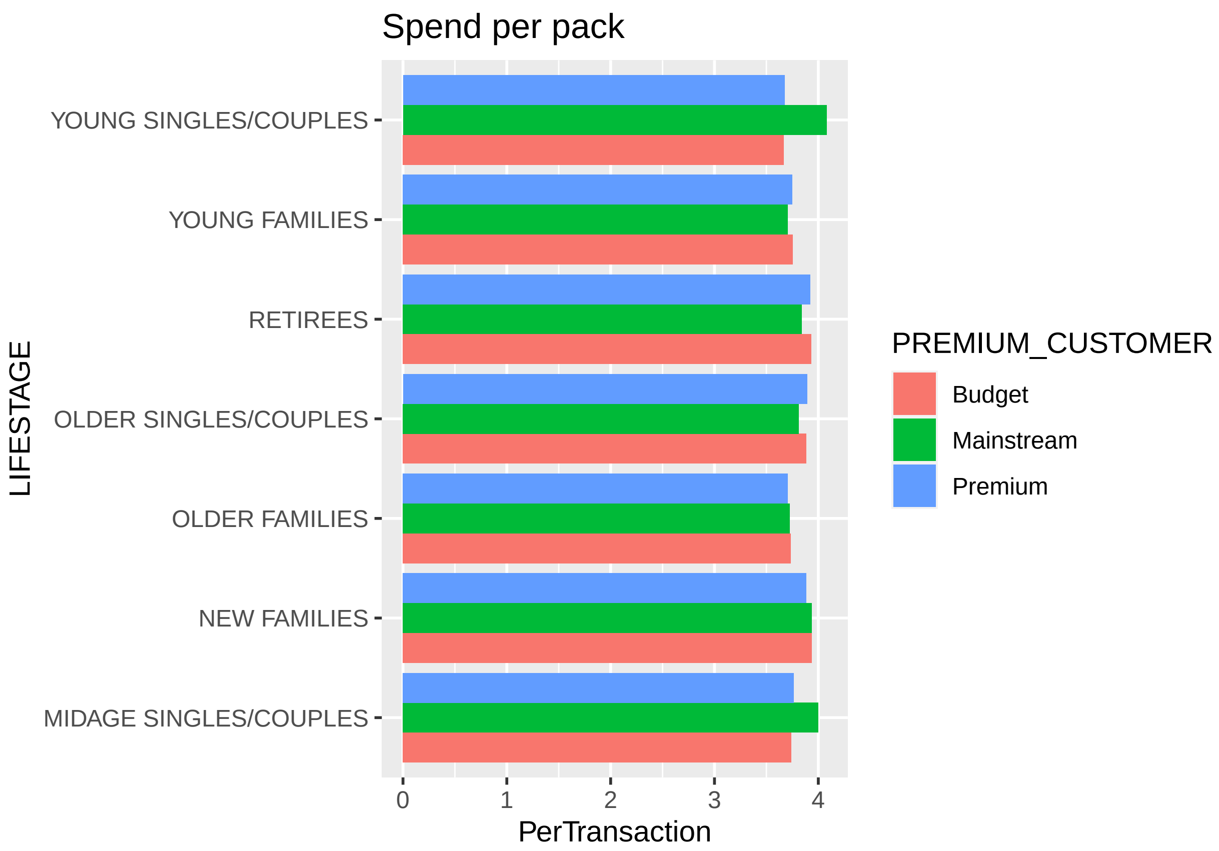


Mainstream-Older families segment are buying more chips, and then Mainstream-Young families. Now, we will find the segment spending most per pack by calculating average price per unit chips bought.

```
PricePerUnit <- Merged_Data %>% group_by(LIFESTAGE,PREMIUM_CUSTOMER) %>% summarise(PerTransaction=sum(T
```

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

```
ggplot(data=PricePerUnit) +
  geom_col(mapping=aes(x=PerTransaction,y=LIFESTAGE,fill=PREMIUM_CUSTOMER), position='dodge') +
  labs(title='Spend per pack')
```



Mainstream-Young single couple & Mid-age single/couples are tend to spend more per unit. Now, since mainstream young single couple segment are contributing in sales significantly we will see the brand they prefer and size of packet.

```
target <- Merged_Data %>% filter(LIFESTAGE=='YOUNG SINGLES/COUPLES' & PREMIUM_CUSTOMER=='Mainstream')
other <- Merged_Data %>% filter(LIFESTAGE!='YOUNG SINGLES/COUPLES'& PREMIUM_CUSTOMER!='Mainstream')
##Quantity
qty_target <- target %>% summarize(sum(PROD_QTY)) %>% as.numeric()
qty_other <- other %>% summarize(sum(PROD_QTY)) %>%  as.numeric()

##quantity by brand
qty_target_brand <- target  %>% summarise(TargetSegment=sum(PROD_QTY)/qty_target,.by = BRAND)
qty_other_brand <- other %>% summarise(OtherSegment=sum(PROD_QTY)/qty_other,.by = BRAND)

brand_proportions <- merge(qty_target_brand,qty_other_brand) %>% mutate(affinity=TargetSegment/OtherSeg
setDT(brand_proportions)
#brand_proportions=data.table(brand_proportions)
brand_proportions[order(-affinity)]
```

```
##          BRAND TargetSegment OtherSegment  affinity
## 1:    TYRRELLS   0.029586871  0.023967746 1.2344453
## 2:      DORITO   0.014728722  0.011985641 1.2288639
## 3:    TWISTIES   0.043306068  0.035355697 1.2248682
## 4:      KETTLE   0.185649203  0.155243939 1.1958548
## 5:    TOSTITOS   0.042581280  0.035744726 1.1912605
## 6:         OLD   0.041597639  0.034931301 1.1908414
##  7:   PRINGLES   0.111979706  0.094240597 1.1882321
##  8:      GRAIN   0.027308967  0.023200297 1.1770956
##  9:       COBS   0.041856492  0.035836678 1.1679791
## 10:    DORITOS   0.108148685  0.093292780 1.1592396
## 11:  INFUZIONS   0.060649203  0.053509222 1.1334346
## 12:      THINS   0.056611100  0.053275804 1.0626043
## 13:   CHEEZELS   0.016851315  0.017619494 0.9564018
## 14:     SMITHS   0.087233382  0.109794699 0.7945136
## 15:     FRENCH   0.003701595  0.005319093 0.6959072
## 16:    CHEETOS   0.007532615  0.010960018 0.6872813
## 17:        RRD   0.045376890  0.068310021 0.6642787
## 18:    NATURAL   0.014961690  0.023207370 0.6446956
## 19:        CCS   0.010483537  0.017191562 0.6098071
## 20:        NCC   0.003416856  0.005647999 0.6049676
## 21:    GRNWVES   0.003365086  0.005757635 0.5844563
## 22:      SMITH   0.006186581  0.011525879 0.5367557
## 23:      SNBTS   0.003261545  0.006295203 0.5181001
## 24: WOOLWORTHS   0.028189066  0.056232427 0.5012956
## 25:   SUNBITES   0.002692069  0.005460558 0.4930025
## 26:     BURGER   0.002743839  0.006093615 0.4502811
##           BRAND TargetSegment OtherSegment  affinity
```

mainstream young single couple segment are tend to buy TYRRELLS chips most and BURGER the least. Now we will check the pack size they prefer.

```
##quantity by brand
qty_target_pack <- target  %>% summarise(TargetSegment=sum(PROD_QTY)/qty_target,.by = PACK_SIZE_g)
qty_other_pack <- other %>% summarise(OtherSegment=sum(PROD_QTY)/qty_other,.by = PACK_SIZE_g)

pack_proportions <- merge(qty_target_pack,qty_other_pack) %>% mutate(affinity=TargetSegment/OtherSegmen
setDT(brand_proportions)
#brand_proportions=data.table(brand_proportions)
brand_proportions[order(-affinity)]
```

```
##           BRAND TargetSegment OtherSegment  affinity
##  1:   TYRRELLS   0.029586871  0.023967746 1.2344453
##  2:     DORITO   0.014728722  0.011985641 1.2288639
##  3:   TWISTIES   0.043306068  0.035355697 1.2248682
##  4:     KETTLE   0.185649203  0.155243939 1.1958548
##  5:   TOSTITOS   0.042581280  0.035744726 1.1912605
##  6:        OLD   0.041597639  0.034931301 1.1908414
##  7:   PRINGLES   0.111979706  0.094240597 1.1882321
##  8:      GRAIN   0.027308967  0.023200297 1.1770956
##  9:       COBS   0.041856492  0.035836678 1.1679791
## 10:    DORITOS   0.108148685  0.093292780 1.1592396
## 11:  INFUZIONS   0.060649203  0.053509222 1.1334346
## 12:      THINS   0.056611100  0.053275804 1.0626043
## 13:   CHEEZELS   0.016851315  0.017619494 0.9564018
## 14:     SMITHS   0.087233382  0.109794699 0.7945136
## 15:     FRENCH   0.003701595  0.005319093 0.6959072
## 16:    CHEETOS   0.007532615  0.010960018 0.6872813
## 17:        RRD   0.045376890  0.068310021 0.6642787
## 18:    NATURAL   0.014961690  0.023207370 0.6446956
## 19:        CCS   0.010483537  0.017191562 0.6098071
## 20:        NCC   0.003416856  0.005647999 0.6049676
## 21:    GRNWVES   0.003365086  0.005757635 0.5844563
## 22:      SMITH   0.006186581  0.011525879 0.5367557
## 23:      SNBTS   0.003261545  0.006295203 0.5181001
## 24: WOOLWORTHS   0.028189066  0.056232427 0.5012956
## 25:   SUNBITES   0.002692069  0.005460558 0.4930025
## 26:     BURGER   0.002743839  0.006093615 0.4502811
##          BRAND TargetSegment OtherSegment  affinity
```

Mainstream young single couple segment are tend to buy 270g pack size most and 220 the least. Let's check the brand who sells 270g size chips.

```
Merged_Data[PACK_SIZE_g==270,unique(PROD_NAME)]
```

```
## [1] "Twisties Cheese     270g" "Twisties Chicken270g"
```

The brand which sells 270g pack size is Twisties Cheese.

Conclusion

Most sales is coming from Budget-Mid-age single/couples,then Mainstream-young single/couples. Highest number of customers are in Mainstream-Young Single/Couples segment.

Mainstream-Young single couple & Mid-age single/couples are tend to spend more per unit.

mainstream young single couple segment are buying most chips compared to other segment, this segment prefers to buy TYRRELLS BRAND chips and 270 g pack size which is sold by only one brand Twisties. Recommendation- Category Manager can focus more on TYRRELLS chips as Mainstream-young single/couples are tend to buy this chips by increasing the visibility of the product to attract customers of this segment.