Sprocket Central Pty Ltd Data Set Data Inspection

Dear Customer your datasets were well received and an inspection done as attached. There are codes contained that explains each process. Thank you

```r
library(tidyverse)
library(lubridate)
library(scales) ## for scales
library(VIM)  ## aggregate plotting of missing values
library(utf8)
```

```r
Transactions <- read_csv("Transactions.csv")
NewCustomerList <- read_csv("NewCustomerList.csv")
CustomerDemographic <- read_csv("CustomerDemographic.csv")
CustomerAddress <- read_csv("CustomerAddress.csv")
```

load the data and clean the names, dplyr::select useful columns

1 Transactions

```r
str(Transactions)
```

```
## spc_tbl_ [20,000 x 13] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ transaction_id         : num [1:20000] 1 2 3 4 5 6 7 8 9 10 ...
##  $ product_id             : num [1:20000] 2 3 37 88 78 25 22 15 67 12 ...
##  $ customer_id            : num [1:20000] 2950 3120 402 3135 787 ...
##  $ transaction_date       : chr [1:20000] "2/25/2017" "5/21/2017" "10/16/2017" "8/31/2017" ...
##  $ online_order           : num [1:20000] 0 1 0 0 1 1 1 0 0 1 ...
##  $ order_status           : chr [1:20000] "Approved" "Approved" "Approved" "Approved" ...
##  $ brand                  : chr [1:20000] "Solex" "Trek Bicycles" "OHM Cycles" "Norco Bicycles" ...
##  $ product_line           : chr [1:20000] "Standard" "Standard" "Standard" "Standard" ...
##  $ product_class          : chr [1:20000] "medium" "medium" "low" "medium" ...
##  $ product_size           : chr [1:20000] "medium" "large" "medium" "medium" ...
##  $ list_price             : num [1:20000] 71.5 2091.5 1793.4 1198.5 1765.3 ...
##  $ standard_cost          : chr [1:20000] "$53.62" "$388.92" "$248.82" "$381.10" ...
##  $ product_first_sold_date: num [1:20000] 41245 41701 36361 36145 42226 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   transaction_id = col_double(),
##   ..   product_id = col_double(),
##   ..   customer_id = col_double(),
##   ..   transaction_date = col_character(),
##   ..   online_order = col_double(),
##   ..   order_status = col_character(),
##   ..   brand = col_character(),
##   ..   product_line = col_character(),
##   ..   product_class = col_character(),
##   ..   product_size = col_character(),
##   ..   list_price = col_double(),
##   ..   standard_cost = col_character(),
##   ..   product_first_sold_date = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Each transaction should be unique, therefore the transaction id should be unique for all transactions

```r
n_distinct(Transactions$transaction_id)
```

```
## [1] 20000
```

All the transaction_id are distinct, implying all transactions are unique that is there aren't any duplicate transactions.

missing values

```r
sum(is.na(Transactions))
```

```
## [1] 1542
```

There are 1542 missing values in Transactions data

columns with missing values

```r
names(which(colSums(is.na(Transactions)) > 0))
```

```
## [1] "online_order"            "brand"
## [3] "product_line"            "product_class"
## [5] "product_size"            "standard_cost"
## [7] "product_first_sold_date"
```

7 columns have missing values while 6 do not have any missing value.

sum of missing values per column

```r
map(Transactions, ~ sum(is.na(.)))
```

```
## $transaction_id
## [1] 0
##
## $product_id
## [1] 0
##
## $customer_id
## [1] 0
##
## $transaction_date
## [1] 0
##
## $online_order
## [1] 360
##
## $order_status
## [1] 0
##
## $brand
## [1] 197
##
## $product_line
## [1] 197
##
## $product_class
## [1] 197
##
## $product_size
## [1] 197
##
## $list_price
## [1] 0
##
## $standard_cost
## [1] 197
##
## $product_first_sold_date
## [1] 197
```

online_order

```r
Transactions %>% count(online_order, sort = T)
```

```
## # A tibble: 3 x 2
##   online_order     n
##          <dbl> <int>
## 1            1  9829
## 2            0  9811
## 3           NA   360
```

online_order would be for whether the product was ordered online or not where 0 stands for no and 1 for yes, thus NAs should not be present in this column.

brand

```
Transactions %>% count(brand, sort = T)
```

```
## # A tibble: 7 x 2
##   brand               n
##   <chr>           <int>
## 1 Solex            4253
## 2 Giant Bicycles   3312
## 3 WeareA2B         3295
## 4 OHM Cycles       3043
## 5 Trek Bicycles    2990
## 6 Norco Bicycles   2910
## 7 <NA>              197
```

brand should also not have missing values

product_line

```
Transactions %>% count(product_line, sort = T)
```

```
## # A tibble: 5 x 2
##   product_line     n
##   <chr>        <int>
## 1 Standard     14176
## 2 Road          3970
## 3 Touring       1234
## 4 Mountain       423
## 5 <NA>           197
```

product_line should also not have missing values

product_class

```
Transactions %>% count(product_class, sort = T)
```

```
## # A tibble: 4 x 2
##   product_class     n
##   <chr>         <int>
## 1 medium        13826
## 2 high           3013
## 3 low            2964
## 4 <NA>            197
```

product__class should be three classes without missing values

product__size

```
Transactions %>% count(product_size, sort = T)
```

```
## # A tibble: 4 x 2
##   product_size     n
##   <chr>        <int>
## 1 medium       12990
## 2 large         3976
## 3 small         2837
## 4 <NA>           197
```

product__class should be three classes without missing values

standard__cost

Here some standard cost have dollar sign, therefore we can remove the dollar sign and remove any white space then convert to numeric, then round to 2 decimal places.

First we create a duplicate of the data

```
Transactions_2 <- Transactions
Transactions_2$standard_cost <- gsub("[\\$,]", "", Transactions_2$standard_cost)
Transactions_2$standard_cost <- str_squish(Transactions_2$standard_cost)
Transactions_2$standard_cost <- as.numeric(Transactions_2$standard_cost)
Transactions_2$standard_cost <- round(Transactions_2$standard_cost, digits = 2)
```

```
Transactions_2 %>% count(standard_cost, sort = T)
```

```
## # A tibble: 101 x 2
##    standard_cost     n
##            <dbl> <int>
##  1         389.    465
##  2         955.    396
##  3          53.6   274
##  4         162.    235
##  5         260.    233
##  6         677.    232
##  7         934.    229
##  8         599.    228
##  9         508.    226
## 10         460.    223
## # i 91 more rows
```

```
summary(Transactions_2$standard_cost)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    7.21  215.14  507.58  556.05  795.10 1759.85     197
```

product first sold date should be a date but its in numeric

Thus convert it to date but first we create another duplicate data of Transactions_2

Excel is said to use 1900-01-01 as day 1 (Windows default)

```
Transactions_3 <- Transactions_2
Transactions_3$product_first_sold_date <- as.Date(
  Transactions_3$product_first_sold_date, origin = "1900-01-01"
)
Transactions_3 %>% count(product_first_sold_date, sort = T)
```

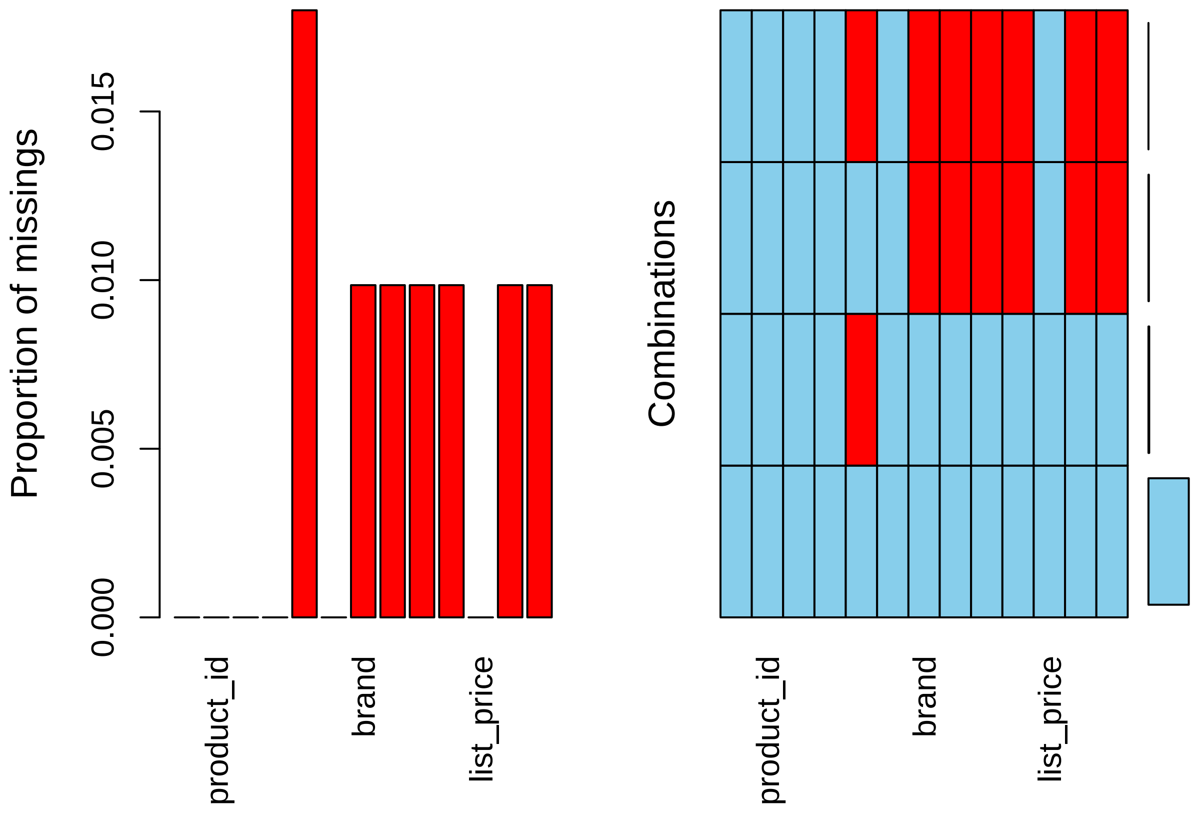
```
## # A tibble: 101 x 2
##    product_first_sold_date     n
##    <date>                  <int>
##  1 1992-10-04                234
##  2 2012-06-06                229
##  3 2003-07-23                227
##  4 2009-03-10                222
##  5 2004-08-19                220
##  6 2005-05-12                219
##  7 2010-06-09                218
##  8 2016-07-11                216
##  9 2004-01-18                215
## 10 2012-04-12                215
## # i 91 more rows
```

It is seen that brand, product_line, product_class, product_size, standard_cost and product_first_sold_date all have the same number of missing values.

Could they be the same product?

missing values plot proportions and combinations

```
aggr(Transactions_3)
```



From the plot, it is seen that online_order has the highest proportion of missing values while brand, product_line, product_class, product_size, standard_cost and product_first_sold_date have the same combinations of missing values.

we can filter with any of the products that have the same number of missing values because the combinations are the same but we can also filter with online_order because it will capture all missing values in the other variables too.

transactions when online_order is missing

```r
tran_online_order_miss <- Transactions_3 %>% filter(is.na(online_order) > 0)
dim(tran_online_order_miss)
```

```
## [1] 360  13
```

The transactions where online_order are missing are 360 transactions

Where only online order are missing

```r
tran_miss_online_order_only <- Transactions_3 %>% filter(is.na(online_order) > 0
                                                         & !is.na(brand))
dim(tran_miss_online_order_only)
```

```
## [1] 358  13
```

```
tran_miss_online_order_only %>% count(order_status, sort = T)
```

```
## # A tibble: 2 x 2
##   order_status     n
##   <chr>        <int>
## 1 Approved       352
## 2 Cancelled        6
```

There are 358 transactions where only the online_order are missing. 352 orders were approved while 6 were cancelled.

Data of missing values from the variables that are not online_order

```
tran_missing_brand <- Transactions_3 %>% filter(is.na(brand) > 0 &
                                                  !is.na(online_order))
```

```
dim(tran_missing_brand)
```

```
## [1] 195  13
```

There are 195 transactions that have the other variables as missing values but online_order isn't missing

Their product_id

```
tran_missing_brand %>% count(product_id, sort = T)
```

```
## # A tibble: 1 x 2
##   product_id     n
##        <dbl> <int>
## 1          0   195
```

These transactions have their product_id as 0

```
tran_missing_brand %>% count(order_status, sort = T)
```

```
## # A tibble: 2 x 2
##   order_status     n
##   <chr>        <int>
## 1 Approved       194
## 2 Cancelled        1
```

These products have never been sold before. One order was cancelled

Missing online_order and the other variables too

```
tran_miss_all_7 <- Transactions_3 %>% filter(is.na(online_order) > 0
                                               & is.na(brand) > 0)
dim(tran_miss_all_7)
```

```
## [1]  2 13
```

These transactions were only two. They were both approved.

0 product_id

```
tran_productid0 <- Transactions_3 %>% filter(product_id == 0)
dim(tran_productid0)
```

```
## [1] 1378   13
```

```


tran_productid0 %>% count(order_status, sort = T)
```

The transactions are 1378

```
## # A tibble: 2 x 2
##   order_status     n
##   <chr>        <int>
## 1 Approved      1371
## 2 Cancelled        7
```

most of the orders were approved

It can be concluded that products that did not have first sale date did not have product id, brand name, whether they were ordered online the product_line, product_class, product_size and standard_cost.

These transactions existed and thus the missing values were entry mistakes.

column summary

```
summary(Transactions_3$product_id)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   18.00   44.00   45.36   72.00  100.00
```

```
summary(Transactions_3$customer_id)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##     1.0   857.8  1736.0  1738.2  2613.0  5034.0
```

```
summary(Transactions_3$online_order)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##  0.0000  0.0000  1.0000  0.5005  1.0000  1.0000     360
```

```
summary(Transactions_3$list_price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   12.01  575.27 1163.89 1107.83 1635.30 2091.47
```

Convert transaction_date to date

```
Transactions_3 <- Transactions_3 %>% mutate(transaction_date
                                            = mdy(transaction_date))
```

Is the data up to date

Top 10 from the latest date

```
head(Transactions_3 %>% dplyr::select(transaction_date)
     %>% arrange(desc(transaction_date)), 10)
```

```
## # A tibble: 10 x 1
##    transaction_date
##    <date>          
##  1 2017-12-30      
##  2 2017-12-30      
##  3 2017-12-30      
##  4 2017-12-30      
##  5 2017-12-30      
##  6 2017-12-30      
##  7 2017-12-30      
##  8 2017-12-30      
##  9 2017-12-30      
## 10 2017-12-30
```

Bottom 10 from the first date

```
tail(Transactions_3 %>% dplyr::select(transaction_date)
     %>% arrange(desc(transaction_date)), 10)
```

```
## # A tibble: 10 x 1
##    transaction_date
##    <date>
##  1 2017-01-01
##  2 2017-01-01
##  3 2017-01-01
##  4 2017-01-01
##  5 2017-01-01
##  6 2017-01-01
##  7 2017-01-01
##  8 2017-01-01
##  9 2017-01-01
## 10 2017-01-01
```

The records were from 1st January 2017 to 30th December 2017

2 CUSTOMER DEMOCRAPHIC

```
str(CustomerDemographic)
```

```
## spc_tbl_ [4,000 x 13] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ customer_id                        : num [1:4000] 1 2 3 4 5 6 7 8 9 10 ...
##  $ first_name                         : chr [1:4000] "Laraine" "Eli" "Arlin" "Talbot" ...
##  $ last_name                          : chr [1:4000] "Medendorp" "Bockman" "Dearle" NA ...
##  $ gender                             : chr [1:4000] "F" "Male" "Male" "Male" ...
##  $ past_3_years_bike_related_purchases: num [1:4000] 93 81 61 33 56 35 6 31 97 49 ...
##  $ DOB                                : Date[1:4000], format: "1953-10-12" "1980-12-16" ...
##  $ job_title                          : chr [1:4000] "Executive Secretary" "Administrative Officer"
##  $ job_industry_category              : chr [1:4000] "Health" "Financial Services" "Property" "IT" .
##  $ wealth_segment                     : chr [1:4000] "Mass Customer" "Mass Customer" "Mass Customer"
##  $ deceased_indicator                 : chr [1:4000] "N" "N" "N" "N" ...
##  $ default                            : chr [1:4000] "\"'" "<script>alert('hi')</script>" "1-Feb" "(
##  $ owns_car                           : chr [1:4000] "Yes" "Yes" "Yes" "No" ...
##  $ tenure                             : num [1:4000] 11 16 15 7 8 13 11 7 8 20 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   customer_id = col_double(),
##   ..   first_name = col_character(),
##   ..   last_name = col_character(),
##   ..   gender = col_character(),
##   ..   past_3_years_bike_related_purchases = col_double(),
##   ..   DOB = col_date(format = ""),
##   ..   job_title = col_character(),
##   ..   job_industry_category = col_character(),
##   ..   wealth_segment = col_character(),
##   ..   deceased_indicator = col_character(),
##   ..   default = col_character(),
##   ..   owns_car = col_character(),
##   ..   tenure = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

The data has 4000 observations with 13 variables

```
n_distinct(CustomerDemographic$customer_id)
```

```
## [1] 4000
```

The column customer_id has 4000 unique entries implying the data has the unique demographics of each customer.

```
class(CustomerDemographic$customer_id)
```

```
## [1] "numeric"
```

The entries are 1 to 4000

Missing values

```
sum(is.na(CustomerDemographic))
```

```
## [1] 1045
```

There are 1045 missing values

columns with missing values

```
names(which(colSums(is.na(CustomerDemographic)) > 0))
```

```
## [1] "last_name" "DOB"       "job_title" "default"   "tenure"
```

5 columns have missing values

Technically Gender should be Male or Female

```
CustomerDemographic %>% count(gender, sort = T)
```

```
## # A tibble: 6 x 2
##   gender     n
##   <chr>  <int>
## 1 Female  2037
## 2 Male    1872
## 3 U         88
## 4 F          1
## 5 Femal      1
## 6 M          1
```

In gender we have some coding problems as we have a mix of M and Male meaning the same thing and F, Femal and Female meaning Female. We can encode with M for Male, F for Female and U for Unidentified.

```
CustomerDemographic <- CustomerDemographic %>%
    mutate(gender = case_when(
        str_detect(gender, "^F") ~ "Female",
        str_detect(gender, "^M") ~ "Male",
        str_detect(gender, "^U") ~ "Unidentified",
        TRUE ~ gender
    ))
```

Checking gender

```
CustomerDemographic %>% count(gender, sort = T)
```

```
## # A tibble: 3 x 2
##   gender           n
##   <chr>        <int>
## 1 Female        2039
## 2 Male          1873
## 3 Unidentified    88
```

Thus we have three genders-Female, Male and Unidentified.

Create a copy of the dataset

```
CustomerDemographic_2 <- CustomerDemographic
```

past 3 years related bike purchases

```
class(CustomerDemographic_2$past_3_years_bike_related_purchases)
```

```
## [1] "numeric"
```

it should be numeric, thus we can get a summary

```
summary(CustomerDemographic_2$past_3_years_bike_related_purchases)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   24.00   48.00   48.89   73.00   99.00
```

Date of Birth

```r
class(CustomerDemographic_2$DOB)
```

```
## [1] "Date"
```

It's a date and has missing values

Check the years of each individual as at the year 2023

Checking the years of customers

```r
CustomerDemographic_2 <- CustomerDemographic_2 %>% mutate(
  age = trunc((DOB %--% today())/ years(1))
)

CustomerDemographic_2 <- CustomerDemographic_2 %>% dplyr::select(1:6, 14, 7:13)
```

summary of age

```r
summary(CustomerDemographic_2$age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   21.00   36.00   46.00   45.72   55.00  179.00      87
```

we have NAs and someone who is 179 years

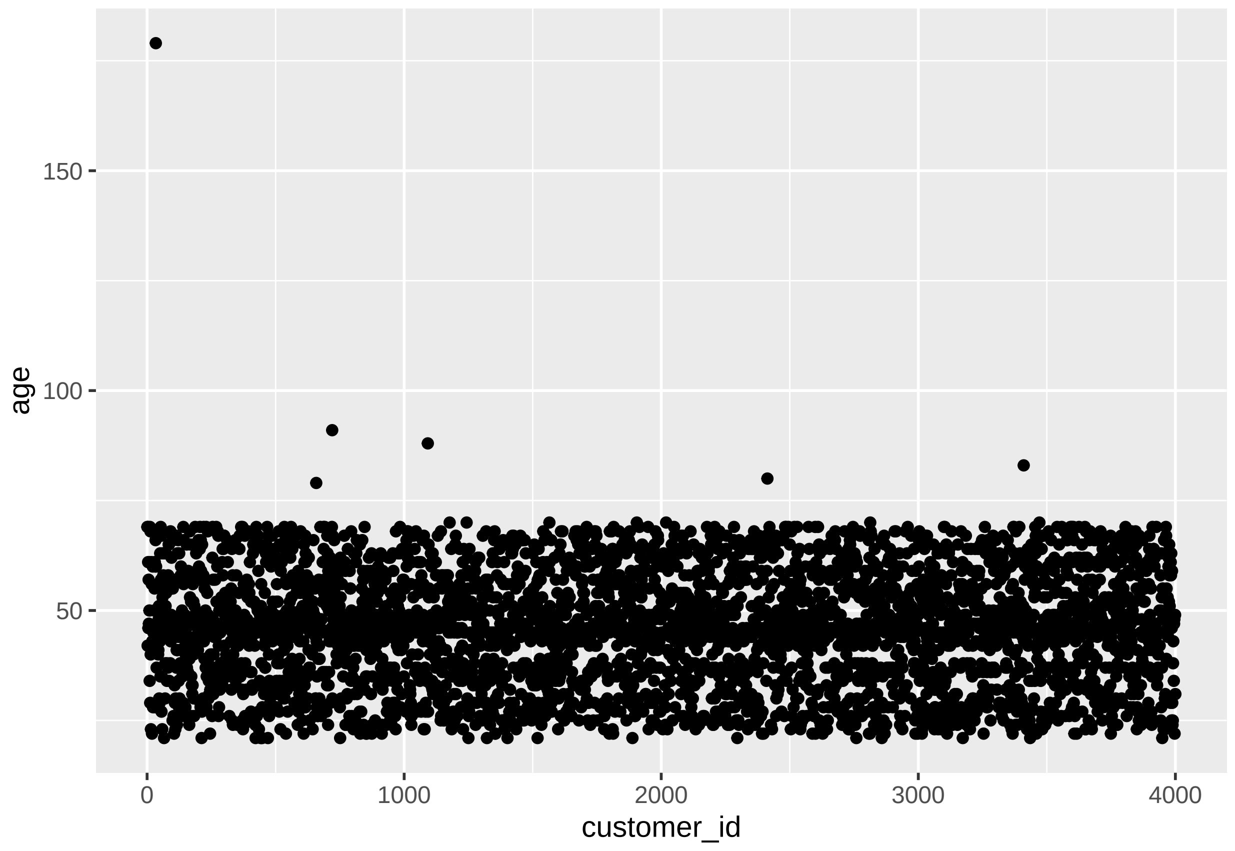
the oldest 10 people

```r
head(CustomerDemographic_2 %>% dplyr::select(age) %>% arrange(desc(age)), 10)
```

```
## # A tibble: 10 x 1
##      age
##    <dbl>
##  1   179
##  2    91
##  3    88
##  4    83
##  5    80
##  6    79
##  7    70
##  8    70
##  9    70
## 10    70
```

Point plot

```r
ggplot(CustomerDemographic_2, aes(customer_id, age)) +
  geom_point()
```



It is only age 179 years that is an oulier

There are 87 NAs

Missing values in DOB

```r
sum(is.na(CustomerDemographic_2$DOB))
```

```
## [1] 87
```

There are also 87 missing DOB values.

The date of 1843-12-21 can be converted to 1943-12-21

```r
CustomerDemographic_3 <- CustomerDemographic_2
CustomerDemographic_3 <- CustomerDemographic_3[, -7]
CustomerDemographic_3$DOB[CustomerDemographic_3$DOB == "1843-12-21"] <- "1943-12-21"
CustomerDemographic_3 <- CustomerDemographic_3 %>% mutate(
  age = trunc((DOB %--% today())/ years(1))
)
CustomerDemographic_3 <- CustomerDemographic_3 %>% dplyr::select(1:6, 14, 7:13)
```

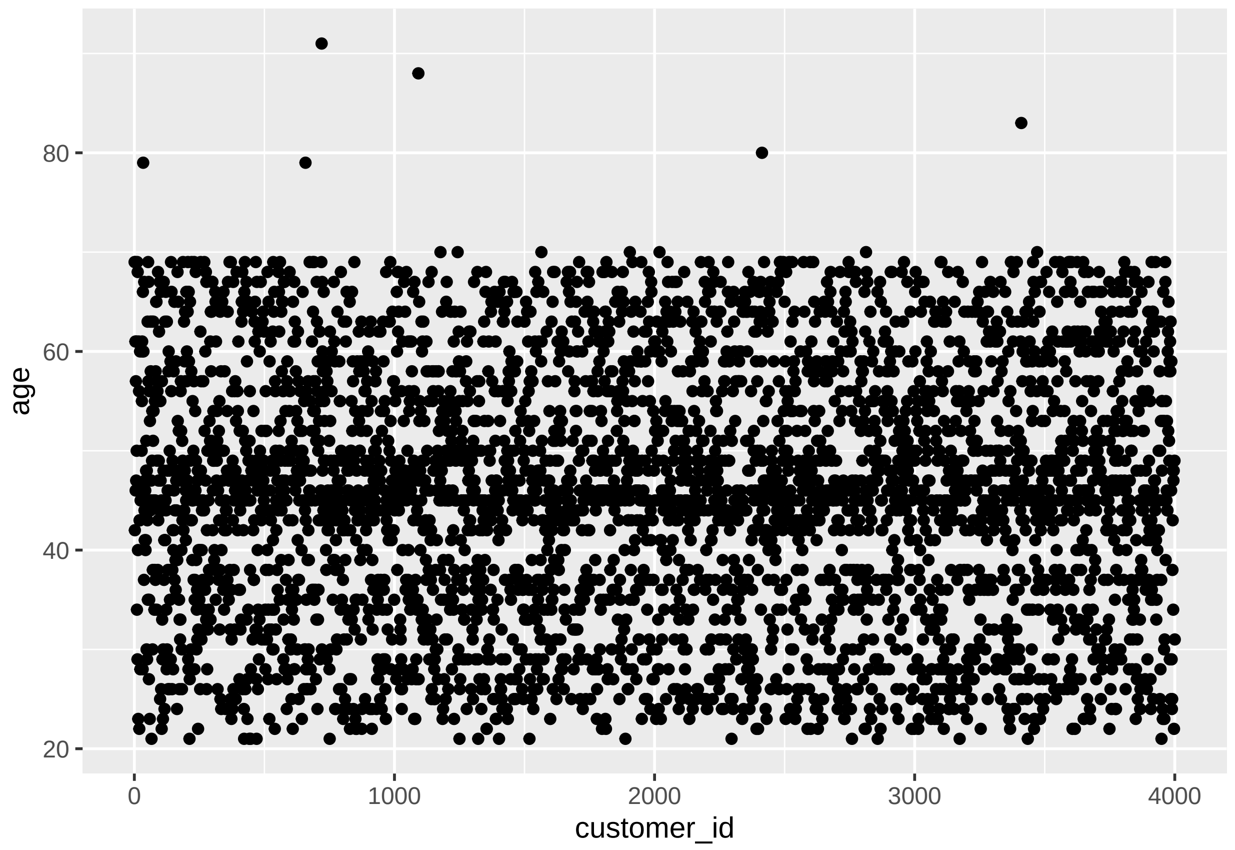
checking after correcting

```
head(CustomerDemographic_3 %>% dplyr::select(age) %>% arrange(desc(age)), 10)
```

```
## # A tibble: 10 x 1
##      age
##    <dbl>
##  1    91
##  2    88
##  3    83
##  4    80
##  5    79
##  6    79
##  7    70
##  8    70
##  9    70
## 10    70
```

Point plot

```
ggplot(CustomerDemographic_3, aes(customer_id, age)) +
  geom_point()
```



Job Title

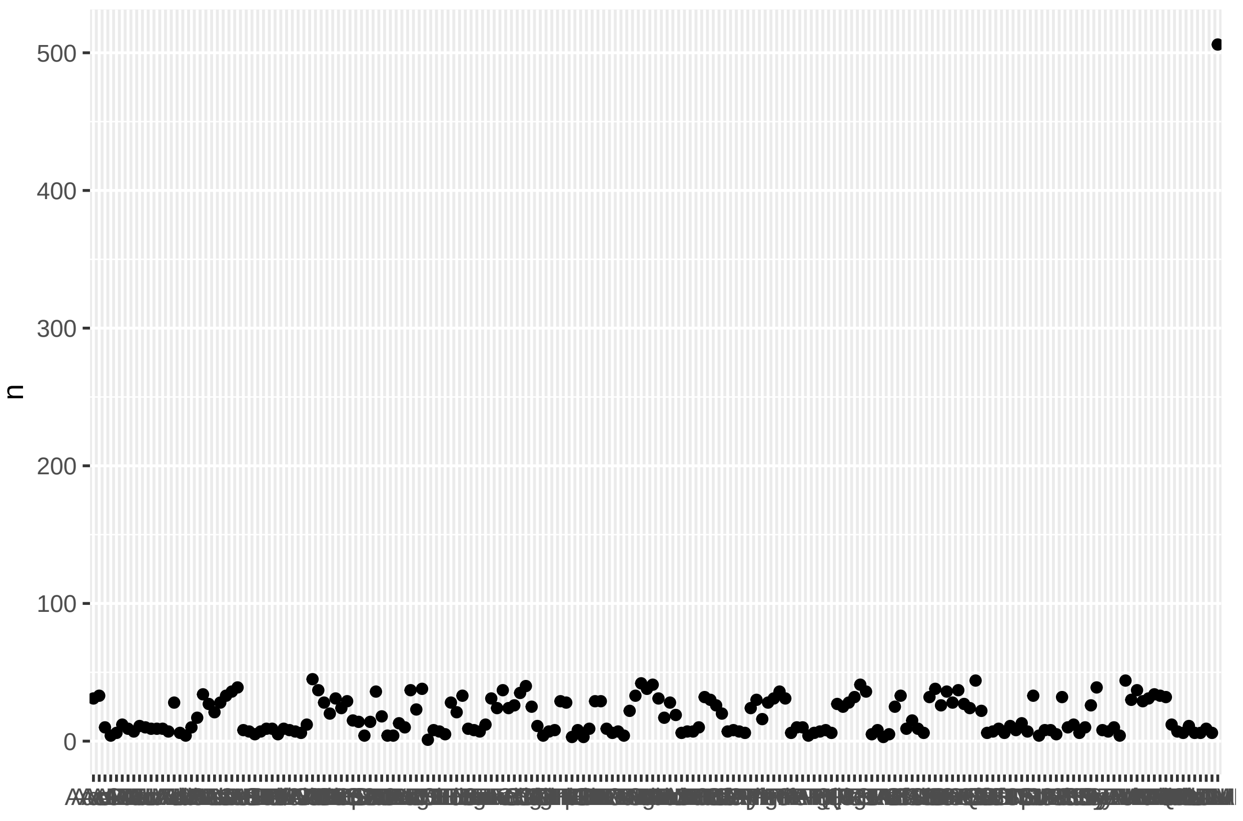
```r
head(CustomerDemographic_3 %>% count(job_title, sort = T), 10)
```

```
## # A tibble: 10 x 2
##    job_title                                n
##    <chr>                                <int>
##  1 <NA>                                   506
##  2 Business Systems Development Analyst    45
##  3 Social Worker                           44
##  4 Tax Accountant                          44
##  5 Internal Auditor                        42
##  6 Legal Assistant                         41
##  7 Recruiting Manager                      41
##  8 General Manager                         40
##  9 Associate Professor                     39
## 10 Structural Engineer                     39
```

```r
tail(CustomerDemographic_3 %>% count(job_title, sort = T), 10)
```

```
## # A tibble: 10 x 2
##    job_title                        n
##    <chr>                        <int>
##  1 Database Administrator II        4
##  2 Geologist II                   4
##  3 Human Resources Assistant IV   4
##  4 Programmer Analyst IV          4
##  5 Staff Accountant I             4
##  6 Systems Administrator IV       4
##  7 Health Coach I                 3
##  8 Health Coach III               3
##  9 Research Assistant III         3
## 10 Developer I                    1
```

```
CustomerDemographic_3 %>% count(job_title) %>%
  ggplot(aes(job_title, n)) +
  geom_point() +
  scale_x_discrete(labels = abbreviate)+
  labs(x = "")
```



Job industry category

```
CustomerDemographic_3 %>% count(job_industry_category, sort = T)
```

```
## # A tibble: 10 x 2
##    job_industry_category     n
##    <chr>                 <int>
##  1 Manufacturing           799
##  2 Financial Services      774
##  3 n/a                        656
##  4 Health                     602
##  5 Retail                     358
##  6 Property                   267
##  7 IT                         223
##  8 Entertainment              136
##  9 Argiculture                113
## 10 Telecommunications          72
```

The Categories are only 10.

We have a category encoded as n/a ranking as number 3 in count which probably was to be "Not Available"

```
CustomerDemographic_3$job_industry_category[CustomerDemographic_3$job_industry_category == "n/a"] <- "N
```

```
CustomerDemographic_3 %>% count(job_industry_category, sort = T)
```

```
## # A tibble: 10 x 2
##    job_industry_category     n
##    <chr>                 <int>
##  1 Manufacturing           799
##  2 Financial Services      774
##  3 Not Available           656
##  4 Health                  602
##  5 Retail                  358
##  6 Property                267
##  7 IT                      223
##  8 Entertainment           136
##  9 Argiculture             113
## 10 Telecommunications       72
```

Wealth Segment

```
class(CustomerDemographic_3$wealth_segment)
```

```
## [1] "character"
```

```
CustomerDemographic_3 %>% count(wealth_segment, sort = T)
```

```
## # A tibble: 3 x 2
##   wealth_segment        n
##   <chr>             <int>
## 1 Mass Customer      2000
## 2 High Net Worth     1021
## 3 Affluent Customer   979
```

Wealth Segment has three categories and the column is rightly tabulated

Deceased Indicator

```r
class(CustomerDemographic_3$deceased_indicator)
```

```
## [1] "character"
```

```r
CustomerDemographic_3 %>% count(deceased_indicator, sort = T)
```

```
## # A tibble: 2 x 2
##   deceased_indicator     n
##   <chr>              <int>
## 1 N                   3998
## 2 Y                      2
```

Deceased indicator has two indicators, column is rightly tabulated

column indicated as default

```r
class(CustomerDemographic_3$default)
```

```
## [1] "character"
```

```r
customers_default <- CustomerDemographic_3 %>% count(default, sort = T)
```

```r
CustomerDemographic_3$default <- utf8_encode(CustomerDemographic_3$default)
```

Column default does not contain useful information on customer demograhic and NAs are the largest by count

NAs are the largest job title by far which suggests that by the number of missing data then the variable is not necessary because customers are not willing to give out such information or maybe they are just not employed. The column should be left out. The job_industry_category can be considered sufficient since it has all the information.

We can remove the job_title and default category

```r
CustomerDemographic_4 <- CustomerDemographic_3 %>%
  dplyr::select(-job_title, -default)
```

owns a car

```r
class(CustomerDemographic_4$owns_car)
```

```
## [1] "character"
```

```r
CustomerDemographic_4 %>% count(owns_car, sort = T)
```

```
## # A tibble: 2 x 2
##   owns_car     n
##   <chr>    <int>
## 1 Yes       2024
## 2 No        1976
```

Has two options-Yes or No. Data correctly tabulated

tenure

```r
class(CustomerDemographic_4$tenure)
```

```
## [1] "numeric"
```

```r
summary(CustomerDemographic_4$tenure)
```

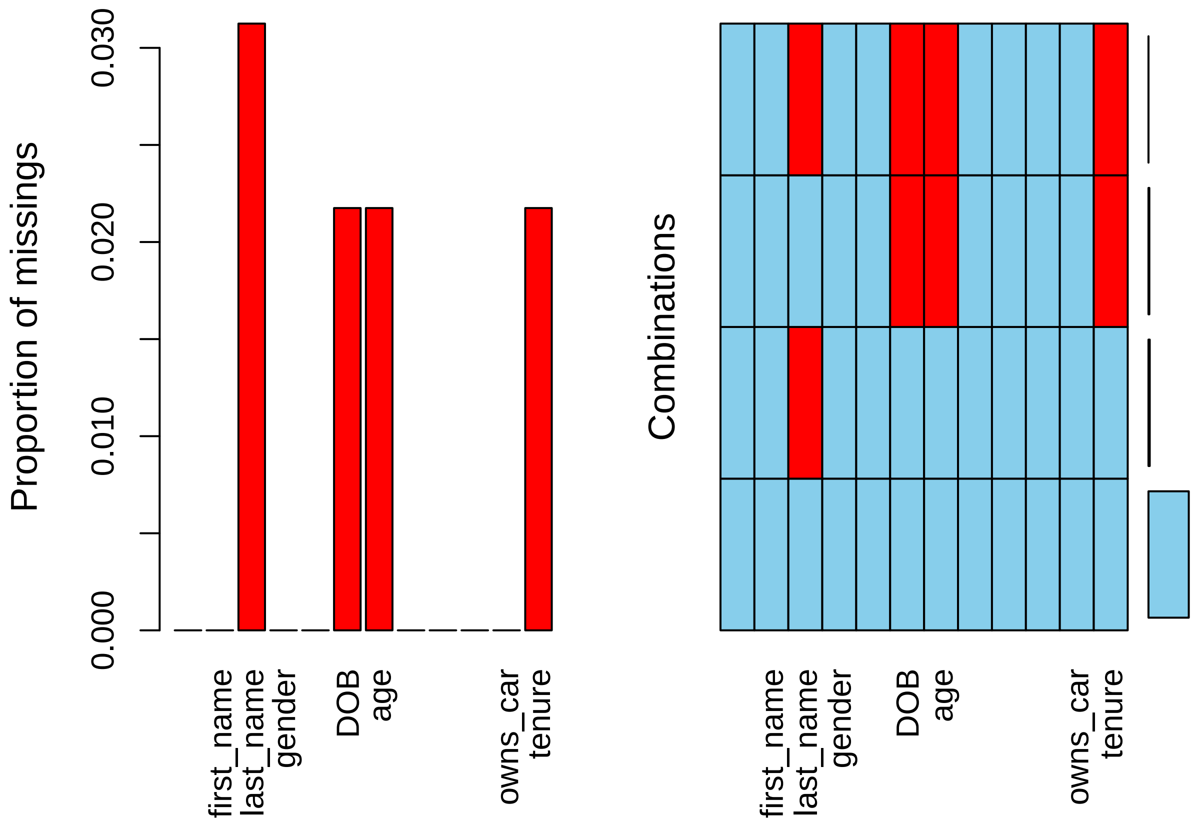
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##    1.00    6.00   11.00   10.66   15.00   22.00      87
```

```r
CustomerDemographic_4 %>% count(tenure, sort = T)
```

```
## # A tibble: 23 x 2
##    tenure     n
##     <dbl> <int>
##  1      7   235
##  2      5   228
##  3     11   221
##  4     10   218
##  5     16   215
##  6      8   211
##  7     18   208
##  8     12   202
##  9      9   200
## 10     14   200
## # i 13 more rows
```

tenure is numeric and has 87 NAs

```r
aggr(CustomerDemographic_4)
```



Date of birth and tenure have the same combination of missing values. Since age is an extraction of date of birth, it is expected that their missing values would be the same.

missing values for date of birth

```
customer_no_dob <- CustomerDemographic_4 %>% filter(is.na(DOB) > 0)
dim(customer_no_dob)
```

```
## [1] 87 12
```

missing values for tenure

```
customer_no_tenure <- CustomerDemographic_4 %>% filter(is.na(tenure) > 0)
dim(customer_no_tenure)
```

```
## [1] 87 12
```

compare the two with customer_id, filter customer id that is in tenure

```
equality <- customer_no_dob %>% filter(customer_id == customer_no_tenure$customer_id)
dim(equality)
```

```
## [1] 87 12
```

CUSTOMER ADRESS

Dimensions

```r
dim(CustomerAddress)
```

```
## [1] 3999    6
```

The dataset has 3999 observations and 6 variables

The uniqueness of the dataset;

```r

n_distinct(CustomerAddress$customer_id)
```

Customer id

```
## [1] 3999
```

All the 3999 observations are unique

```r
summary(CustomerAddress$customer_id)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##       1    1004    2004    2004    3004    4003
```

The maximum value assigned to customer id is 4003 while the minimum is 1. The mean value of numeric number from 1 to 3999 is 2000 while the mean value of customer id is 2004. It therefore implies that there were missing intergers between 1 to 3999.

missing values-customer id

```r
sum(is.na(CustomerAddress$customer_id))
```

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

customer id has no missing values

```r
customer_id_notseen <- setdiff(1:4003, CustomerAddress$customer_id)
customer_id_notseen
```

```
## [1]  3 10 22 23
```

Integers 3, 10, 22 and 23 were not assigned to any customer as customer id

Missing values

```
sum(is.na(CustomerAddress))
```

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

All the observations have cell values for all the columns

CustomerAdress-adress

We have adress with a combination of digits and characters and the next column is the post-code, it is therefore unnecessary to have the leading digits in address

Remove leading digits and then remove any extra space

First create a duplicate

```
CustomerAddress_2 <- CustomerAddress
CustomerAddress_2$address <- gsub("^[0-9]+", "", CustomerAddress_2$address)
CustomerAddress_2$address <- str_squish(CustomerAddress_2$address)
```

Address count

```
Customer_addresscount <- CustomerAddress_2 %>% count(address, sort = T)
dim(Customer_addresscount)
```

```
## [1] 3291    2
```

There are 3291 different locations.

CustomerAddress-postcode

```
class(CustomerAddress$postcode)
```

```
## [1] "numeric"
```

The column is loaded as numeric

count

```
Customer_postcodecount <- CustomerAddress_2 %>% count(postcode, sort = T)
dim(Customer_postcodecount)
```

```
## [1] 873   2
```

There were some shared postcodes

summary-postcode

```
summary(CustomerAddress_2$postcode)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2000    2200    2768    2986    3750    4883
```

CustomerAddress-state

```
class(CustomerAddress_2$state)
```

```
## [1] "character"
```

state-count

```
CustomerAddress_2 %>% count(state, sort = T)
```

```
## # A tibble: 5 x 2
##   state               n
##   <chr>           <int>
## 1 NSW              2054
## 2 VIC               939
## 3 QLD               838
## 4 New South Wales    86
## 5 Victoria           82
```

We have states encoded as NSW and New South Wales which probably mean the same thing and we have Victoria and VIC. The encoding can be corrected.

We create another duplicate

```
CustomerAddress_3 <- CustomerAddress_2 %>%
    mutate(state = case_when(
        str_detect(state, "New") ~ "NSW",
        str_detect(state, "Vict") ~ "VIC",
        TRUE ~ state
    ))
```

state-count

```
CustomerAddress_3 %>% count(state, sort = T)
```

```
## # A tibble: 3 x 2
##   state     n
##   <chr> <int>
## 1 NSW    2140
## 2 VIC    1021
## 3 QLD     838
```

The customers were from 3 different states

CustomerAddress-country

```
class(CustomerAddress_3$country)
```

```
## [1] "character"
```

Was loaded as character

CustomerAddress-country count

```
CustomerAddress_3 %>% count(country, sort = T)
```

```
## # A tibble: 1 x 2
##   country       n
##   <chr>     <int>
## 1 Australia  3999
```

All customers are from Australia.

The column is not necessary as all the states are also from Australia but the column has no harm even though it will be left out in analysis.

CustomerAdress-property valuation

```
class(CustomerAddress_3$property_valuation)
```

```
## [1] "numeric"
```

Its numeric

property valuation-summary

```
summary(CustomerAddress_3$property_valuation)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   6.000   8.000   7.514  10.000  12.000
```

Minimum valuation is 1 and the maximum valuation is 12

What digits does property valuation contain

```
customervaluation <- setdiff(1:12, CustomerAddress_3$property_valuation)
customervaluation
```

```
## integer(0)
```

The digits are between 1 and 12.

4 NEWCUSTOMER LIST

parameters

```
dim(NewCustomerList)
```

```
## [1] 1000   23
```

NewCustomerList does not have a unique customer identifier that is customer_id and the column 17 to 23 were calculations involving the values of other columns.

- For the Transactions dataset;
  - All the transaction_id are distinct, implying all transactions are unique.
  - There are 1542 missing values.
  - 7 columns have missing values while 6 do not have any missing value. online_order, brand, product_line, product_class, product_size, standard_cost and product_first_sold_date have missing values.
  - It is seen that brand, product_line, product_class, product_size, standard_cost and product_first_sold_date all have the same number of missing values.
  - online_order would be for whether the product was ordered online or not where 0 stands for no and 1 for yes, thus NAs should not be present in this column.
  - brand should also not have missing values.
  - product_line should also not have missing values
  - product_class should be three classes without missing values
  - Some standard cost have dollar sign, therefore we can remove the dollar sign and remove any white space then convert to numeric, then round to 2 decimal places.
  - product first sold date should be a date but its in numeric.
  - There are 195 transactions that have brand, product_line, product_class, product_size, standard_cost and product_first_sold_date as missing values but online_order isn't missing. These transactions have their product_id as 0.
  - The records were from 1st January 2017 to 30th December 2017.
- For Customer Demograhic dataset;
  - The data has 4000 observations with 13 variables
  - The column customer_id has 4000 unique entries implying the data has the unique demographics of each customer. The entries are 1 to 4000
  - There are 1045 missing values
  - 5 columns have missing values

  - Technically Gender should be Male or Female and another for other categories.
  - In gender we have some coding problems as we have a mix of M and Male meaning the same thing and F, Femal and Female meaning Female. We can encode with M for Male, F for Female and U for Unidentified.
  - It's a date and has missing values which is allowed since some customers would not give out such information but that should be indicated with another response.
  - In calculating years from the DOB column, we have NAs and someone who is 179 years. It is only age 179 years that is an oulier from DOB of 1843-12-21 customer id 34. The company can have an age column to always use to counter check the date of birth column.
  - Job Industry Categories are only 10. We have a category encoded as n/a ranking as number 3 in count which probably was to be "Not Available".
  - NAs are the largest job title by far which suggests that by the number of missing data then the variable is not necessary because customers are not willing to give out such information or maybe they are just not employed. The column should be left out. The job_industry_category can be considered sufficient since it has all the information.
  - Wealth Segment has three categories and the column is rightly tabulated.
  - Deceased indicator has two indicators, column is rightly tabulated
  - Column default does not contain useful information on customer demograhic and NAs are the largest by count and it is text encoding that is not readable.
  - owns car has two options-Yes or No. Data correctly tabulated.
  - tenure is numeric and has 87 NAs.
  - Date of birth and tenure have the same combination of missing values. Since age is an extraction of date of birth, it is expected that their missing values would be the same.

- For Customer Address
  - The dataset has 3999 observations and 6 variables.
  - All the 3999 observations are unique as per the customer id.
  - The maximum value assigned to customer id is 4003 while the minimum is 1. The mean value of numeric number from 1 to 3999 is 2000 while the mean value of customer id is 2004. It therefore implies that there were missing intergers between 1 to 3999. Integers 3, 10, 22 and 23 were not assigned to any customer as customer id.
  - All the observations have cell values for all the columns.
  - We have adress with a combination of digits and characters and the next column is the postcode, it is therefore unnecessary to have the leading digits in address.
  - We have states encoded as NSW and New South Wales which probably mean the same thing and we have Victoria and VIC. The encoding can be corrected. The customers were from 3 different states
  - All customers are from Australia. The column is not necessary as all the states are also from Australia but the column has no harm even though it will be left out in analysis.

- For New Customer List
  - NewCustomerList does not have a unique customer identifier that is customer_id and the column 17 to 23 were calculations involving the values of other columns.

- customer_id should be the unique identifier in the three datasets and as seen transactions dataset has a minimum of 1 and a maximum of 5034 as the customer id values, customer address has 1 as the minimum and 4003 as the maximum with some missing integers while demographic dataset has 1 as the minimum and 4000 as the maximum, it therefore implies that the datasets do not have the same number of customers which can result in some values not connecting in the three datasets.

New Data

- write_csv(Transactions_3, "transactions_data.csv")
- write_csv(CustomerDemographic_4, "demographic_data.csv")

- write_csv(CustomerAddress_3, “address_data.csv”)