Assignment\_2BA

phani varshitha

2023-10-22

Online\_Retail <- read.csv("C:/Users/varshitha/Downloads/Online\_Retail (1).csv")  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

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

library(zoo)

##   
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric

library(readxl)

#Questions and Solutions

#Solution-1  
total\_countries=Online\_Retail %>% group\_by(Country) %>% count(Country)  
  
percent\_countries=Online\_Retail %>% group\_by(Country) %>% summarise(percent = 100\* n()/nrow(Online\_Retail))  
  
countries\_filtered\_percent=filter(percent\_countries, percent>1)  
  
##Total Countries Count.  
total\_countries

## # A tibble: 38 × 2  
## # Groups: Country [38]  
## Country n  
## <chr> <int>  
## 1 Australia 1259  
## 2 Austria 401  
## 3 Bahrain 19  
## 4 Belgium 2069  
## 5 Brazil 32  
## 6 Canada 151  
## 7 Channel Islands 758  
## 8 Cyprus 622  
## 9 Czech Republic 30  
## 10 Denmark 389  
## # ℹ 28 more rows

##Percentage of Transactions which are greater than > 1.  
countries\_filtered\_percent

## # A tibble: 4 × 2  
## Country percent  
## <chr> <dbl>  
## 1 EIRE 1.51  
## 2 France 1.58  
## 3 Germany 1.75  
## 4 United Kingdom 91.4

#Solution-2  
TransactionValue=(Online\_Retail$Quantity\*Online\_Retail$UnitPrice)  
  
#Addition of the Transaction Value column to the Online Retail table(import data).  
  
updated\_data=cbind(Online\_Retail,TransactionValue)  
  
##Note:The data which has transaction value column added named as updated\_data.

#Solution-3  
transaction\_total=updated\_data %>% group\_by(Country) %>% summarise(sum=sum(TransactionValue))  
  
transaction\_total\_filtered= filter(transaction\_total,transaction\_total$sum>130000)  
  
#Total of Transaction value for Each Countries.  
transaction\_total

## # A tibble: 38 × 2  
## Country sum  
## <chr> <dbl>  
## 1 Australia 137077.  
## 2 Austria 10154.  
## 3 Bahrain 548.  
## 4 Belgium 40911.  
## 5 Brazil 1144.  
## 6 Canada 3666.  
## 7 Channel Islands 20086.  
## 8 Cyprus 12946.  
## 9 Czech Republic 708.  
## 10 Denmark 18768.  
## # ℹ 28 more rows

#Filtering the transactions which are greater than 130000.  
transaction\_total\_filtered

## # A tibble: 6 × 2  
## Country sum  
## <chr> <dbl>  
## 1 Australia 137077.  
## 2 EIRE 263277.  
## 3 France 197404.  
## 4 Germany 221698.  
## 5 Netherlands 284662.  
## 6 United Kingdom 8187806.

#Solution-4  
Temp=strptime(updated\_data$InvoiceDate,format='%m/%d/%Y %H:%M',tz='GMT')  
head(Temp)

## [1] "2010-12-01 08:26:00 GMT" "2010-12-01 08:26:00 GMT"  
## [3] "2010-12-01 08:26:00 GMT" "2010-12-01 08:26:00 GMT"  
## [5] "2010-12-01 08:26:00 GMT" "2010-12-01 08:26:00 GMT"

updated\_data$New\_Invoice\_Date <- as.Date(Temp)  
  
updated\_data$Invoice\_Day\_Week= weekdays(updated\_data$New\_Invoice\_Date)  
  
updated\_data$New\_Invoice\_Hour = as.numeric(format(Temp, "%H"))  
  
updated\_data$New\_Invoice\_Month = as.numeric(format(Temp, "%m"))  
  
updated\_data$New\_Invoice\_Date[20000]-updated\_data$New\_Invoice\_Date[10]

## Time difference of 8 days

#Solution-4A  
#Percentage of number of transactions based on week days.  
days\_week\_count=updated\_data %>% group\_by(Invoice\_Day\_Week) %>% summarise(percent = 100\* n()/nrow(Online\_Retail))  
days\_week\_count

## # A tibble: 6 × 2  
## Invoice\_Day\_Week percent  
## <chr> <dbl>  
## 1 Friday 15.2  
## 2 Monday 17.6  
## 3 Sunday 11.9  
## 4 Thursday 19.2  
## 5 Tuesday 18.8  
## 6 Wednesday 17.5

#Solution-4B  
#Percentage of Transactions Value.  
days\_week\_sum = updated\_data %>% group\_by(Invoice\_Day\_Week) %>% summarise(sum=sum(TransactionValue))  
  
#Calculating the percentage for Transaction Value by week days.  
Week\_quan\_percent = 100\*(days\_week\_sum$sum)/sum(days\_week\_sum$sum)  
  
#Replacing the sum with the percentage value.  
days\_week\_sum$sum = Week\_quan\_percent  
days\_week\_sum

## # A tibble: 6 × 2  
## Invoice\_Day\_Week sum  
## <chr> <dbl>  
## 1 Friday 15.8   
## 2 Monday 16.3   
## 3 Sunday 8.27  
## 4 Thursday 21.7   
## 5 Tuesday 20.2   
## 6 Wednesday 17.8

#Solution-4C  
#Percentage of Transactions Value by month of the year.  
Invoice\_month\_sum = updated\_data %>% group\_by(New\_Invoice\_Month) %>% summarise(sum=sum(TransactionValue))  
Month\_quan\_percent = 100\*(Invoice\_month\_sum$sum)/sum(Invoice\_month\_sum$sum)  
Invoice\_month\_sum$sum = Month\_quan\_percent  
Invoice\_month\_sum

## # A tibble: 12 × 2  
## New\_Invoice\_Month sum  
## <dbl> <dbl>  
## 1 1 5.74  
## 2 2 5.11  
## 3 3 7.01  
## 4 4 5.06  
## 5 5 7.42  
## 6 6 7.09  
## 7 7 6.99  
## 8 8 7.00  
## 9 9 10.5   
## 10 10 11.0   
## 11 11 15.0   
## 12 12 12.1

#Solution-4d  
#Filtering the Australia's transactions based on New\_Invoice\_date.  
Australia\_transaction = updated\_data %>% filter(Country == "Australia") %>% group\_by(New\_Invoice\_Date) %>% summarise(total=n())  
  
#Finding the date which has maximum number of transactions.  
Max\_transaction\_date = Australia\_transaction[which.max(Australia\_transaction$total),]  
Max\_transaction\_date

## # A tibble: 1 × 2  
## New\_Invoice\_Date total  
## <date> <int>  
## 1 2011-06-15 139

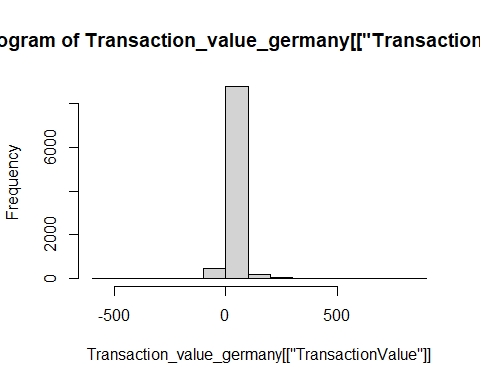
#This Package is used for handling time-indexed data and is particularly useful when dealing with irregularly spaced time series.  
library(zoo)

#Solution-4e  
#Filtering the transactions for the hours between 7:00 to 20:00.  
Sum\_quan = updated\_data %>% filter( New\_Invoice\_Hour >=7) %>% group\_by(New\_Invoice\_Hour) %>% summarise(sum\_val= sum(Quantity))  
  
#Adding the two consecutive rows.  
Con\_sum=rollapply(Sum\_quan$sum\_val,2,sum)  
  
#Creating the maintenance column.  
maintainance=c(7:19)  
  
#creating the data frame for the maintenance and Con\_sum.  
Main\_table=data.frame(maintainance,Con\_sum)  
  
#checking the minimum value of Con\_sum and the hour where they can start maintenance.  
maintainance\_hour=Main\_table[which.min(Main\_table$Con\_sum),]  
maintainance\_hour

## maintainance Con\_sum  
## 13 19 40298

##From the above result we can say that At 19th Hour can start the maintenance.

#Solution-5  
#Plotting graph between transaction value with the frequency for Germany country.  
Transaction\_value\_germany = filter(updated\_data, updated\_data$Country == "Germany")  
hist(Transaction\_value\_germany[["TransactionValue"]])



#Solution-6  
#Removing the NA values in Customer ID Column.  
NA\_custid=updated\_data[!is.na(updated\_data$CustomerID),]  
  
#Number of transactions with respect to Customer ID.  
Count\_transactions\_custid = NA\_custid %>% group\_by(CustomerID) %>% summarise(count=n())  
  
#printing the row which has max count of transactions.  
Max\_Count\_transactions= Count\_transactions\_custid[which.max(Count\_transactions\_custid$count),]  
  
# Adding the transaction value with respect to Customer ID.  
Sum\_transactions\_custid = NA\_custid %>% group\_by(CustomerID) %>% summarise(Numoftransactions=(sum(TransactionValue,na.rm = T)))  
  
#printing the row which has max sum of transaction value.  
Max\_Sum\_transactions= Sum\_transactions\_custid[which.max(Sum\_transactions\_custid$Numoftransactions),]  
Max\_Count\_transactions

## # A tibble: 1 × 2  
## CustomerID count  
## <int> <int>  
## 1 17841 7983

Max\_Sum\_transactions

## # A tibble: 1 × 2  
## CustomerID Numoftransactions  
## <int> <dbl>  
## 1 14646 279489.

#Solution-7  
#Percentage of NA's for each column.  
NA\_per\_each = colMeans(is.na(updated\_data))\*100  
NA\_per\_each

## InvoiceNo StockCode Description Quantity   
## 0.00000 0.00000 0.00000 0.00000   
## InvoiceDate UnitPrice CustomerID Country   
## 0.00000 0.00000 24.92669 0.00000   
## TransactionValue New\_Invoice\_Date Invoice\_Day\_Week New\_Invoice\_Hour   
## 0.00000 0.00000 0.00000 0.00000   
## New\_Invoice\_Month   
## 0.00000

#Solution-8  
#Number of Transactions with customer ID missing.  
null\_Custid = updated\_data[is.na(updated\_data$CustomerID),]  
  
# Segregating the missing CustomerID based on countries.  
table(null\_Custid$Country)

##   
## Bahrain EIRE France Hong Kong Israel   
## 2 711 66 288 47   
## Portugal Switzerland United Kingdom Unspecified   
## 39 125 133600 202

#Solution-9  
# Assuming the column name for customer ID is "CustomerID" and visit timestamp is "New\_Invoice\_Date"  
customer\_data= updated\_data %>%  
 arrange(CustomerID, New\_Invoice\_Date) %>% # Sort by customer and timestamp  
 group\_by(CustomerID) %>%  
 mutate(days\_between\_visits = c(NA, diff(New\_Invoice\_Date))) %>%  
 ungroup()  
head(customer\_data)

## # A tibble: 6 × 14  
## InvoiceNo StockCode Description Quantity InvoiceDate UnitPrice CustomerID  
## <chr> <chr> <chr> <int> <chr> <dbl> <int>  
## 1 541431 23166 "MEDIUM CERAMIC… 74215 1/18/2011 … 1.04 12346  
## 2 C541433 23166 "MEDIUM CERAMIC… -74215 1/18/2011 … 1.04 12346  
## 3 537626 85116 "BLACK CANDELAB… 12 12/7/2010 … 2.1 12347  
## 4 537626 22375 "AIRLINE BAG VI… 4 12/7/2010 … 4.25 12347  
## 5 537626 71477 "COLOUR GLASS. … 12 12/7/2010 … 3.25 12347  
## 6 537626 22492 "MINI PAINT SET… 36 12/7/2010 … 0.65 12347  
## # ℹ 7 more variables: Country <chr>, TransactionValue <dbl>,  
## # New\_Invoice\_Date <date>, Invoice\_Day\_Week <chr>, New\_Invoice\_Hour <dbl>,  
## # New\_Invoice\_Month <dbl>, days\_between\_visits <dbl>

# Calculate the average time between consecutive visits for all customers  
average\_days\_between\_visits <- mean(customer\_data$days\_between\_visits, na.rm = TRUE)  
  
# Print the result  
cat("Average number of days between consecutive visits:", average\_days\_between\_visits, "days\n")

## Average number of days between consecutive visits: 1.088318 days

#Solution-10  
# Filtering the data set for french customers.  
French\_customers = filter(updated\_data,Country=="France" )  
  
#Return rate for the french customers.  
Return\_rate = nrow(filter(French\_customers,Quantity<1))/nrow(French\_customers)  
Return\_rate

## [1] 0.01741264

#Solution-11  
#revenue of each product.  
Product\_revenue= updated\_data %>% group\_by(StockCode) %>% summarise(Sum\_trnsvalue = sum(TransactionValue))  
  
#Selecting the product with highest revenue.  
Product\_revenue[which.max(Product\_revenue$Sum\_trnsvalue),]

## # A tibble: 1 × 2  
## StockCode Sum\_trnsvalue  
## <chr> <dbl>  
## 1 DOT 206245.

#Solution-12  
#Number of unique customers.  
length(unique(updated\_data$CustomerID))

## [1] 4373