# Final Project

Ednilson Chiambo , Kylie Cancilla , Kaung Thiha and Dante Thomas

2023-12-10

## Ednilson Chiambo

I worked on the introduction part which was to introduce the dataset used for the project, names of the people involved in the project, challenges and issues encountered when working with this dataset and what we would be covering during the presentation. This involved the name of the data, what was in the data, issues such as missing values, wrong values and many more, and where did the data come from and how we would use the data and what analysis we are trying to convey from the data. For instance, the data's name is Retail and it's from a company from UK that sells gifts. After looking at the data we came up with some hypothesis that we would later confirm if that's true or not based on our analysis. Hypothesis and questions such as will be an increase in sales during the holiday season or does the region affect quantity or type of products purchased? or even What is the effect of time on purchasing trends: Do certain times of day see more purchasing, does time affect price?

Code used for cleaning the data :

```
library(readxl)
library(magrittr)
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(ggplot2)
library(stringr)


data = read_excel("~/Documents/Introduction to R/Online Retail.xlsx")


#This line of code creates a new dataframe of the data given
df = data.frame(Stockcode = data$StockCode, Quantity= data$Quantity, Description = data$Description , U



#This line of code filters out rows with negative values for the quantity column
df <- df %>%
 filter(Quantity> 0)
```

```r
# D1 is the original data after cleaning and dealing with all the errors encountered in the data
class(df$Stockcode)
```

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

```r
pattern <- "^[A-Za-z]"
d1 <- df[!grepl(pattern,df$Stockcode),]
d1 <- d1[d1$UnitPrice != 0, ]
```

Furthermore, I also worked on finding the relationship between the Quantity vs. Unit price to check if the price of unit affected the quantities of each product bought.

Code used for this analysis :

```r
#This lone of code finds the linear regression between the quantity and price
linear_model1 = lm(Quantity ~ UnitPrice , d1)


summary(linear_model1)
```

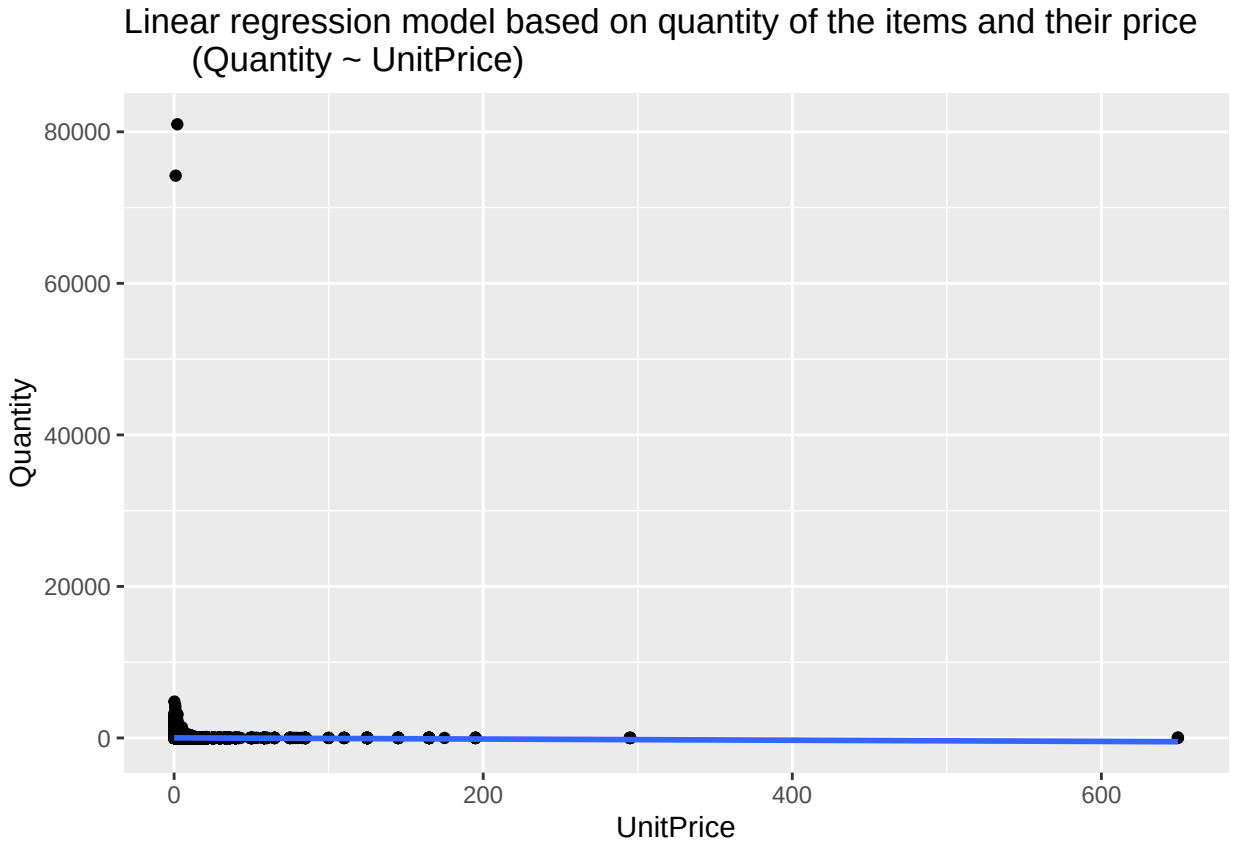
```
##
## Call:
## lm(formula = Quantity ~ UnitPrice, data = d1)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##    -12     -9     -6      0  80984
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13.13121    0.26756   49.08   <2e-16 ***
## UnitPrice   -0.78498    0.04899  -16.02   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 155.8 on 527723 degrees of freedom
## Multiple R-squared:  0.0004863,  Adjusted R-squared:  0.0004844
## F-statistic: 256.8 on 1 and 527723 DF,  p-value: < 2.2e-16
```

```r
#This line of code finds the correlation between the quantity and unitprice
correlation <- cor(d1$Quantity, d1$UnitPrice)
correlation
```

```
## [1] -0.02205208
```

```r
ggplot(d1, aes(x = UnitPrice, y = Quantity))+
  geom_point()+
  geom_smooth(method = "lm", se = FALSE)+
  labs(title = "Linear regression model based on quantity of the items and their price
       (Quantity ~ UnitPrice)")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

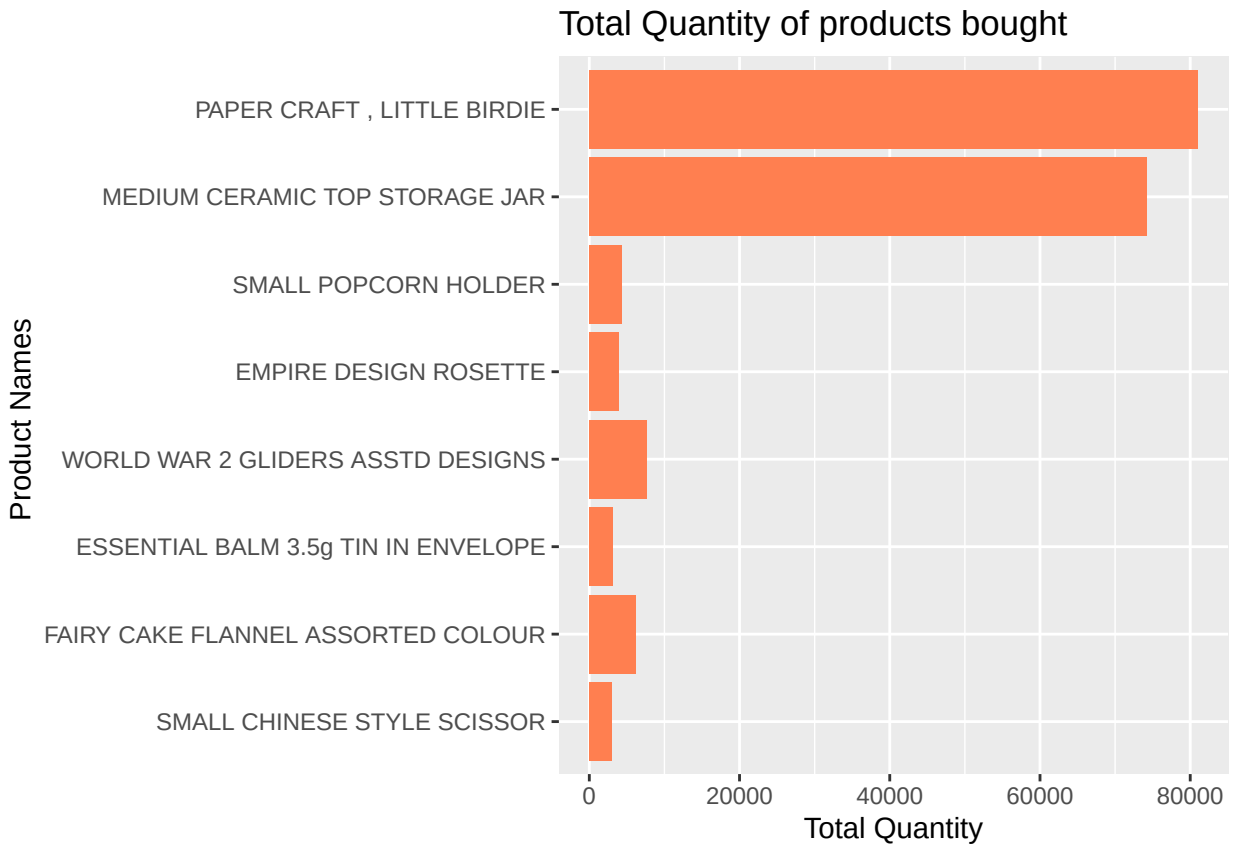



After doing this analysis, we could notice that there was not a correlation or linear relationship between the variable's quantity and unit price. However, this lack of correlation or relationship between them could be due to so many reasons. For instance, maybe this was since the dataset included a wide range of gift products with varying prices and quantities or Purchasing behavior might be influenced by seasonal trends or specific occasions or discounts, promotions, or sales might impact the buying behavior, causing customers to purchase more of a lower-priced item during a sale period, which doesn't necessarily correlate with higher-priced items or even maybe because customer preferences might vary widely. In conclusion, more analysis would be needed to find the reason why there was no correlation at all.

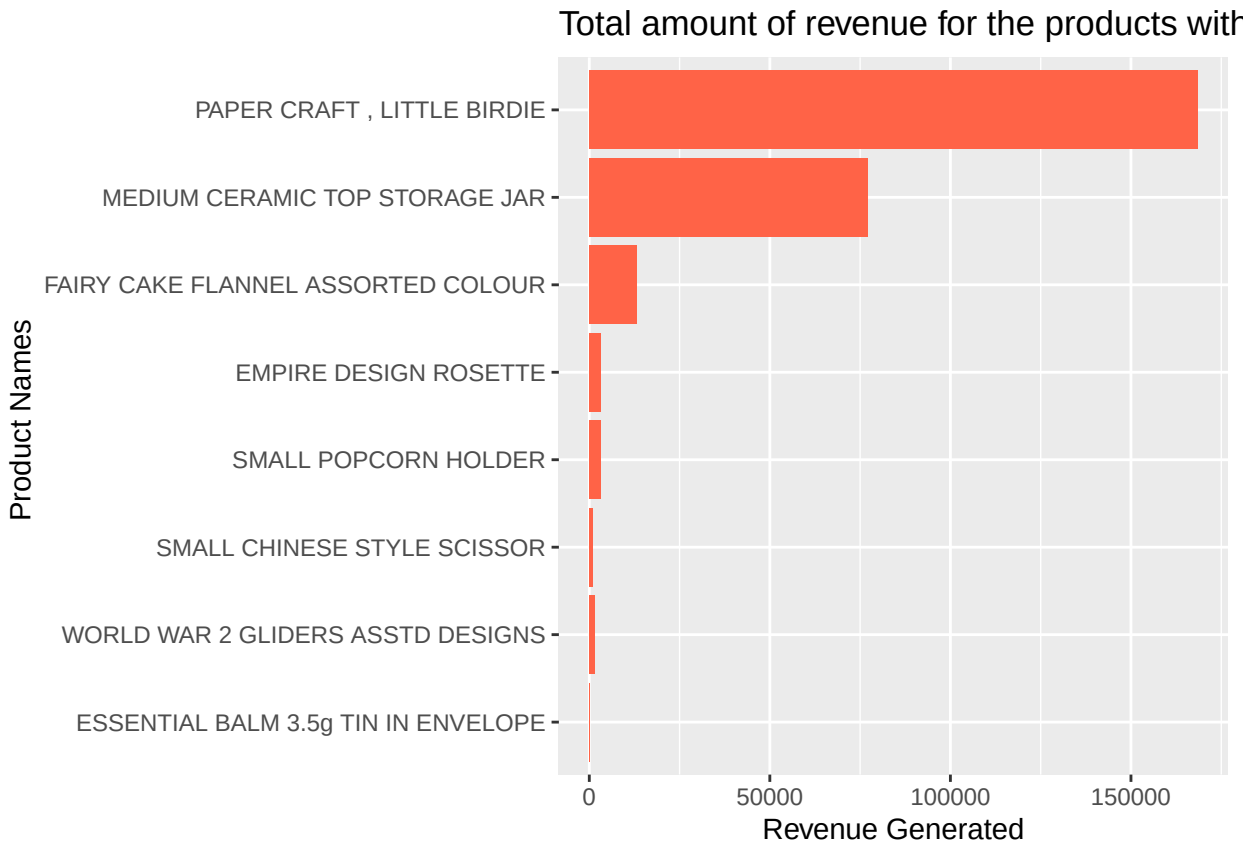
I also worked on finding the top 10 products with most amount bought and how much they revenue they generated. Why would be important? Well, as one of my colleges in the project worked on finding the most popular products frequency according to their regions, I thought it would be a great idea to find if my top 10 products with most amount bought would be part of his list which later, we proved to be true.

```
#This line of code gives me the 10 largest numbers in the quantity column
top_10_largest2 <- d1 %>%
  arrange(desc(d1$Quantity)) %>%
  slice_head(n = 10)


#This code creates a new dataframe with the 10 most amount of item bought,
#their price and the revenue generated.
df_quantity_unitprice = data.frame(Quantities = top_10_largest2$Quantity ,
                                   UnitPrice = top_10_largest2$UnitPrice,
                                   Product_name =  top_10_largest2$Description,
                                   Revenue = (top_10_largest2$Quantity)* top_10_largest2$UnitPrice,
                                   Stockcode = top_10_largest2$Stockcode)
```

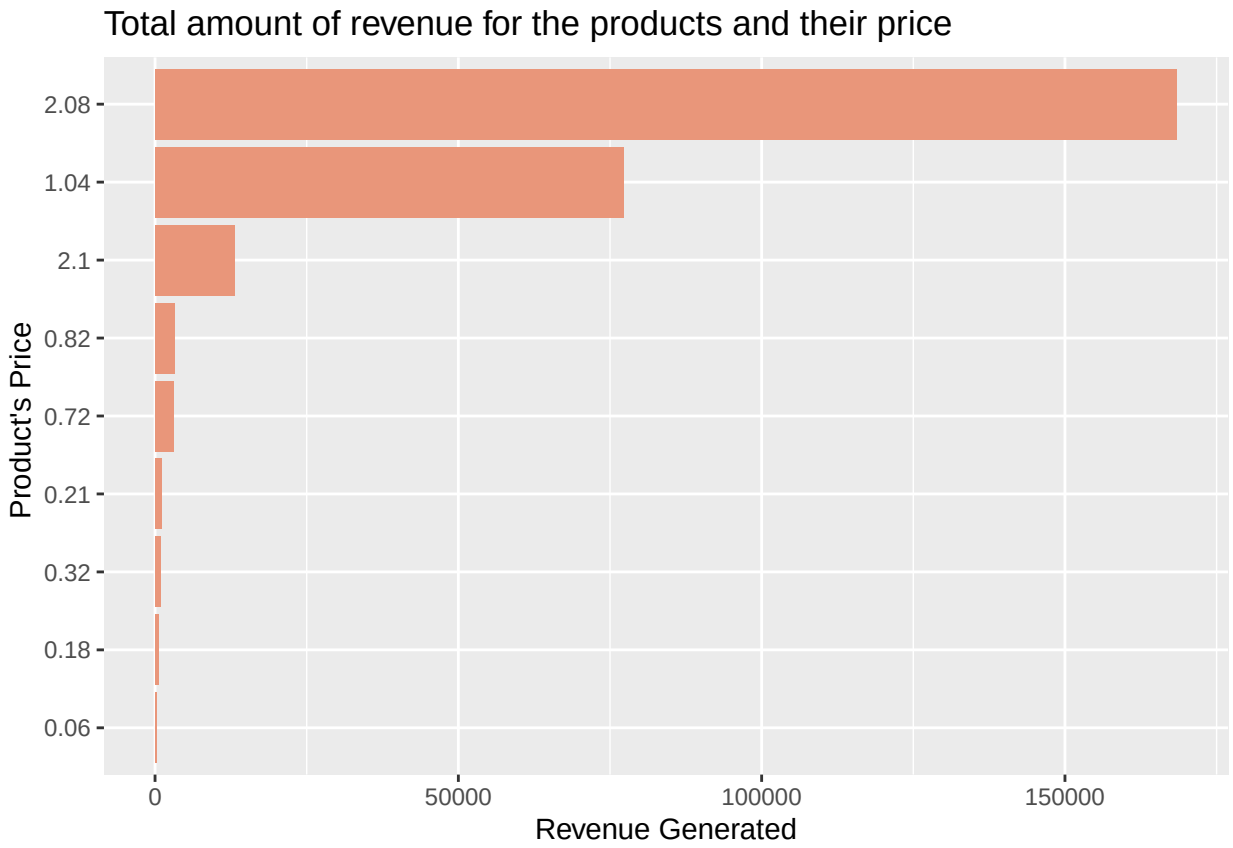
```r
#This code graphs the total amount of products and their names for the top 10 of the items with most am
ggplot(df_quantity_unitprice, aes(x = reorder(Product_name,Quantities ), y =Quantities )) +
  geom_bar(stat = "identity", fill = "coral") +
  coord_flip() +
  labs(title = "Total Quantity of products bought", x = "Product Names ", y = "Total Quantity")
```




```r
#Total amount of revenue for the products with most purchases and the product's names
ggplot(df_quantity_unitprice, aes(x = reorder(Product_name,Revenue ), y =Revenue )) +
  geom_bar(stat = "identity", fill = "tomato") +
  coord_flip() +
  labs(title = "Total amount of revenue for the products with most purchases", x = "Product Names ", y =
```




```
#Total amount of revenue for the products with most purchases and their respective price
ggplot(df_quantity_unitprice, aes(x = reorder(UnitPrice,Revenue ), y =Revenue )) +
  geom_bar(stat = "identity", fill = "darksalmon") +
  coord_flip() +
  labs(title = "Total amount of revenue for the products and their price", x = "Product's Price ", y =
```




## Kylie Cancilla

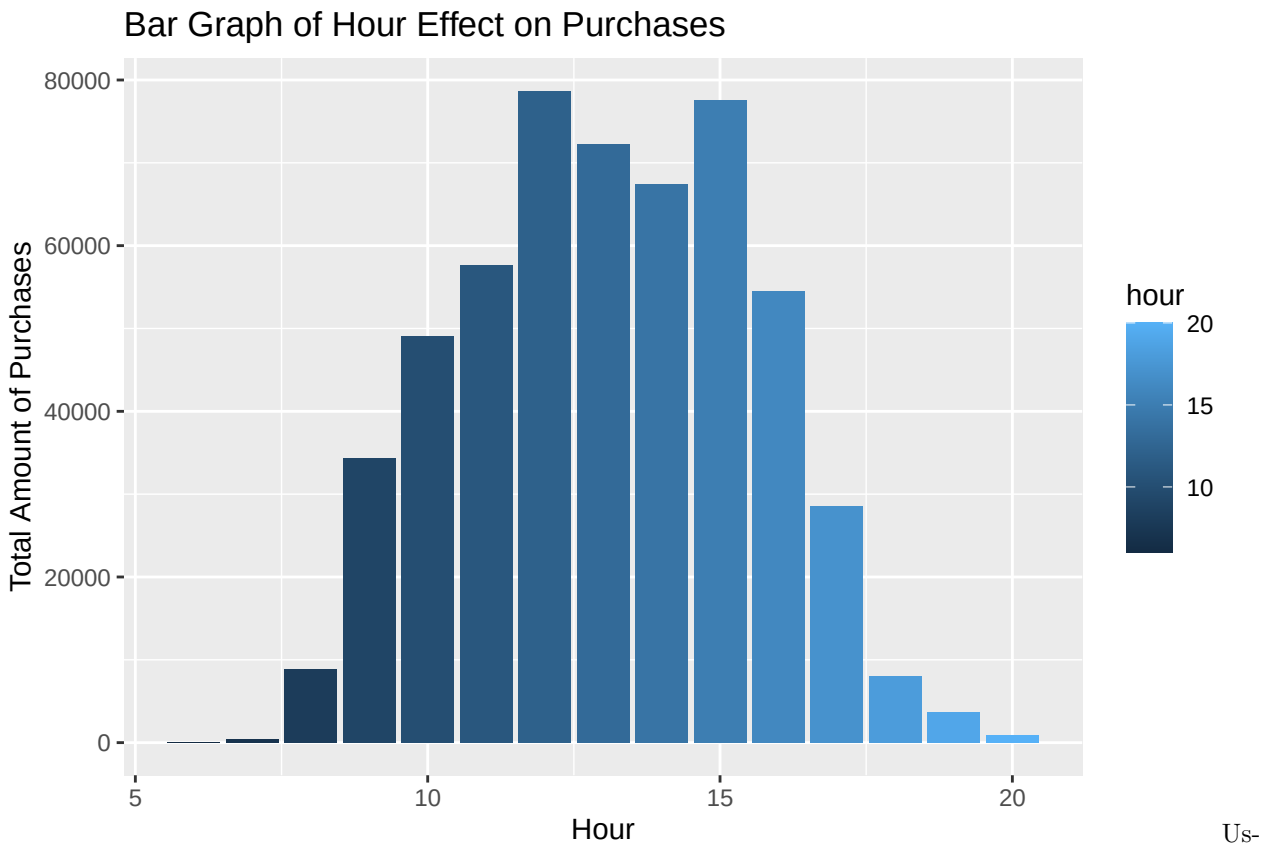
```
data
```

```
## # A tibble: 541,909 x 8
##    InvoiceNo StockCode Description        Quantity InvoiceDate         UnitPrice
##    <chr>     <chr>     <chr>                 <dbl> <dttm>                  <dbl>
##  1 536365    85123A    WHITE HANGING HEA~        6 2010-12-01 08:26:00      2.55
##  2 536365    71053     WHITE METAL LANTE~        6 2010-12-01 08:26:00      3.39
##  3 536365    84406B    CREAM CUPID HEART~        8 2010-12-01 08:26:00      2.75
##  4 536365    84029G    KNITTED UNION FLA~        6 2010-12-01 08:26:00      3.39
##  5 536365    84029E    RED WOOLLY HOTTIE~        6 2010-12-01 08:26:00      3.39
##  6 536365    22752     SET 7 BABUSHKA NE~        2 2010-12-01 08:26:00      7.65
##  7 536365    21730     GLASS STAR FROSTE~        6 2010-12-01 08:26:00      4.25
##  8 536366    22633     HAND WARMER UNION~        6 2010-12-01 08:28:00      1.85
##  9 536366    22632     HAND WARMER RED P~        6 2010-12-01 08:28:00      1.85
## 10 536367    84879     ASSORTED COLOUR B~       32 2010-12-01 08:34:00      1.69
## # i 541,899 more rows
## # i 2 more variables: CustomerID <dbl>, Country <chr>
```

```
day_time_df = str_split_fixed(data$InvoiceDate, ' ', 2)

time_hour_df = str_split_fixed(day_time_df[,2], ':', 3)

data$day = c(as.Date(day_time_df[,1]))
data$hour = c(as.numeric(time_hour_df[,1]))
```

There was only one column with in the data set that gave information about time of day and day of the week. Therefore, I used string analysis to split up the column and create new columns in the data set that were easier to use.

```
ggplot(data, aes(x = hour, group = hour, fill = hour)) +
  geom_bar() +
  labs(title = "Bar Graph of Hour Effect on Purchases",
       x = "Hour",
       y = "Total Amount of Purchases")
```

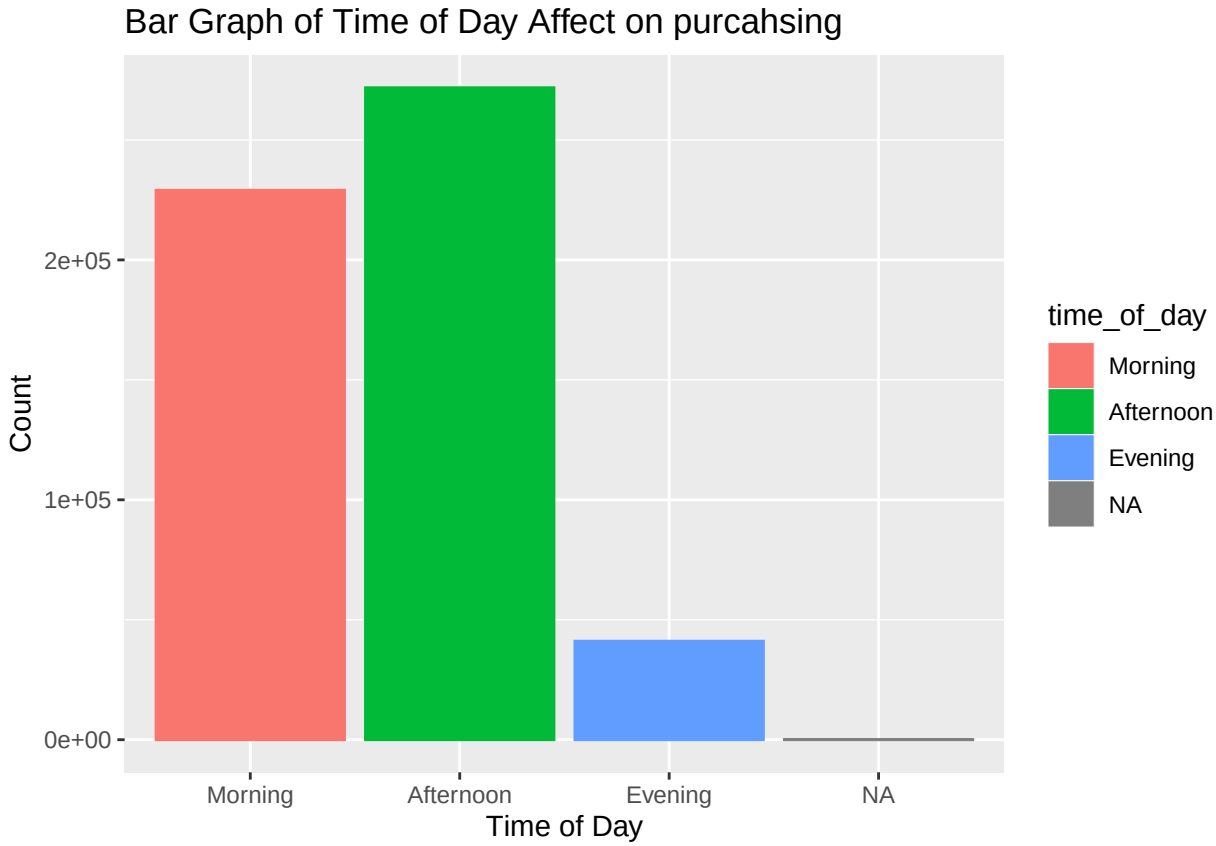



Using the new column, hour, I visualized the effect the hour of the day had on purchases, using a bar chart. it clearly shows a time frame of the day that purchasing is most popular, 9am - 5pm. An online retailer can use this information to decide the hours of a potential store. The peak purchasing hour was 12pm, so an online retailer can take this into consideration when sending alerts to costumers. If they send advertisements around that time they might be more likely to get the maximum amount of customers going onto the website.

```
time_of_dayFunc <- function(hour) {
  if (hour >= 1 && hour <= 6) {
    return("Night")
  } else if (hour <= 12) {
    return("Morning")
  } else if (hour <= 16) {

    return("Afternoon")
  } else {
    return("Evening")
  }
}
```

```r
data$time_of_day <- lapply(data$hour, time_of_dayFunc) %>% as.character()
```

```r
time_of_day_order = c("Morning", "Afternoon", "Evening")

data$time_of_day = factor(data$time_of_day, levels = time_of_day_order)

ggplot(data, aes(x = time_of_day,
                 fill = time_of_day,
                 color = time_of_day)) +
  geom_bar() +
  labs(title = "Bar Graph of Time of Day Affect on purcahsing",
       x = "Time of Day",
       y = "Count"
       )
```




I decided that I wanted to create a column about the time of day (morning, afternoon, evening, night) and see how that effected purchasing. The hour vs. purchasing graph ended up being more informative and useful than this graph. In agreement with the hour graph, the time of day graph showed afternoon (12p-4p) to be the busiest time for purchasing.

```r
weekdayFunc = function(x){
  weekdays(x)
}

data$Weekday <- lapply(data$day, weekdayFunc) %>% as.character()

sum(data$Weekday == "Saturday")
```

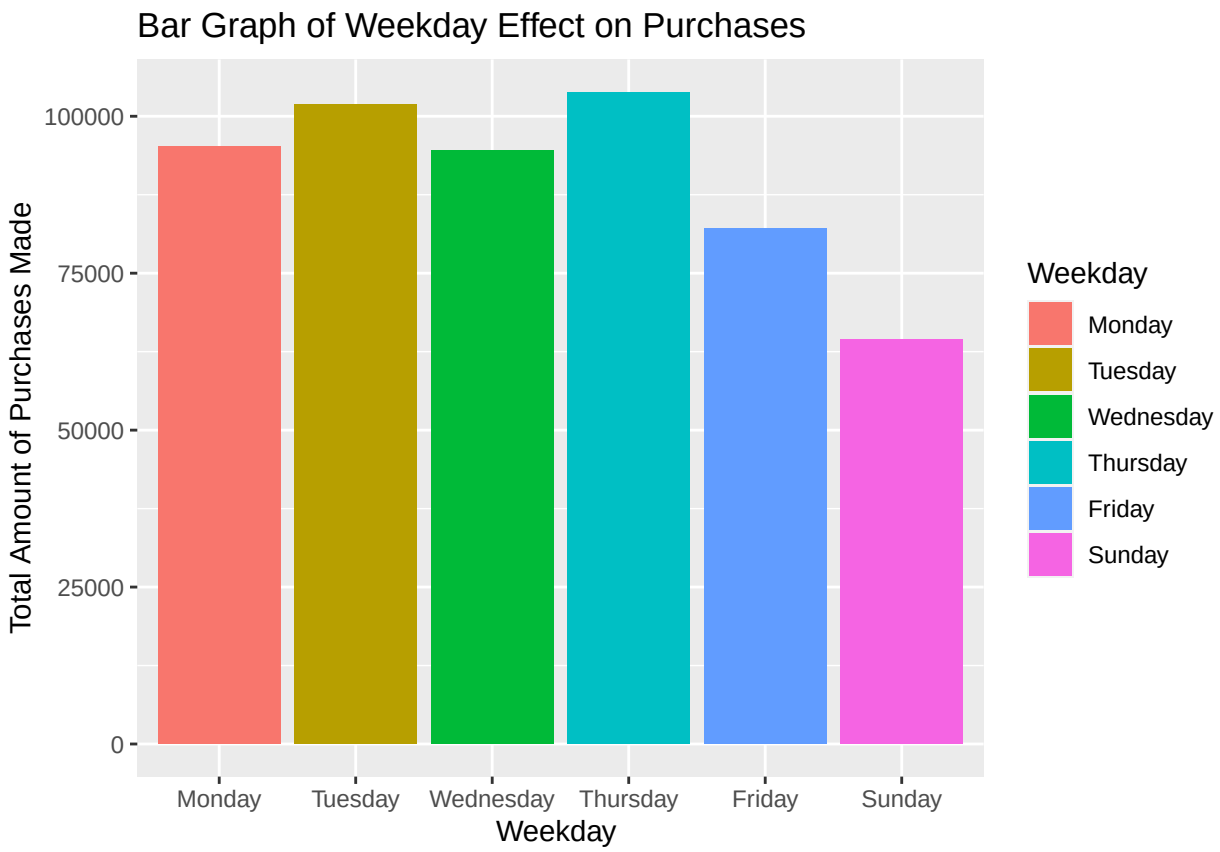
```
## [1] 0
```

```
unique(data$Weekday)
```

```
## [1] "Wednesday" "Thursday"  "Friday"    "Sunday"    "Monday"    "Tuesday"
```

```
#View(data)
```

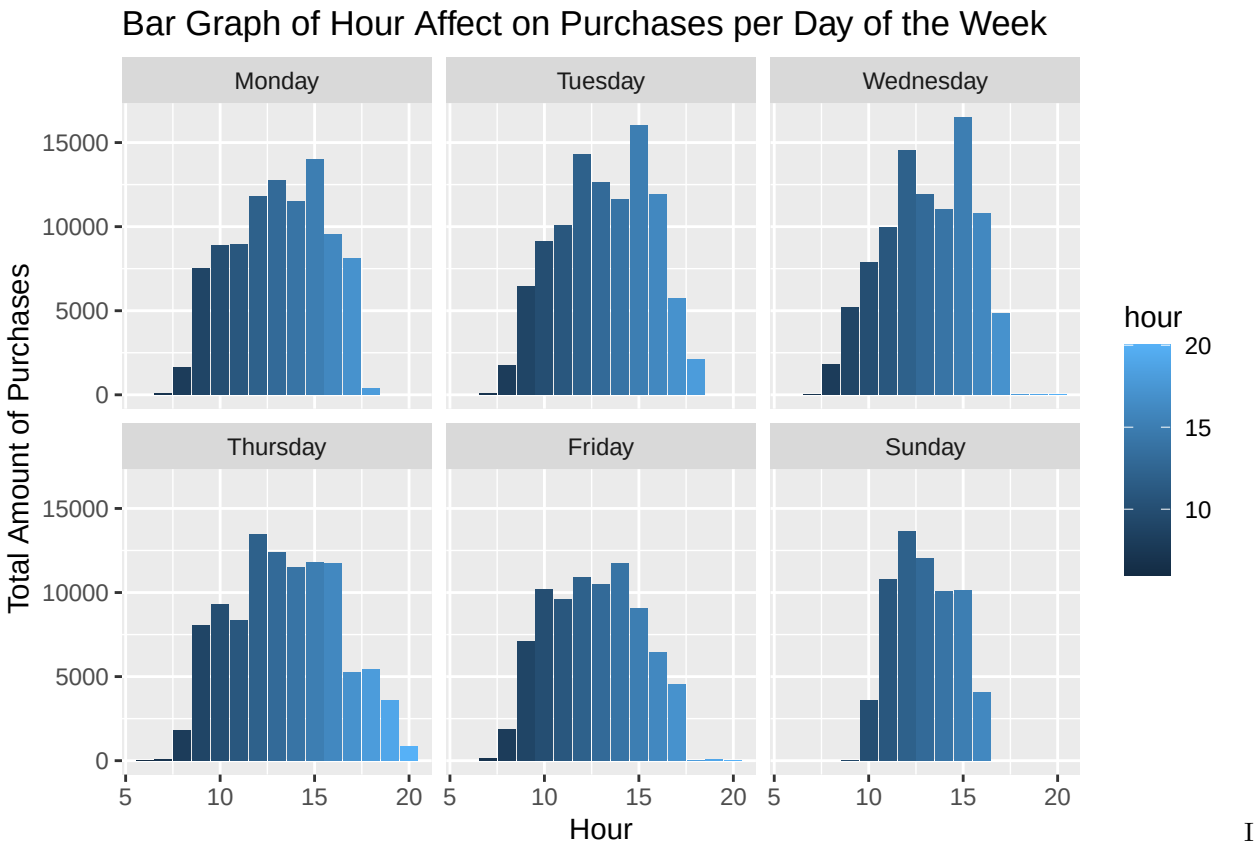
```
weekday_order = c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday")

data$Weekday = factor(data$Weekday, levels = weekday_order)

ggplot(data, aes(x = Weekday, fill = Weekday)) +
  geom_bar() +
  labs(title = "Bar Graph of Weekday Effect on Purchases",
       x = "Weekday ",
       y = "Total Amount of Purchases Made")
```




```
sum(data$Weekday == "Saturday")
```

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

```
ggplot(data, aes(x = hour, group = hour, fill = hour)) +
  geom_bar() +
  labs(title = "Bar Graph of Hour Affect on Purchases per Day of the Week",
       x = "Hour",
       y = "Total Amount of Purchases") +
  facet_wrap(~Weekday)
```




I wanted to know the effect the day of the week had on purchasing so I used two built in R functions (as.Date() & weekdays()) to take the date given by the data set and return a column of the day of the week in which each purchase happened. Using that column I then graphed the day of the week vs. the amount of purchases in the data set that were made on that day. Showing that the most popular day of the week was Thursday and the least popular was Saturday. Surprisingly, there was only 402 total purchases on Saturdays, while the rest of the days had over 70,000. There could be multiple explanations for this. The demographic for these items seems to be adults possibly with children, maybe parents are not shopping a lot online on Saturdays because they busy. It could also be a mistake that we can not see with just the data set such as the website often isn't working on Saturdays, or the data wasn't collected an even amount Saturdays.

## Dante Thomas

```
str(data)
```

```
## tibble [541,909 x 12] (S3: tbl_df/tbl/data.frame)
##  $ InvoiceNo  : chr [1:541909] "536365" "536365" "536365" "536365" ...
##  $ StockCode  : chr [1:541909] "85123A" "71053" "84406B" "84029G" ...
##  $ Description: chr [1:541909] "WHITE HANGING HEART T-LIGHT HOLDER" "WHITE METAL LANTERN" "CREAM CUPI
##  $ Quantity   : num [1:541909] 6 6 8 6 6 2 6 6 6 32 ...
##  $ InvoiceDate: POSIXct[1:541909], format: "2010-12-01 08:26:00" "2010-12-01 08:26:00" ...
##  $ UnitPrice  : num [1:541909] 2.55 3.39 2.75 3.39 3.39 7.65 4.25 1.85 1.85 1.69 ...
##  $ CustomerID : num [1:541909] 17850 17850 17850 17850 17850 ...
##  $ Country    : chr [1:541909] "United Kingdom" "United Kingdom" "United Kingdom" "United Kingdom" .
##  $ day        : Date[1:541909], format: "2010-12-01" "2010-12-01" ...
##  $ hour       : num [1:541909] 8 8 8 8 8 8 8 8 8 8 ...
##  $ time_of_day: Factor w/ 3 levels "Morning","Afternoon",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Weekday    : Factor w/ 7 levels "Monday","Tuesday",..: 3 3 3 3 3 3 3 3 3 3 ...
```

```
index = str_detect(data$StockCode, "[ABCDEFGHIJKLMNOPQRSTUVWXYZ]", )
data = data[!index,]
countries = split(data, data$Country)
```

This code recieves the data from the csv file, and splits the data into multiple data frames, sorted by the country of origin. The data frames are stored in a list called countires. Each entry of counrties is a data frame with all of the observations of data from a specific country.

I also removed any observations with stock codes that contained letters.

```
customers = list()

for(i in 1:length(countries)) {
  customers[[i]] = unique(countries[[i]]$CustomerID)
  #names(customers)[i] = countries[[i]]$Country[1]
}
```

The next block of code creates a list associated with each countries data frame. Each entry of the list contains a vector of cutomer ID's. These vectors contain the IDs of each customer that bought porducts from a certian countrty. I used a for loop and the unique function to create each entry of the list.

```
top_products = function(country) {
products_by_region <- country %>%
  group_by(Country, StockCode) %>%
  summarize(TotalQuantity = sum(Quantity), .groups= 'drop') %>%
  arrange(-TotalQuantity)
return(products_by_region[1:10,])
}
```

The function "top_products" takes a country as its input, and returns the 10 stock codes that had the greatest number of purchased made. It returns this infomation as a tibble that is 10 x 3. The three columns are the country of origin, the stock code for the product, as well as the number of purschases made for that item in the data set. (I had help from KT to create this function, as he had already made a similar function).

```
top_tens = lapply(countries, top_products)
all_top_products = bind_rows(top_tens)

most_bought_items = count(all_top_products, StockCode)
most_bought_items = arrange(most_bought_items, desc(n))
```

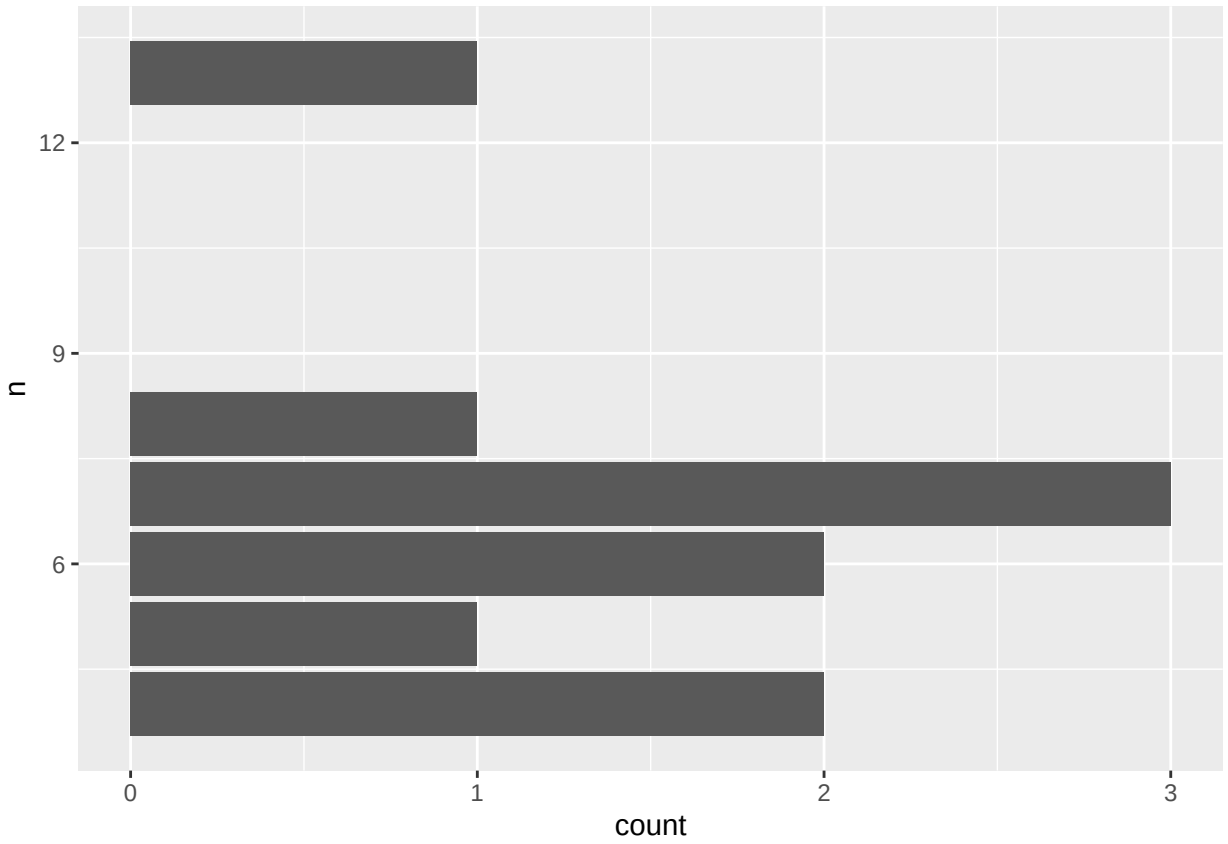
The next section of code uses the top_products function. The function is apllied with lapply to each data frame in the countries list. This returns a list of tibbles with the top ten most purchased products from each country, (in sperate tibbles). I used row bind to combine these into one tibble called "all_top_products".

Using the count and arrange functions, I counted the frequency of each stock code in the all_top_products tibble. The number of times each stock code shows up in the all_top_products is stored into a tibble called "most_bought_items". I then used the arrange function to sort the tibble from most frequent to least frequent.

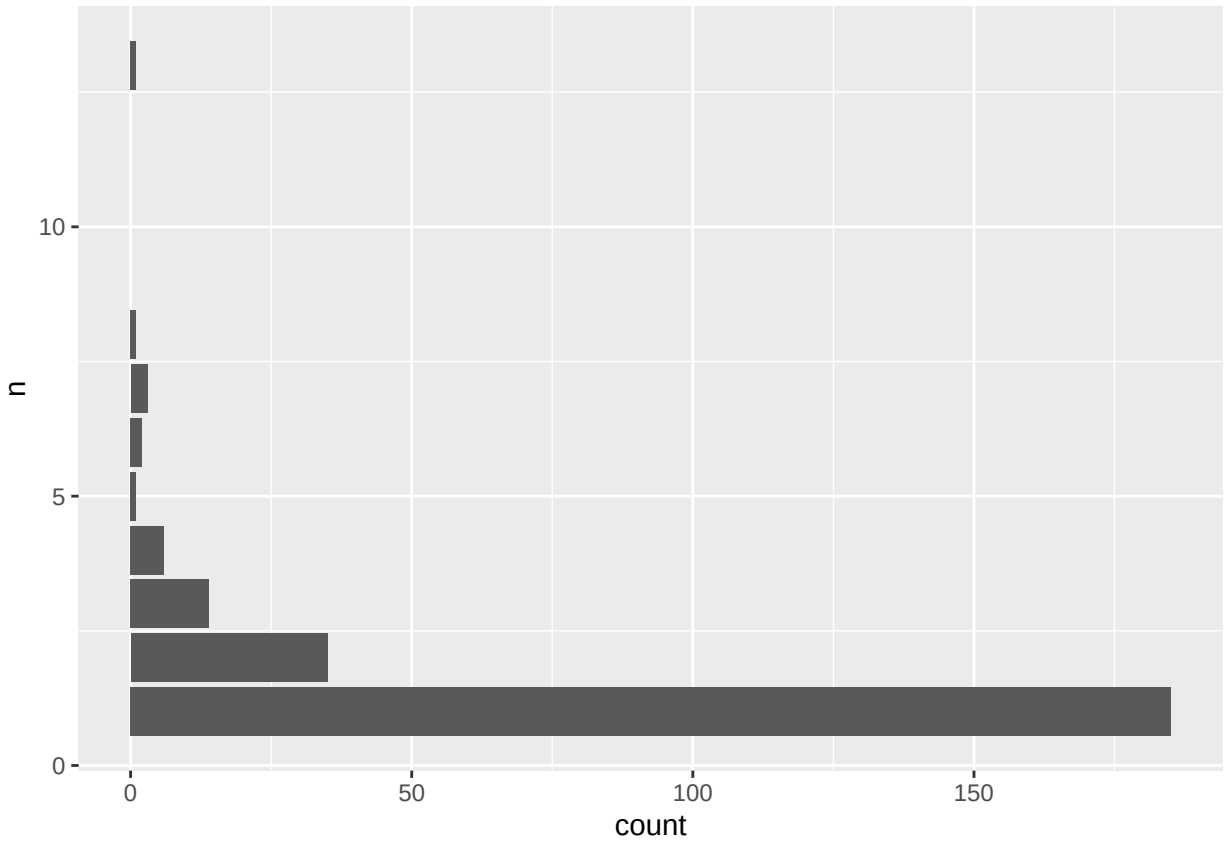
```
top_ten_stockcodes = most_bought_items[1:10,]
```

The next block of code just creates a seperate data structure for the top ten items from the most_bought_items tibble.

```
plot_1 = ggplot(top_ten_stockcodes, aes(y=n)) +
  geom_bar()
plot_1
```



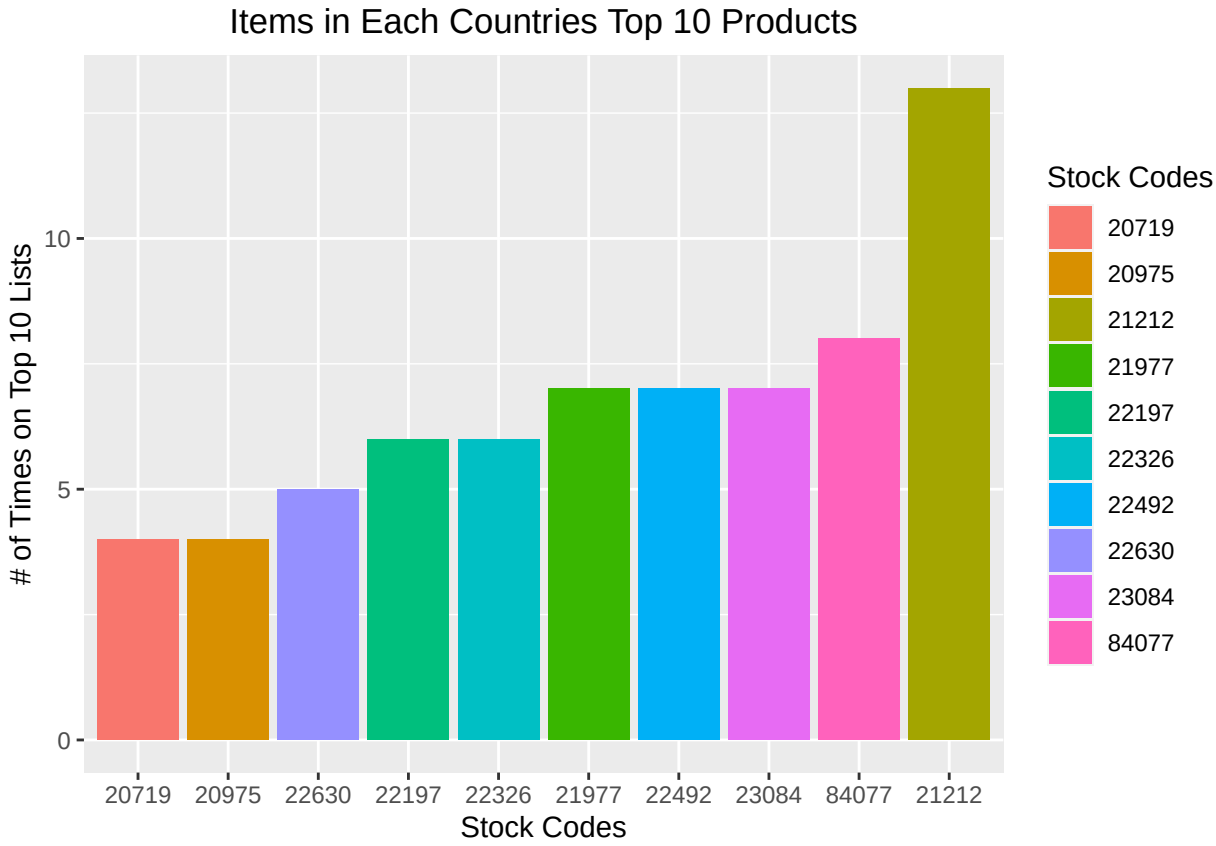

```
plot_2 = ggplot(most_bought_items, aes(y=n)) +
  geom_bar()
plot_2
```




These first two plots were the initial visual analyis for my thesis. They show the same data, the first plot just only shows the 10 highest values of "n". (which is number of times a stockcode occured in all_top_products). These plots show that a few items are polular in a large variety of countires, but a lot more products are only popular in one region. There are around 150 unique products that only show up on the most_bought_items once, but there aere only 5 items that showed up on the list more than 7 times.

This helps reinforce my thesis that the most certain products may only be polular in one region.

```
plot_3 = ggplot(top_ten_stockcodes, aes(y=n, x=reorder(StockCode, n), fill = StockCode)) +
  geom_bar(stat="identity") +
  xlab("Stock Codes") +
  ylab("# of Times on Top 10 Lists") +
  ggtitle("Items in Each Countries Top 10 Products") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_fill_discrete(name = "Stock Codes")
plot_3
```




The next plot is an improved version of the first two plots. This plot seeks to explain the same point as the other plots, but with some added information. The x-axis has the stock code for an item, and the y-axis is still the number of times the products shows up on a top ten list. I added axis labeling and a title for clarify. I also changed the color of each bar based on stock code so it could each stock code can be more easily differentiate.

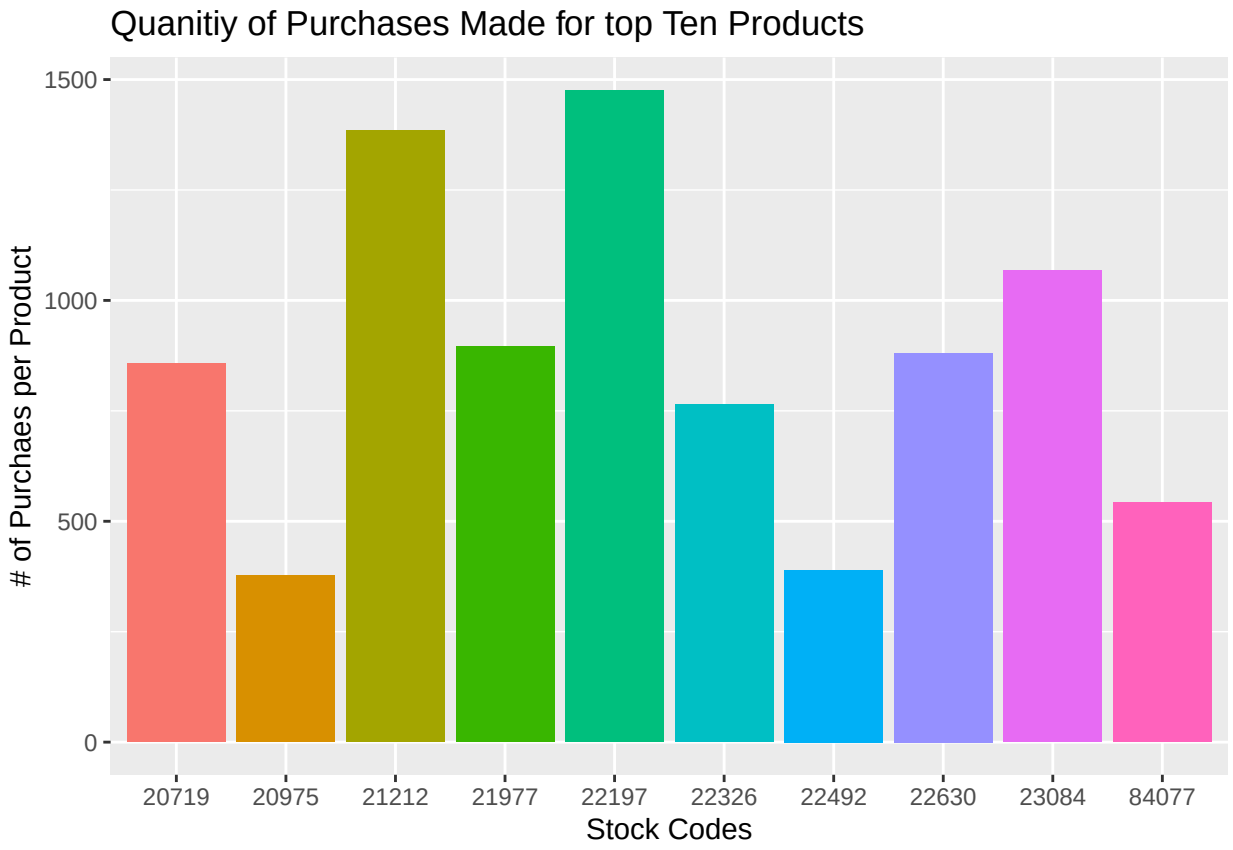
```
stock_codes = top_ten_stockcodes$StockCode

idx = data$StockCode == "0"
for(i in 1:10) {
  idx = idx | data$StockCode == stock_codes[i]
}

top_products_frame = data[idx,]
top_products_frame = count(top_products_frame, StockCode)
```

This next section of code creates a data frame which will be used to create plots. The purpose of this data frame is to obtain the rows of the data set that contain products with the ten stock codes from the top_ten_stock codes data frame. This data frame contains all the rows for products who are most popular in a lot of countries.

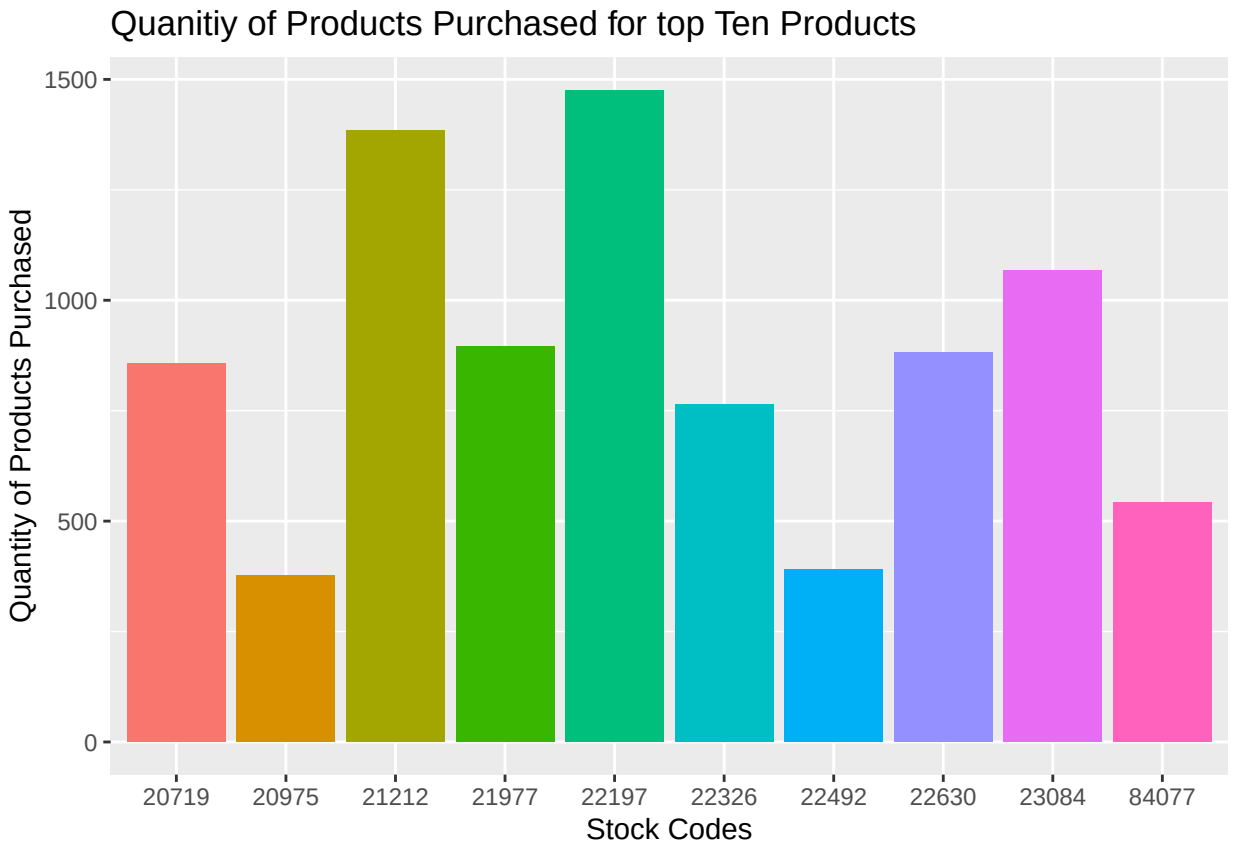
First I initialized a index vector, and created a for loop that changed the values to true if the row being observed has a stock code from the top_ten_stockcodes. I then used this index to do conditional sub-setting on the original data frame. The new data frame has the rows I need to only do visualization on specific stock codes.

```
plot_4 = ggplot(top_products_frame, aes(x=StockCode, y=n, fill=StockCode)) +
  geom_bar(stat="identity") +
  xlab("Stock Codes") +
```

```
  ylab("# of Purchaes per Product") +
  ggtitle("Quanitiy of Purchases Made for top Ten Products") +
  guides(fill="none")
plot_4
```




```
plot_5 = ggplot(top_products_frame, aes(x=StockCode, y=n, fill=StockCode)) +
  geom_bar(stat="identity") +
  xlab("Stock Codes") +
  ylab("Quantity of Products Purchased") +
  ggtitle("Quanitiy of Products Purchased for top Ten Products") +
  guides(fill="none")
plot_5
```



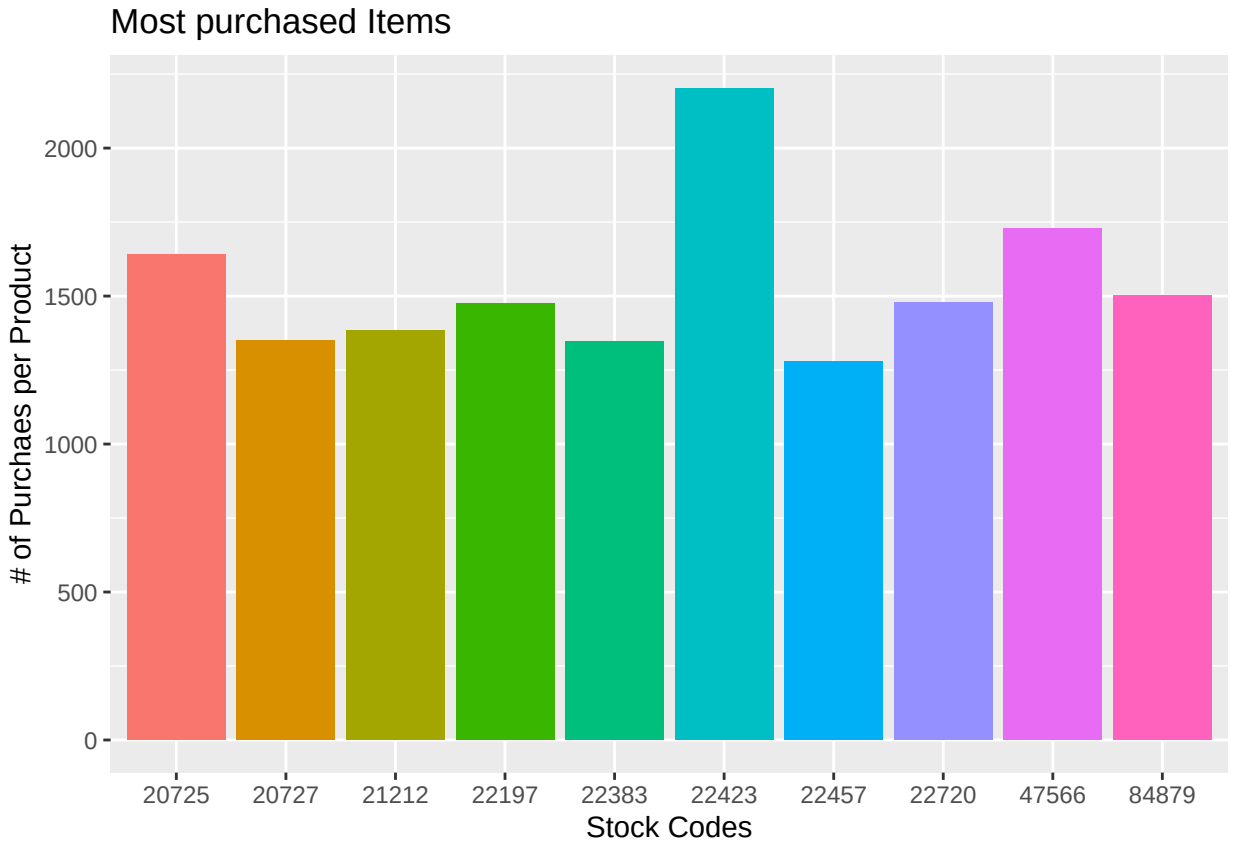

These plot begins to examine the validity of my thesis. We can observe the difference between the number of purchases made, and the quantity of products bought differ. The number of purchases made for the products I thought would be popular did not necessarily have high quantities of products purchased. The stock code 84077 had the third lowest number of purchases, but had the second highest quantity of products purchased overall.

This data does not really support my thesis, but it is not enough to say it is completely false.

```
real_top_products_frame =  count(data, StockCode)
real_top_products_frame = arrange(real_top_products_frame, desc(n))
real_top_products_frame = real_top_products_frame[1:10,]
```

This section of code also prepares a data frame which will be used to create plots. First I count the frequency of each stock code among the whole data set, then arrange the data frame from most frequent to less frequent. I only want to observe the top ten values, so I just take the ten top rows of the sorted data frame.

```
plot_6 = ggplot(real_top_products_frame, aes(x=StockCode, y=n, fill=StockCode)) +
  geom_bar(stat="identity") +
  xlab("Stock Codes") +
  ylab("# of Purchaes per Product") +
  ggtitle("Most purchased Items") +
  guides(fill="none")
plot_6
```




The plot titled, "Most purchased items" is the critical plot in further proving/disproving my thesis. The plot is structured the same as the plot titled "Quantity of Products Purchased for top Ten Products", but the ten stock codes observed are the actual ten most popular products, not the ones predicted by my thesis.

When comparing these plots, we can see my thesis is clearly in fact not correct. The variation in quantity of purchases has little variety in the most purchased item plot. However, the plot with my predicted popular products has a large variation in quantity of purchases. I believe this to mean that my hypothesis is not a good predictor of popular products in the data set. This is because the 10 most popular products all had similar number of purchases made, this makes sense because the items are all very popular. However, we don't see the same trend in my prediction plot. Because there is a large variance in the number of purchases, we can tell that the products with lower quantities of purchases are not very popular.

With this line of thinking you could keep making predictions based on various parameters, and see if the results given are similar to the actual most popular products.

The hypothesis that give the most accurate predictions could be used on other data sets to see if the hypothesis is a good predictor of online sales trends in general.

All in all, products being highly purchased in a lot of countries does not mean that that item is purchased a lot on a global scale. Furthermore, items that are popular in a lot of countries might not actually be popular for a quantity perceptive. We would have to observe more about this data set to understand how my prediction actually interacts with the data set. But this gives a pretty good grasp that the hypothesis is overall not correct.

## Kaung Thiha

Hypothesis 3 was an attempt by me to try and see if I could make business suggestions by exploring seasonality. We'll start by playing around with the data a little bit. We know that there's 500,000+ rows but not all of them are going to be useful.

```r
library(readxl)
library(dplyr)
library(ggplot2)


online_retail <- read_excel("Online Retail.xlsx")

str(online_retail)
```

```
## tibble [541,909 x 8] (S3: tbl_df/tbl/data.frame)
##  $ InvoiceNo  : chr [1:541909] "536365" "536365" "536365" "536365" ...
##  $ StockCode  : chr [1:541909] "85123A" "71053" "84406B" "84029G" ...
##  $ Description: chr [1:541909] "WHITE HANGING HEART T-LIGHT HOLDER" "WHITE METAL LANTERN" "CREAM CUPI
##  $ Quantity   : num [1:541909] 6 6 8 6 6 2 6 6 6 32 ...
##  $ InvoiceDate: POSIXct[1:541909], format: "2010-12-01 08:26:00" "2010-12-01 08:26:00" ...
##  $ UnitPrice  : num [1:541909] 2.55 3.39 2.75 3.39 3.39 7.65 4.25 1.85 1.85 1.69 ...
##  $ CustomerID : num [1:541909] 17850 17850 17850 17850 17850 ...
##  $ Country    : chr [1:541909] "United Kingdom" "United Kingdom" "United Kingdom" "United Kingdom" .
```

```r
summary(online_retail)
```

```
##   InvoiceNo          StockCode         Description           Quantity
##  Length:541909      Length:541909      Length:541909      Min.   :-80995.00
##  Class :character   Class :character   Class :character   1st Qu.:     1.00
##  Mode  :character   Mode  :character   Mode  :character   Median :     3.00
##                                                           Mean   :     9.55
##                                                           3rd Qu.:    10.00
##                                                           Max.   : 80995.00
##
##   InvoiceDate                     UnitPrice           CustomerID
##  Min.   :2010-12-01 08:26:00.00   Min.   :-11062.06   Min.   :12346
##  1st Qu.:2011-03-28 11:34:00.00   1st Qu.:     1.25   1st Qu.:13953
##  Median :2011-07-19 17:17:00.00   Median :     2.08   Median :15152
##  Mean   :2011-07-04 13:34:57.16   Mean   :     4.61   Mean   :15288
##  3rd Qu.:2011-10-19 11:27:00.00   3rd Qu.:     4.13   3rd Qu.:16791
##  Max.   :2011-12-09 12:50:00.00   Max.   : 38970.00   Max.   :18287
##                                                       NA's   :135080
##    Country
##  Length:541909
##  Class :character
##  Mode  :character
##
##
##
##
```

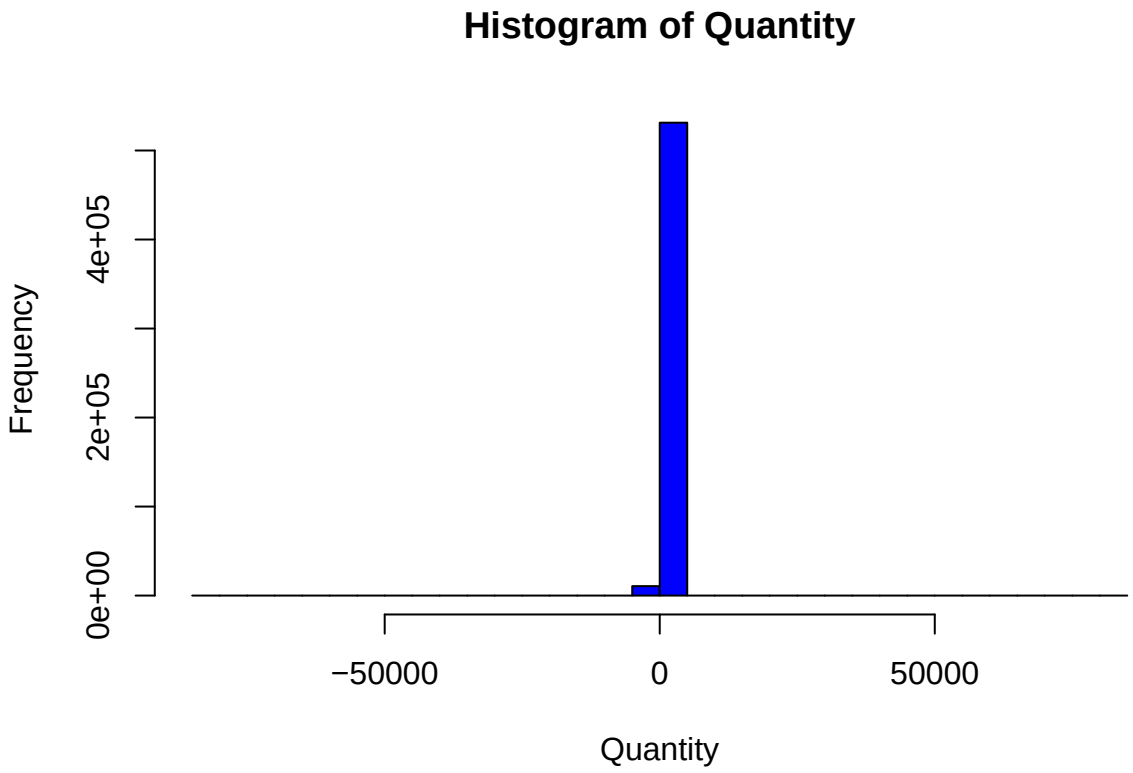
```r
missing_values <- colSums(is.na(online_retail))
print(missing_values)
```

```
##   InvoiceNo   StockCode Description    Quantity InvoiceDate   UnitPrice
##           0           0        1454           0           0           0
##  CustomerID     Country
##      135080           0
```

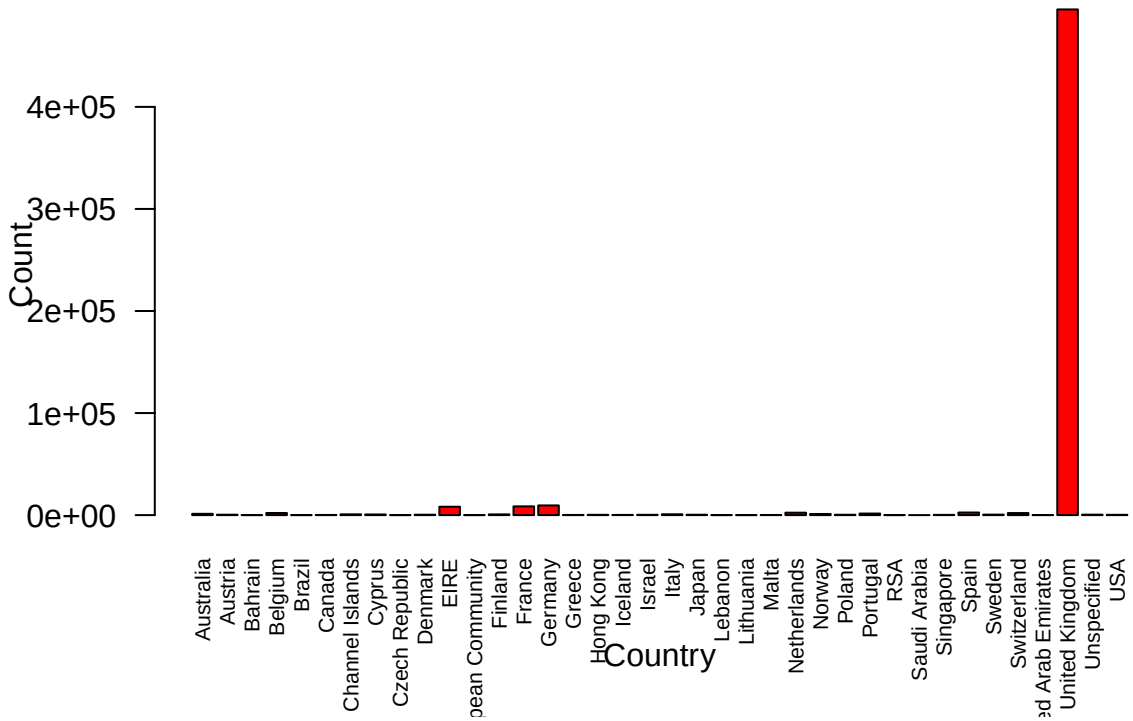
```r
head(online_retail)
```

```
## # A tibble: 6 x 8
##   InvoiceNo StockCode Description         Quantity InvoiceDate         UnitPrice
##   <chr>     <chr>     <chr>                  <dbl> <dttm>                  <dbl>
## 1 536365    85123A    WHITE HANGING HEAR~        6 2010-12-01 08:26:00      2.55
## 2 536365    71053     WHITE METAL LANTERN        6 2010-12-01 08:26:00      3.39
## 3 536365    84406B    CREAM CUPID HEARTS~        8 2010-12-01 08:26:00      2.75
## 4 536365    84029G    KNITTED UNION FLAG~        6 2010-12-01 08:26:00      3.39
## 5 536365    84029E    RED WOOLLY HOTTIE ~        6 2010-12-01 08:26:00      3.39
## 6 536365    22752     SET 7 BABUSHKA NES~        2 2010-12-01 08:26:00      7.65
## # i 2 more variables: CustomerID <dbl>, Country <chr>
```

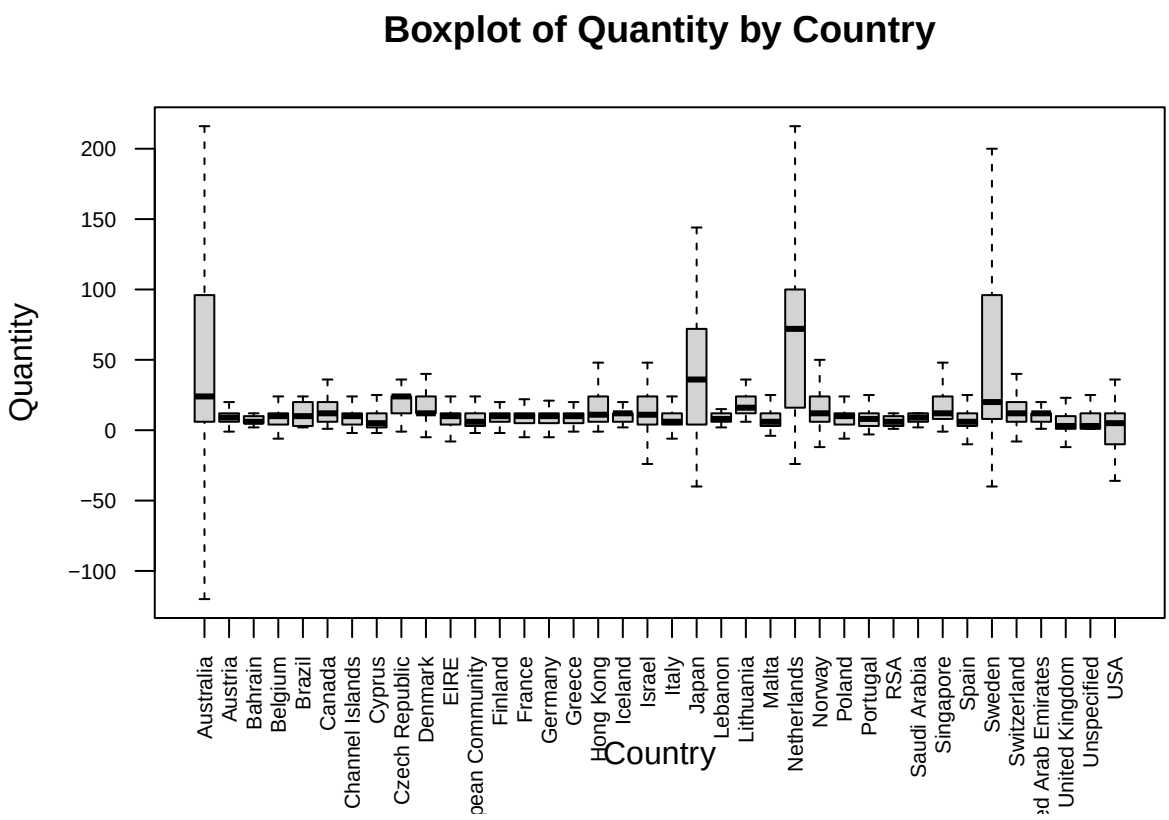
```
hist(online_retail$Quantity, main="Histogram of Quantity", xlab="Quantity", col="blue", breaks=50)
```




```
country_count <- table(online_retail$Country)
barplot(country_count, main="Bar Plot of Countries", xlab="Country", ylab="Count", col="red", las=2, ce
```



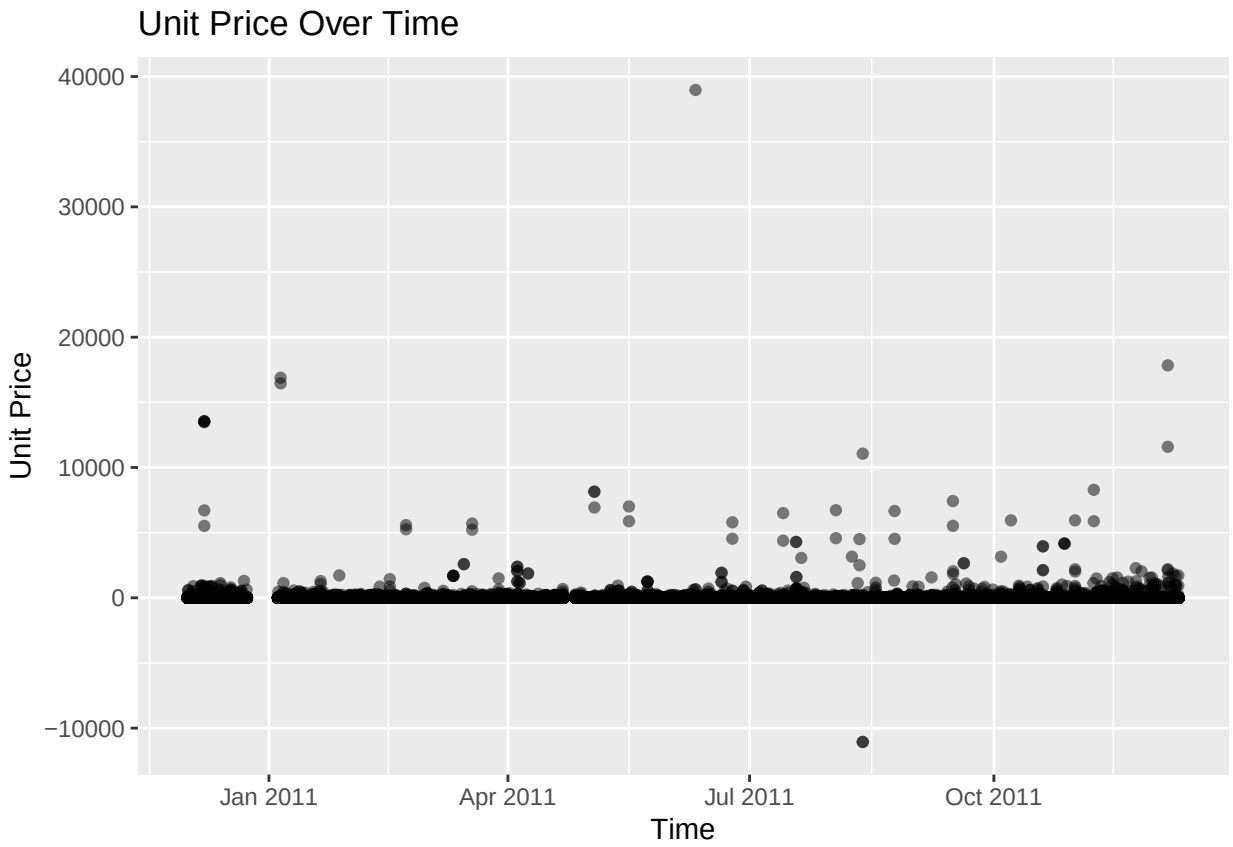

```
boxplot(Quantity ~ Country, data=online_retail, main="Boxplot of Quantity by Country", xlab="Country", y
```




Right off the bat, I'm seeing that quantity is not going to be the most useful metric for my exploration unless

I make some edits to it. It's basically aggregating ALL the items but we want to get into specific seasonal behavior. A note on as.POSIXct, a base R function that allowed me to convert the InvoiceDate column into the standard date-time format used in R. I think I did this on in one of my earlier run throughs and altered the dataset that way but I'll keep the step in since I think it's a useful tidbit! We know the UK is overrepresented here, as evident by Dante's analysis as well. To help narrow my focus, I'm going to ignore geographic behavior and instead just focus on seasonality. I ended up also playing around with time based exploration to see if I could garner anything useful but Kylie's visualizations are much more thorough in that regard. Onwards!

```
online_retail$InvoiceDate <- as.POSIXct(online_retail$InvoiceDate, format="%Y-%m-%d %H:%M:%S")

# Investigate time and price relationship
ggplot(online_retail, aes(x = InvoiceDate, y = UnitPrice)) +
  geom_point(alpha=0.5) +
  labs(title="Unit Price Over Time", x="Time", y="Unit Price")
```




I started by removing the non-numeric columns. Initially, I thought that it'd be easy to split the stockcode into items using as numeric but I realized that some of the stockcode had letters at the end that still represented goods.Thus, I needed to find a way to only exclude rows that started with non-numerics while keeping those that end with non-numerics. I then dealt with the negatives in the Quantity column. I'm guessing that we won't need them to predict seasonality especially since we're not really sure what these negatives represent. Perhaps returns? Some descriptions note it as accounting items though, like debt.

```
online_retail
```

```
## # A tibble: 541,909 x 8
##    InvoiceNo StockCode Description        Quantity InvoiceDate         UnitPrice
##    <chr>     <chr>     <chr>                 <dbl> <dttm>                  <dbl>
##  1 536365    85123A    WHITE HANGING HEA~        6 2010-12-01 08:26:00      2.55
```

```
##  2 536365    71053     WHITE METAL LANTE~        6 2010-12-01 08:26:00      3.39
##  3 536365    84406B    CREAM CUPID HEART~        8 2010-12-01 08:26:00      2.75
##  4 536365    84029G    KNITTED UNION FLA~        6 2010-12-01 08:26:00      3.39
##  5 536365    84029E    RED WOOLLY HOTTIE~        6 2010-12-01 08:26:00      3.39
##  6 536365    22752     SET 7 BABUSHKA NE~        2 2010-12-01 08:26:00      7.65
##  7 536365    21730     GLASS STAR FROSTE~        6 2010-12-01 08:26:00      4.25
##  8 536366    22633     HAND WARMER UNION~        6 2010-12-01 08:28:00      1.85
##  9 536366    22632     HAND WARMER RED P~        6 2010-12-01 08:28:00      1.85
## 10 536367    84879     ASSORTED COLOUR B~       32 2010-12-01 08:34:00      1.69
## # i 541,899 more rows
## # i 2 more variables: CustomerID <dbl>, Country <chr>
```

```
online_retail['StockCode'] <- sapply(online_retail['StockCode'], function(x) as.numeric(as.character(x)))
```

```
## Warning in FUN(X[[i]], ...): NAs introduced by coercion
```

```
online_retail
```

```
## # A tibble: 541,909 x 8
##    InvoiceNo StockCode Description        Quantity InvoiceDate         UnitPrice
##    <chr>         <dbl> <chr>                 <dbl> <dttm>                  <dbl>
##  1 536365           NA WHITE HANGING HEA~        6 2010-12-01 08:26:00      2.55
##  2 536365        71053 WHITE METAL LANTE~        6 2010-12-01 08:26:00      3.39
##  3 536365           NA CREAM CUPID HEART~        8 2010-12-01 08:26:00      2.75
##  4 536365           NA KNITTED UNION FLA~        6 2010-12-01 08:26:00      3.39
##  5 536365        84029 RED WOOLLY HOTTIE~        6 2010-12-01 08:26:00      3.39
##  6 536365        22752 SET 7 BABUSHKA NE~        2 2010-12-01 08:26:00      7.65
##  7 536365        21730 GLASS STAR FROSTE~        6 2010-12-01 08:26:00      4.25
##  8 536366        22633 HAND WARMER UNION~        6 2010-12-01 08:28:00      1.85
##  9 536366        22632 HAND WARMER RED P~        6 2010-12-01 08:28:00      1.85
## 10 536367        84879 ASSORTED COLOUR B~       32 2010-12-01 08:34:00      1.69
## # i 541,899 more rows
## # i 2 more variables: CustomerID <dbl>, Country <chr>
```

```
# This is it!
class(online_retail$StockCode)
```

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

```
pattern <- "^[A-Za-z]"
d1 <- online_retail[!grepl(pattern,online_retail$StockCode),]

d2 <- d1[d1$Quantity >=0,]
d2
```

```
## # A tibble: 531,285 x 8
##    InvoiceNo StockCode Description        Quantity InvoiceDate         UnitPrice
##    <chr>         <dbl> <chr>                 <dbl> <dttm>                  <dbl>
##  1 536365           NA WHITE HANGING HEA~        6 2010-12-01 08:26:00      2.55
##  2 536365        71053 WHITE METAL LANTE~        6 2010-12-01 08:26:00      3.39
##  3 536365           NA CREAM CUPID HEART~        8 2010-12-01 08:26:00      2.75
##  4 536365           NA KNITTED UNION FLA~        6 2010-12-01 08:26:00      3.39
##  5 536365        84029 RED WOOLLY HOTTIE~        6 2010-12-01 08:26:00      3.39
##  6 536365        22752 SET 7 BABUSHKA NE~        2 2010-12-01 08:26:00      7.65
##  7 536365        21730 GLASS STAR FROSTE~        6 2010-12-01 08:26:00      4.25
##  8 536366        22633 HAND WARMER UNION~        6 2010-12-01 08:28:00      1.85
##  9 536366        22632 HAND WARMER RED P~        6 2010-12-01 08:28:00      1.85
```

```
## 10 536367        84879 ASSORTED COLOUR B~       32 2010-12-01 08:34:00      1.69
## # i 531,275 more rows
## # i 2 more variables: CustomerID <dbl>, Country <chr>
```

Now the descriptions. We have to check if these are actually items. It looks like even those with wonky descriptions are items! But the issue now that we should think about is items that have 0 UnitPrice but a $> 0$ Quantity. What does that imply? Do we need to redo d2 to include those? My thoughts are that we should just ignore this. Let's get on with the hypothesis then. I think that Winter is going to be the busiest season for the business because it primarily sells novelty gifts and winter is traditionally when a lot of gifts are given. We're going to show the 10 most popular items sold and see if that has any bearing or correlation with the top items by season. Being able to see which of items are the best sellers definitely would help the business inform their sourcing decisions! Furthermore, the plot helps with deciding how much of each item the business should buy. For example, the top two selling items- Paper Craft, Little Birdie and Medium Ceramic Top Storage Jar- outsell all the items by a considerable margin. Perhaps the business should stock more of these. This suggestion is limited though by me not including UnitPrice in my hypotheses. High volume doesn't always neccesarily equal high profit so if I were to improve on this analysis, I'd definitely factor in the price of each item.

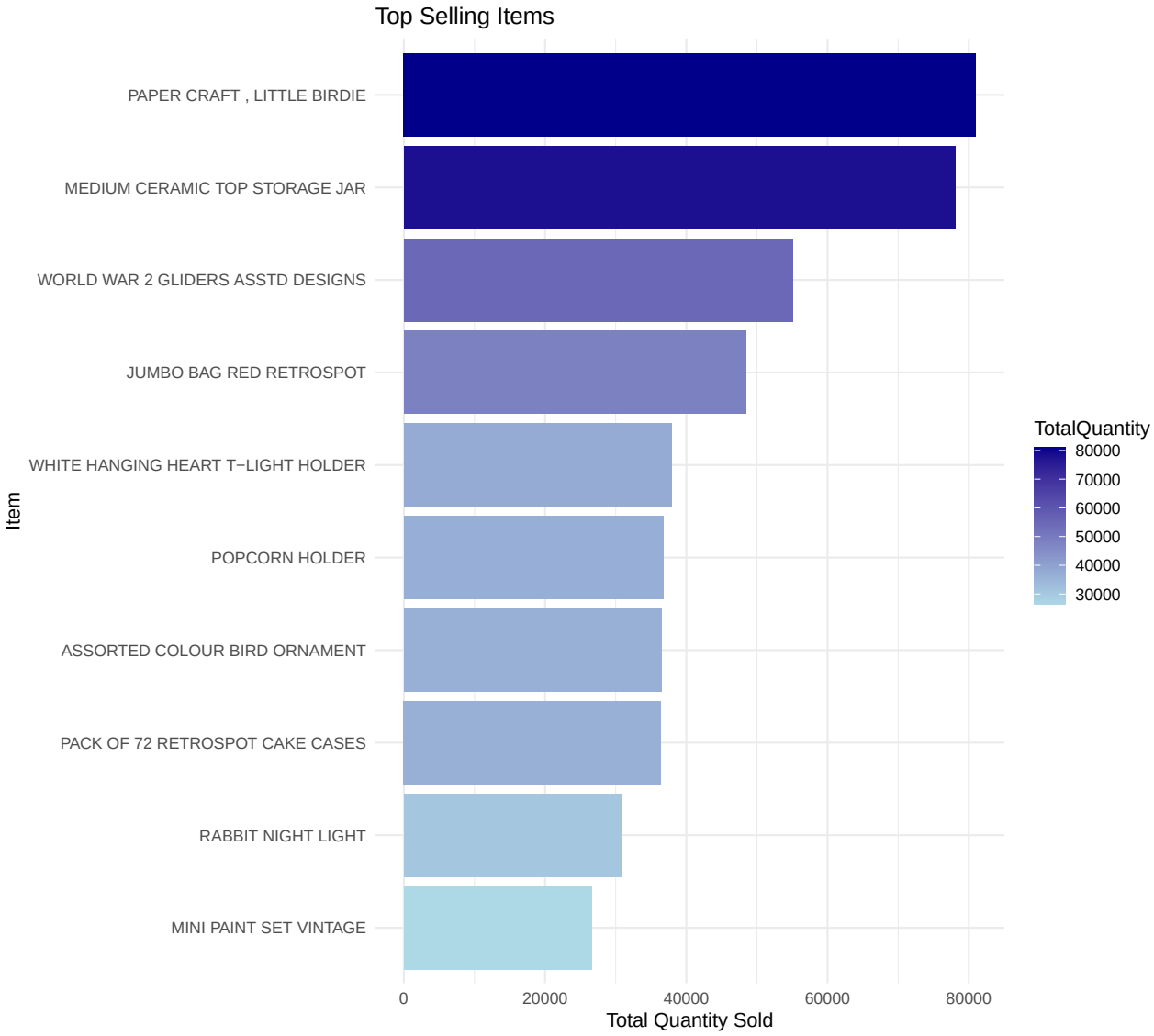
```r
library(dplyr)
library(ggplot2)
top_selling_items <- d2 %>%
  group_by(StockCode, Description) %>% summarize(TotalQuantity = sum(Quantity)) %>%
  arrange(desc(TotalQuantity))
```

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

```r
print(top_selling_items)
```

```
## # A tibble: 4,472 x 3
## # Groups:   StockCode [2,913]
##    StockCode Description                       TotalQuantity
##        <dbl> <chr>                                     <dbl>
##  1     23843 PAPER CRAFT , LITTLE BIRDIE               80995
##  2     23166 MEDIUM CERAMIC TOP STORAGE JAR            78033
##  3     84077 WORLD WAR 2 GLIDERS ASSTD DESIGNS         55047
##  4        NA JUMBO BAG RED RETROSPOT                   48478
##  5        NA WHITE HANGING HEART T-LIGHT HOLDER        37895
##  6     22197 POPCORN HOLDER                            36761
##  7     84879 ASSORTED COLOUR BIRD ORNAMENT             36461
##  8     21212 PACK OF 72 RETROSPOT CAKE CASES           36419
##  9     23084 RABBIT NIGHT LIGHT                        30788
## 10     22492 MINI PAINT SET VINTAGE                    26633
## # i 4,462 more rows
```

```r
# This will show the 10 most popular items sold .
ggplot(head(top_selling_items,10), aes(x = reorder(Description, TotalQuantity), y = TotalQuantity, fill
  geom_bar(stat = "identity") +
  scale_fill_gradient(low="lightblue", high= "darkblue") +
  theme_minimal() +
  labs(x = "Item", y = "Total Quantity Sold", title = "Top Selling Items") +
  coord_flip() # Flipping coordinates for better readability
```




Seasonality might be a bit challenging. A way to do this might be to partition the dates into four different seasons. Even though this company sells to a bunch of places around the world that might not have the same seasonal procession, let's do Spring, Summer, Fall, Winter . The data set runs from 12/1/2010 to 12/9/2011. So we'll partition accordingly- I did a quick Google search to determine what the seasonal splits by month should be. As follow: Spring-> March to June; Summer -> June to September, Fall -> September to December, Winter-> December to March. I thought I'd have to use lubridate here to do some magic to return the corresponding string but I ended up just writing a function get_season that gets the job done. We have a series of vectors that represent the date partitions outlined above, and then perform a series of checks on the month to determine which vector-and associated season- it will fall in.

```
library(lubridate)
```

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

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

```r
get_season <- function(month) {
  if (month %in% c(3, 4, 5)) {
    return("Spring")
  } else if (month %in% c(6, 7, 8)) {
    return("Summer")
  } else if (month %in% c(9, 10, 11)) {
    return("Fall")
  } else {
    return("Winter")
  }
}
```

I then created a new data frame d3, for the final stretch. I found it a good habit for myself to create new dataframes everytime I modified the data. Part of me is curious whether this affects performance for larger and larger datasets but I found it useful to rollback any changes I made to the dataframe, especially when I was tinkering around.I then grouped the seasons accordingly and then looked for the top 10 items in each season. After grouping the items, I created four bar graphs to represent the sales performance of the top 10 items in each season. This insight is pretty actionable- the firm can focus on which items to push in each season. We see a bit of correlation as well with the top 10 items sold overall, some of which appear in the top 10 items sold in each season. For example, we see that Paper Craft, Little Birdie and World War 2 Gliders Asstd Designs are the top selling items in Winter and Spring respectively.

```r
# Adding a new column for the season
d3 <- d2 %>%
  mutate(Month = month(InvoiceDate),
         Season = sapply(Month, get_season))
d3
```

```
## # A tibble: 531,285 x 10
##    InvoiceNo StockCode Description        Quantity InvoiceDate         UnitPrice
##    <chr>         <dbl> <chr>                 <dbl> <dttm>                  <dbl>
##  1 536365           NA WHITE HANGING HEA~        6 2010-12-01 08:26:00      2.55
##  2 536365        71053 WHITE METAL LANTE~        6 2010-12-01 08:26:00      3.39
##  3 536365           NA CREAM CUPID HEART~        8 2010-12-01 08:26:00      2.75
##  4 536365           NA KNITTED UNION FLA~        6 2010-12-01 08:26:00      3.39
##  5 536365        84029 RED WOOLLY HOTTIE~        6 2010-12-01 08:26:00      3.39
##  6 536365        22752 SET 7 BABUSHKA NE~        2 2010-12-01 08:26:00      7.65
##  7 536365        21730 GLASS STAR FROSTE~        6 2010-12-01 08:26:00      4.25
##  8 536366        22633 HAND WARMER UNION~        6 2010-12-01 08:28:00      1.85
##  9 536366        22632 HAND WARMER RED P~        6 2010-12-01 08:28:00      1.85
## 10 536367        84879 ASSORTED COLOUR B~       32 2010-12-01 08:34:00      1.69
## # i 531,275 more rows
## # i 4 more variables: CustomerID <dbl>, Country <chr>, Month <dbl>,
## #   Season <chr>
```

```r
# Grouping
get_top_items_by_season <- function(d3, season) {
  d3 %>%
    filter(Season == season) %>%
    group_by(StockCode, Description) %>%
    summarize(TotalQuantity = sum(Quantity)) %>%
    arrange(desc(TotalQuantity)) %>%
    head(10)
}
#
```

```r
top_items_spring <- get_top_items_by_season(d3, "Spring")
```

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

```r
top_items_summer <- get_top_items_by_season(d3, "Summer")
```

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

```r
top_items_fall <- get_top_items_by_season(d3, "Fall")
```

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

```r
top_items_winter <- get_top_items_by_season(d3, "Winter")
```

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

```r
top_items_spring
```

```
## # A tibble: 10 x 3
## # Groups:   StockCode [9]
##    StockCode Description                         TotalQuantity
##        <dbl> <chr>                                       <dbl>
##  1     84077 WORLD WAR 2 GLIDERS ASSTD DESIGNS           19076
##  2     22197 SMALL POPCORN HOLDER                        11521
##  3        NA JUMBO BAG RED RETROSPOT                     11359
##  4     21212 PACK OF 72 RETROSPOT CAKE CASES             10460
##  5        NA WHITE HANGING HEART T-LIGHT HOLDER           9878
##  6     15036 ASSORTED COLOURS SILK FAN                    9754
##  7     21977 PACK OF 60 PINK PAISLEY CAKE CASES           9297
##  8     84879 ASSORTED COLOUR BIRD ORNAMENT                8282
##  9     22616 PACK OF 12 LONDON TISSUES                    7347
## 10     22178 VICTORIAN GLASS HANGING T-LIGHT              7233
```

```r
top_items_summer
```

```
## # A tibble: 10 x 3
## # Groups:   StockCode [9]
##    StockCode Description                         TotalQuantity
##        <dbl> <chr>                                       <dbl>
##  1        NA JUMBO BAG RED RETROSPOT                     12313
##  2     84879 ASSORTED COLOUR BIRD ORNAMENT                9962
##  3     15036 ASSORTED COLOURS SILK FAN                    8259
##  4     84077 WORLD WAR 2 GLIDERS ASSTD DESIGNS            8145
##  5     22197 POPCORN HOLDER                               8129
##  6     21212 PACK OF 72 RETROSPOT CAKE CASES              7758
##  7        NA WHITE HANGING HEART T-LIGHT HOLDER           6801
##  8     84568 GIRLS ALPHABET IRON ON PATCHES               6693
##  9     47566 PARTY BUNTING                                6674
## 10     23307 SET OF 60 PANTRY DESIGN CAKE CASES           6575
```

```r
top_items_fall
```

```
## # A tibble: 10 x 3
## # Groups:   StockCode [10]
```

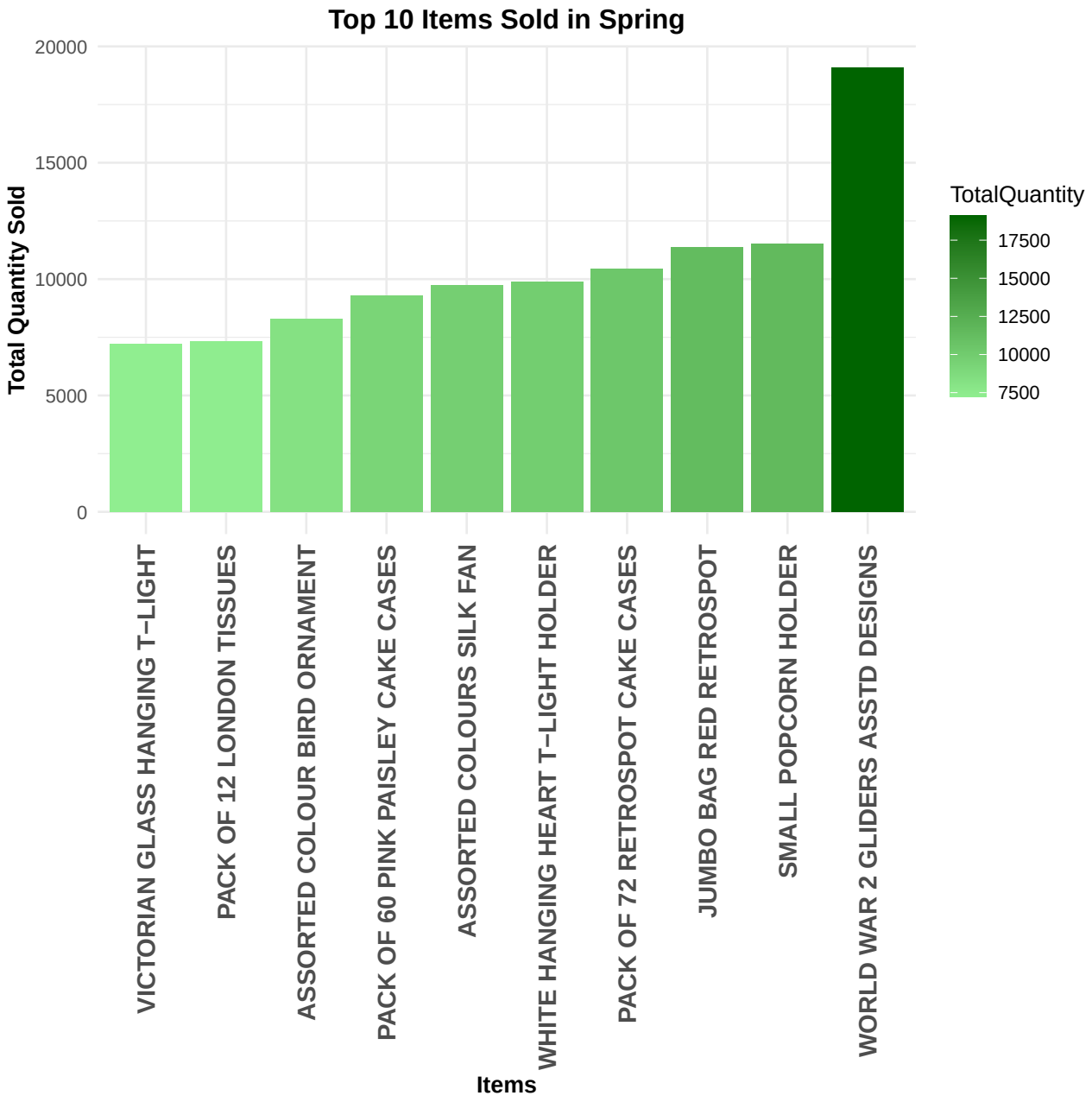
```
##    StockCode Description                       TotalQuantity
##        <dbl> <chr>                                     <dbl>
##  1     22197 POPCORN HOLDER                          22415
##  2     23084 RABBIT NIGHT LIGHT                      21492
##  3     84077 WORLD WAR 2 GLIDERS ASSTD DESIGNS       16254
##  4        NA JUMBO BAG RED RETROSPOT                 15722
##  5     22086 PAPER CHAIN KIT 50'S CHRISTMAS          13443
##  6     84826 ASSTD DESIGN 3D PAPER STICKERS          12562
##  7     22952 60 CAKE CASES VINTAGE CHRISTMAS         11869
##  8     84879 ASSORTED COLOUR BIRD ORNAMENT           11078
##  9     21915 RED  HARMONICA IN BOX                    9984
## 10     22492 MINI PAINT SET VINTAGE                   9776
```

```
top_items_winter
```

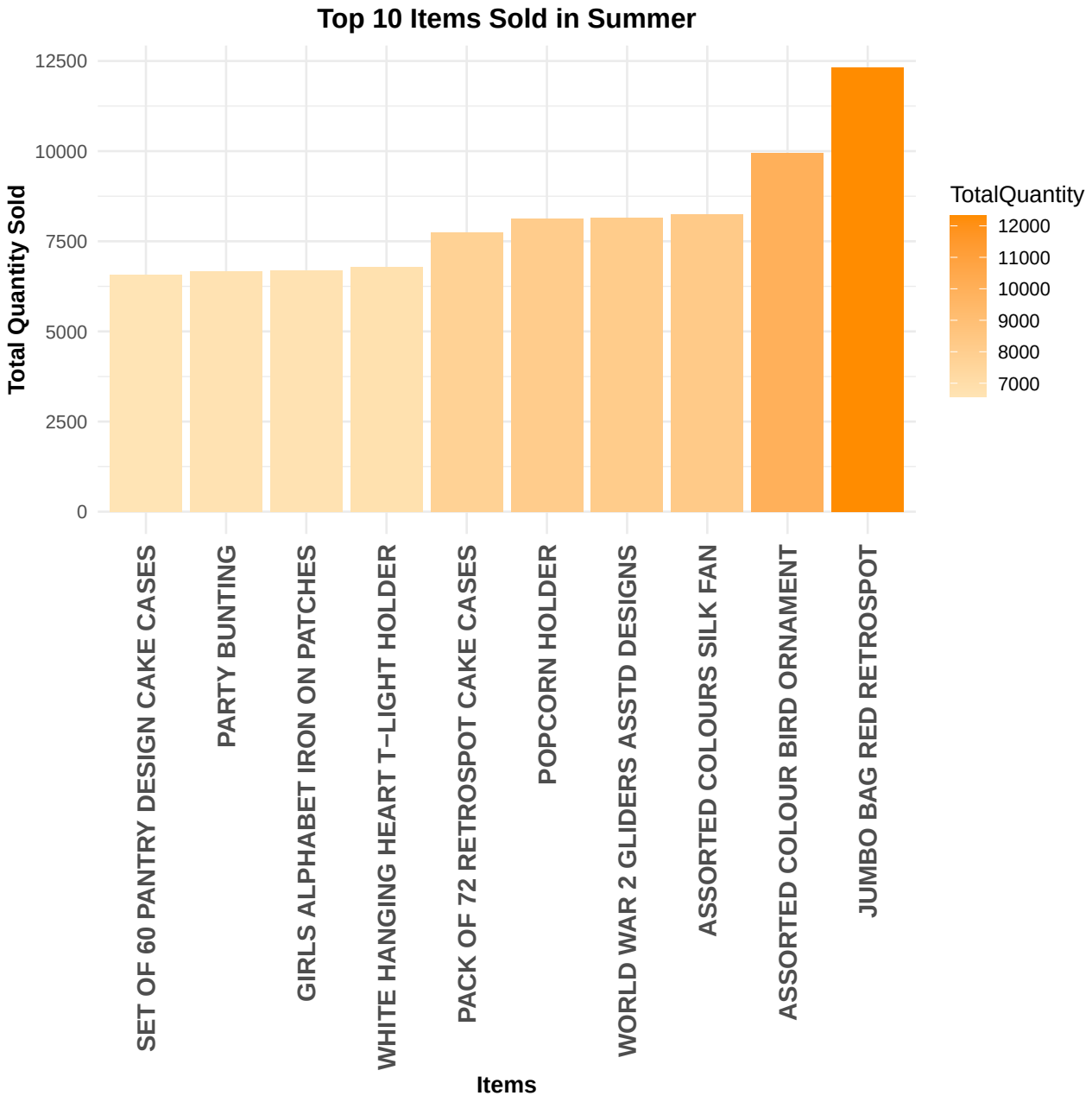
```
## # A tibble: 10 x 3
## # Groups:   StockCode [9]
##    StockCode Description                        TotalQuantity
##        <dbl> <chr>                                     <dbl>
##  1     23843 PAPER CRAFT , LITTLE BIRDIE             80995
##  2     23166 MEDIUM CERAMIC TOP STORAGE JAR          74422
##  3        NA WHITE HANGING HEART T-LIGHT HOLDER      12067
##  4     84077 WORLD WAR 2 GLIDERS ASSTD DESIGNS       11572
##  5     21212 PACK OF 72 RETROSPOT CAKE CASES         10071
##  6        NA JUMBO BAG RED RETROSPOT                  9084
##  7     22197 SMALL POPCORN HOLDER                     7228
##  8     84879 ASSORTED COLOUR BIRD ORNAMENT            7139
##  9     22693 GROW A FLYTRAP OR SUNFLOWER IN TIN       7125
## 10     22616 PACK OF 12 LONDON TISSUES                6507
```

```
create_season_graph <- function(season_data, season_name) {
  # Define gradient colors for each season
  gradient_colors <- list(
    "Winter" = c("lightblue", "blue"),
    "Spring" = c("lightgreen", "darkgreen"),
    "Summer" = c("moccasin", "darkorange"),
    "Fall" = c("burlywood", "brown")
  )

  ggplot(season_data, aes(x = reorder(Description, TotalQuantity), y = TotalQuantity, fill = TotalQuanti
    geom_bar(stat = "identity") +
    scale_fill_gradient(low = gradient_colors[[season_name]][1], high = gradient_colors[[season_name]][
    labs(title = paste("Top 10 Items Sold in", season_name),
         x = "Items",
         y = "Total Quantity Sold") +
    theme_minimal() +
    theme(plot.title = element_text(hjust = 0.5, face = "bold"),
          axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1, size = 12, face = "bold"),
          axis.title = element_text(face = "bold"),
          plot.margin = margin(1, 1, 1, 1, "cm"))
}


graph_spring <- create_season_graph(top_items_spring, "Spring")
graph_summer <- create_season_graph(top_items_summer, "Summer")
```

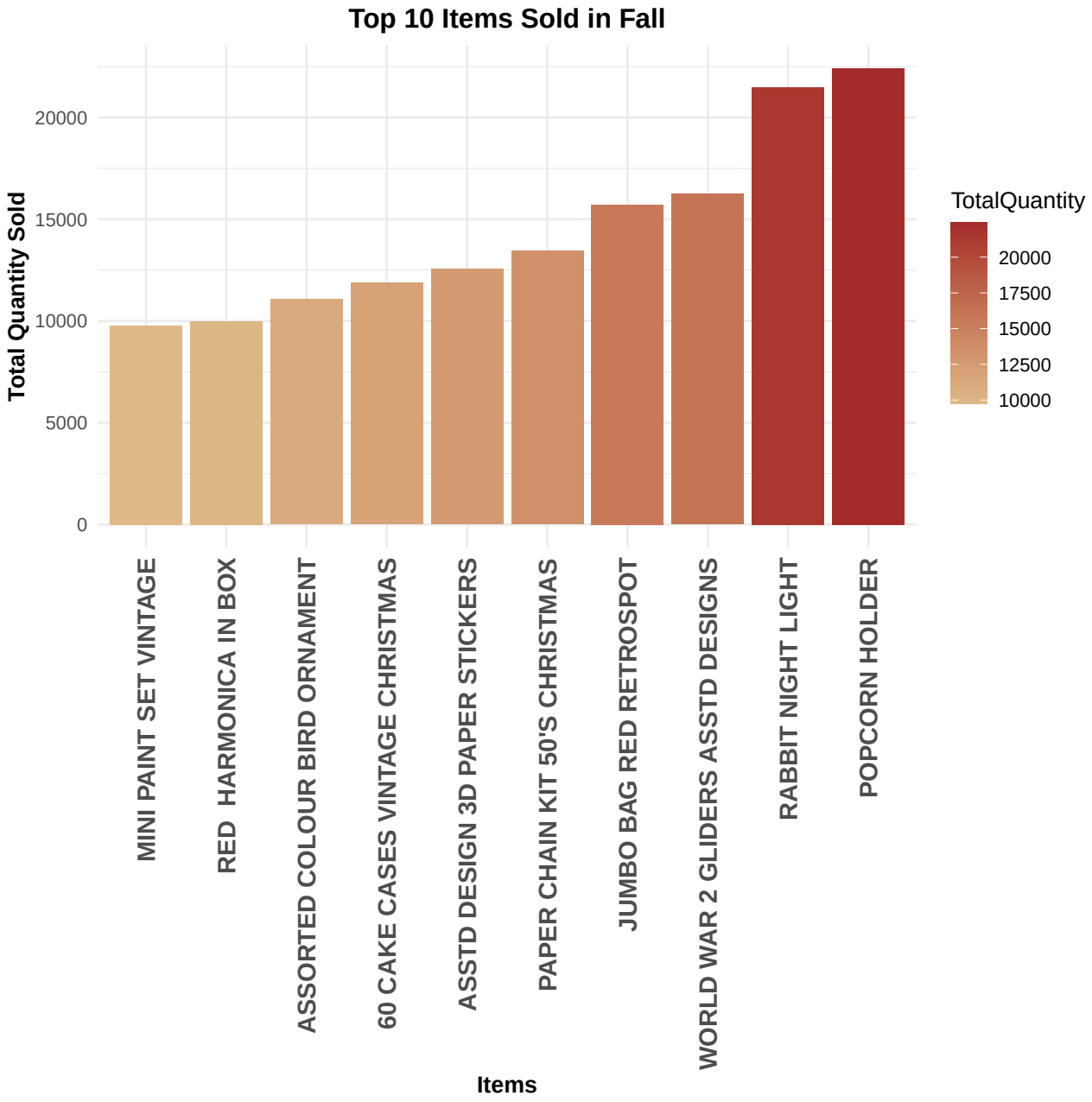
```
graph_fall <- create_season_graph(top_items_fall, "Fall")
graph_winter <- create_season_graph(top_items_winter, "Winter")

graph_spring
```



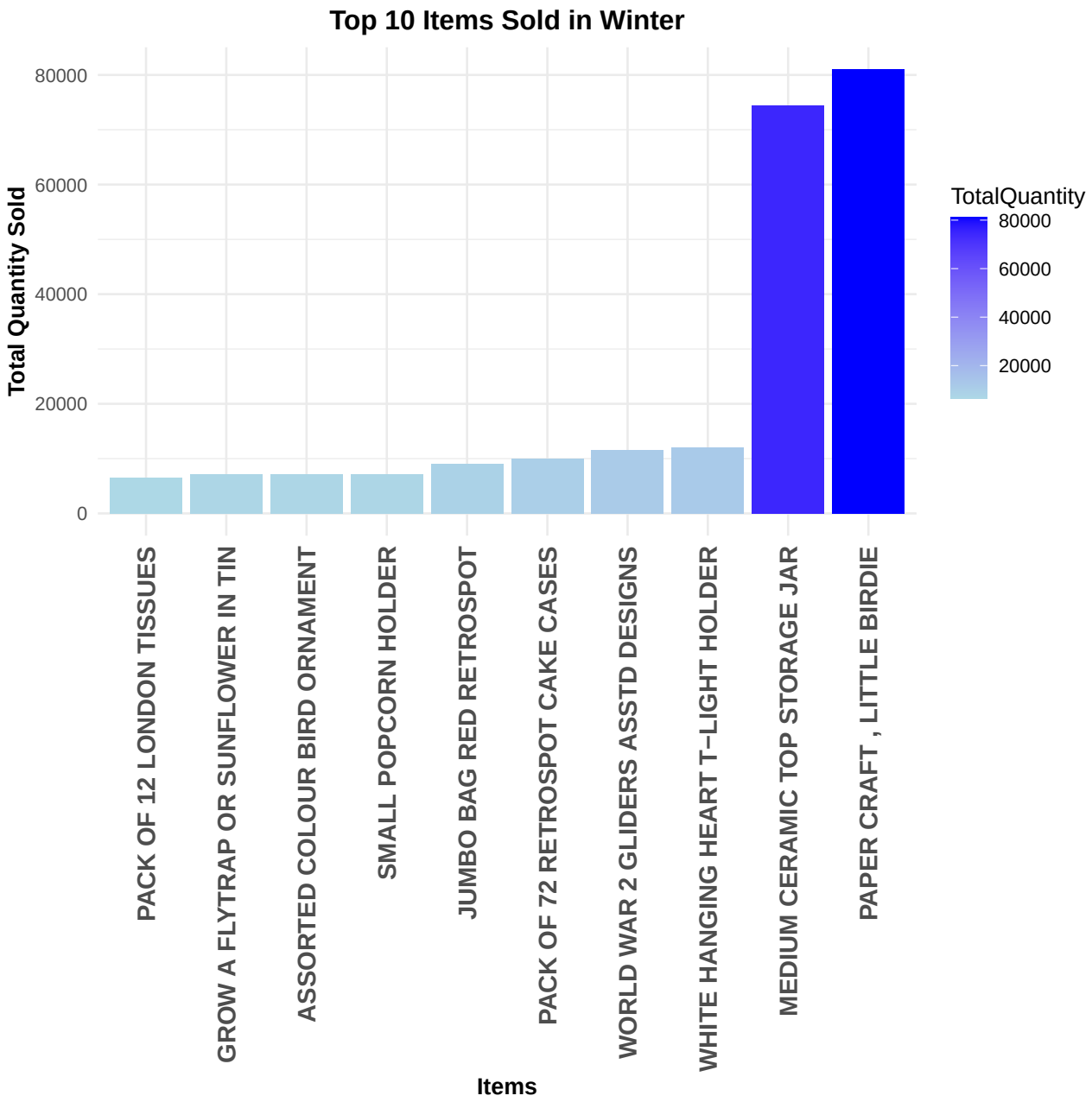

```
graph_summer
```


Top 10 Items Sold in Summer
Total Quantity Sold
12500
10000
7500
5000
2500
0
SET OF 60 PANTRY DESIGN CAKE CASES
PARTY BUNTING
GIRLS ALPHABET IRON ON PATCHES
WHITE HANGING HEART T−LIGHT HOLDER
PACK OF 72 RETROSPOT CAKE CASES
POPCORN HOLDER
WORLD WAR 2 GLIDERS ASSTD DESIGNS
ASSORTED COLOURS SILK FAN
ASSORTED COLOUR BIRD ORNAMENT
JUMBO BAG RED RETROSPOT
Items
TotalQuantity
12000
11000
10000
9000
8000
7000

```
graph_fall
```




```
graph_winter
```

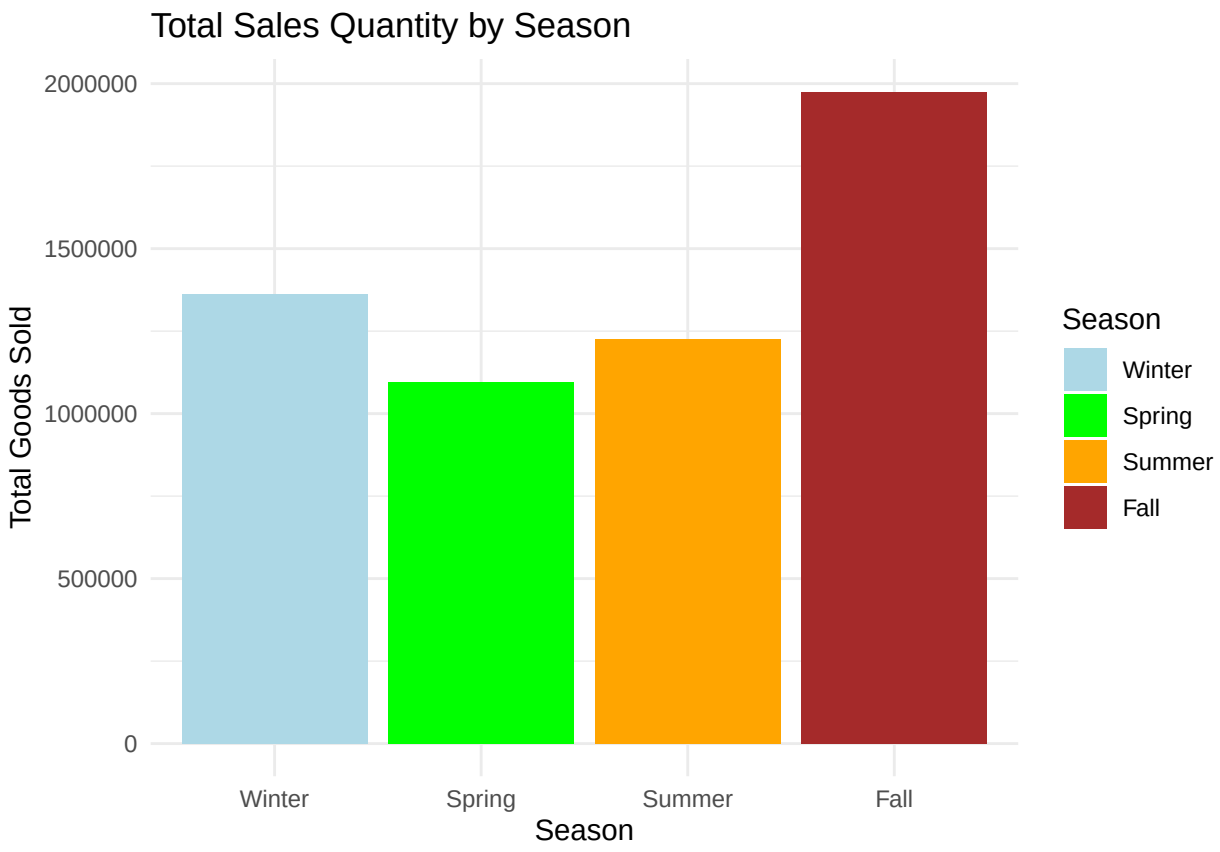



But here's another useful insight. Now that we know the seasonality of items, we can also look at the performance of the seasons themselves. Looking at the plots above, it's pretty clear that some seasons just have more goods sold then others. I decided to visualize this through a bar chart and a pie graph. The bar chart was way more useful as it highlighted the numbers pretty clearly while the pie graph was effective at highlighting which were the busier seasons, did not do as good at job at directly comparing the quantities sold. We see that Fall and Winter are, by a huge margin, the busiest seasons for the firm with significantly more goods sold. Our hypothesis is actually proven to be incorrect as Fall was the busiest season for the business. I believe that this is because the business is a wholesaler as opposed to something oriented to the general consumer. Since direct consumer gift shops would likely forecast their demand in the season(s) before and then purchase accordingly, winter would not be the prime period for this business. It's still a busy season though, just not as much as fall is- smaller shops or the occasional late requisition might come through. This perspective supports why Spring and Summer aren't as busy as well since most purchasers are probably in their demand forecasting stages. With this information, the business can think about workforce requirements as well as securing additional logisitical support or whatnot for the seasons that are the busiest!

```r
# Let's now see which season has the highest number of goods sold.
seasonal_sales <- d3 %>%
  group_by(Season) %>%
  summarize(TotalQuantity = sum(Quantity)) %>%
  arrange(desc(TotalQuantity))
highest_season <- head(seasonal_sales, 1)
highest_season
```

```
## # A tibble: 1 x 2
##   Season TotalQuantity
##   <chr>          <dbl>
## 1 Fall         1975759
```

```r
seasonal_sales$Season <- factor(seasonal_sales$Season, levels = c("Winter", "Spring", "Summer", "Fall"))
season_colors <- c("Winter" = "lightblue", "Spring" = "green", "Summer" = "orange", "Fall" = "brown")

ggplot(seasonal_sales, aes(x = Season, y = TotalQuantity, fill = Season)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = season_colors) +
  theme_minimal() +
  labs(x = "Season", y = "Total Goods Sold", title = "Total Sales Quantity by Season")
```




```r
ggplot(seasonal_sales, aes(x = "", y = TotalQuantity, fill = Season)) +
  geom_bar(stat = "identity", width = 1) +
  coord_polar("y", start = 0) +
  scale_fill_manual(values = season_colors) +
  theme_minimal() +
  theme(axis.line = element_blank(),
```

```
        axis.text = element_blank(),
        axis.ticks = element_blank(),
        panel.grid = element_blank(),
        plot.title = element_text(hjust = 0.5, face = "bold", size = 14),
        legend.title = element_text(face = "bold"),
        legend.position = "bottom",
        plot.margin = margin(1, 1, 1, 1, "cm")) +
  labs(fill = "Season", title = "Total Sales Quantity by Season", x = NULL, y = NULL)
```


```
# A pie chart isn't at good as highlighting the values
```

My final bit was to try to predict demand for the years to come. This ended up being quite challenging for me and I so I ended up using a very simple moving average forecast as supposed to something more substantial. There are limitations with this methodology as well as it ignores any sort of seasonal trends and basically just provides a flat line representing the average of the last observations. I was also limited by the number of periods in this dataset as it only collected a year's worth of observations. I think that with a more robust dataset, we'd be able to see the seasonal behavior of fall and winter being the busiest seasons reflected in any such demand forecast. A task for another time, I suppose- or another curious student!

```
# An attempt at a prediction
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
install.packages("dplyr", repos = "http://cran.us.r-project.org")
```

```
##
## The downloaded binary packages are in
##  /var/folders/tz/8s9ky1451fxdd68ykql0n2880000gn/T//Rtmpc9KfV2/downloaded_packages
```

```r
library(dplyr)
data_ts <- d3 %>%
  mutate(Month = floor_date(InvoiceDate, "month")) %>%
  group_by(Month) %>%
  summarize(TotalQuantity = sum(Quantity)) %>%
  ungroup()
ts_data <- ts(data_ts$TotalQuantity, start = c(2010, 12), frequency = 12)
ma_fit <- stats::filter(ts_data, rep(1/3, 3), sides = 2)
ma_extension <- rep(tail(ma_fit, 1), 12)  # Extend for 12 months into 2012

full_ma <- c(ma_fit, ma_extension)
full_ts <- ts(full_ma, start = c(2010, 12), frequency = 12)

# Plotting
plot(full_ts, main = "Extended Moving Average Forecast", xlab = "Time", ylab = "Total Quantity", col =
```