Assignment 1

Lee Zhen Xuan 31860532

Section A

Question 1

```
> #Q1
> sales = read.csv("sales.csv")
> products = read.csv("products.csv")
> inventory = read.csv("inventory.csv")
> #Sales transactions in 2020
> sales_separated = separate(sales, Date, into = c("Year", "Month", "Date"), sep= "-")
> nrow(sales_separated %>% filter(Year == 2020))
[1] 50032
> #there are 50032 sales transactions in 2020
>
```

Firstly, the sales dataset is filtered to only include the year 2020, then the number of rows counted to count the number of transactions that occurred in 2020.

There are 50032 sales transactions that occurred in 2020.

Question 2

```
> #Q2
> sales_revenue = sales %>% group_by(ProductId) %>% mutate(Revenue = UnitPrice * Quantity) %>% summarise(Total_Revenue = sum(Revenue))
>
> #Finding the ProductId with the highest total revenue
> sales_revenue[which.max(sales_revenue$Total_Revenue),]
# A tibble: 1 × 2
  ProductId Total_Revenue
      <int>         <dbl>
1       248       223612.
> productid_highest_revenue = sales_revenue[which.max(sales_revenue$Total_Revenue),]$ProductId
> productid_highest_revenue
[1] 248
> #the ProductId with the highest total revenue is 248, with total revenue of 223612
>
> #Finding the ProductName with the highest total revenue
> product_highest_revenue = (products %>% filter(ProductId == productid_highest_revenue))$ProductName
> product_highest_revenue
[1] "Sultanas"
> #The ProductName with the highest revenue is Sultanas
>
> #Finding the StoreId of the highest revenue
> storename_highest_revenue = (inventory %>% filter(ProductId == productid_highest_revenue))$StoreId
> storename_highest_revenue
[1] 21724
> #The StoreId of the highest revenue is 21724
```

The total revenue for each product is calculated by summing up the total UnitPrice * Quantity for each ProductID. Then, the which.max function, which returns the location of the first maximum value is used to find the highest total revenue of the ProductID.

The ProductID with the highest total revenue is 248.

The “products” dataset is filtered to find the ProductID 248, to search for the ProductName column. **The ProductName with the highest total revenue is “Sultanas”**

The “inventory” dataset is filtered to find the ProductID 248, to search for the StoreId column. **The StoreID with the highest total revenue is 21724.**

Question 3

```
> #Q3
> #Finding the Average Quantity Available for each StoreId
> average_quantity_avail = inventory %>% group_by(StoreId) %>% summarise(AverageQuantityAvailable = mean(QuantityAvailable))
>
> #Finding the store id with the lowest average quantity available
> lowest_average_storeid = average_quantity_avail[which.min(average_quantity_avail$AverageQuantityAvailable),]$StoreId
> lowest_average_storeid
[1] 22914
> #the StoreId with the lowest average quantity available is 22914
```

The average QuantityAvailable is produced by taking the mean of the QuantityAvailable. Then, the “which.min” function, which returns the location of the first minimum value of the average QuantityAvailable is used to find the StoreId with the lowest average quantity available. **The StoreId with the lowest average quantity available is 22914.**

Question 4

```
> #Q4
> sales_category = sales %>% mutate(SalesCategory = case_when(
+   Quantity >= 50 ~ "High",
+   Quantity >= 20 & Quantity <= 49 ~ "Medium",
+   Quantity < 20 ~ "Low"
+ ))
> sales_category
    SalesId StoreId ProductId       Date UnitPrice Quantity SalesCategory
1     82319   22726       590 2019-12-02    0.0525       93          High
2     15022   21754       390 2017-11-19    5.1100       28        Medium
3     11624   71053       883 2020-07-13    7.3675       33        Medium
4     63101   22914       658 2019-05-12    2.0825       76          High
5     29702   22623       632 2020-07-20    0.6475        8           Low
6     35660   22749       170 2019-09-30    5.6700       93          High
```

```
> #Finding the count of "High" category sales
> nrow(sales_category %>% filter(SalesCategory == "High"))
[1] 101742
> #The count of "High" category sales is 101742
```

A new column called “SalesCategory” is created by using the “case_when” function to categorize the Quantity based on its values. Then, the new dataset is filtered with the

sales category “High”, and the number of rows is counted to find the count of “High” category sales.

The count of “High” category sales is 101742.

Question 5

```
> #Q5
> products_arranged = products %>% arrange(desc(ProductCost))
> products_separate = separate(products_arranged, ProductName, into = c("Product", "Brand"), sep = " - ")
Warning messages:
1: Expected 2 pieces. Additional pieces discarded in 26 rows [19, 55, 80, 134, 173, 314, 354, 395, 438, 443, 482, 534, 547, 563, 667, 672,
687, 721, 732, 761, ...].
2: Expected 2 pieces. Missing pieces filled with `NA` in 242 rows [2, 4, 7, 11, 20, 22, 30, 34, 47, 49, 53, 56, 59, 62, 67, 70, 74, 75, 78,
86, ...].
> products_separate[3,]$Brand
[1] "Rye"
> #The Brand of the third most expensive product is "Rye"
```

The product information dataset is arranged using the function “arrange” in descending order of ProductCost. The ProductName is separated into 2 new columns, “Product” and “Brand” with the “-“ separator. The third row of the arranged dataset is the **third most expensive product, which is “Rye”.**

Question 6

```
> #Q6
> sales_summary = sales %>% group_by(ProductId,Date) %>% arrange(ProductId, Date) %>% summarise(AveragePrice = mean(UnitPrice),
AverageQuantity = mean(Quantity))
`summarise()` has grouped output by 'ProductId'. You can override using the `.groups` argument.
>
> sales_summary <- sales_summary %>%
+   group_by(ProductId) %>%
+   mutate(PricePercentageChange = ifelse(ProductId == lag(ProductId), (AveragePrice - lag(AveragePrice)) / lag(AveragePrice) *
100, NA),
+          QuantityPercentageChange = ifelse(ProductId == lag(ProductId), (AverageQuantity - lag(AverageQuantity)) / lag(Averag
eQuantity) * 100, NA))
>
> sales_summary <- sales_summary %>%
+   mutate(PriceElasticity = ifelse(PricePercentageChange == 0, 0, (PricePercentageChange / QuantityPercentageChange)))
>
```

A new dataset has been created by grouping and arranging the “sales” dataset by ProductId and Date, and the AveragePrice and AverageQuantity is calculated using the “mean” function. Next, the percentage changes between consecutive time periods of price and quantity are calculated by finding the change between the previous and next date of each row. The “lag” function is used to retrieve the value from the previous row. Finally, the price elasticity is calculated by dividing the price percentage change in response to the quantity percentage change. If the price percentage change is 0, then the price elasticity is 0.

Section B

Question 1

```
> #Q1
> transaction_category = function(X,Y){
+   tps = Y/X
+ if(tps >3500 ){
+   category = "High"
+ }else if(tps >=2500 && tps <= 3500) {
+   category = "Medium"
+ }else if(tps <2500){
+   category = "Low"
+ }
+   paste("Category: ", category)
+ }
> #From the video, Starbucks operates 30,000 stores worldwide and produces 100 million transactions per week
>
> transaction_category(30000,100)
[1] "Category:  Low"
```

From the video, Starbucks operates 30,000 stores worldwide and produces about 100 million transactions per week.

A function is created using conditional if else statements to categorize the transactions per store per week. By taking in the values X and Y, the transaction level of Starbucks stores is “Low”.

Question 2

```
> #Q2
> #Five variables: population, income_levels, traffic, competitor_presence, proximity_to_other_starbucks_locations
> store_location_category = function(population, income_levels, traffic, competitor_presence, proximity){
+   if (population >= 5000000 && income_levels >= 12000 && traffic %in% c("High", "Medium") && competitor_presence == "Low" && proximity == "Far") {
+     category = "Ideal location"
+   } else if (population >= 1000000 && income_levels >= 11000 && traffic %in% c("High", "Medium") && competitor_presence == "Low" && proximity == "Moderate") {
+     category - "Good location"
+   } else if (population >= 400000 && income_levels >= 5000 && traffic %in% c("Medium", "Low") && competitor_presence == "Medium" && proximity == "Close") {
+     category = "Average location"
+   } else {
+     category = "Poor location"
+   }
+   paste("Location Category: ", category)
+ }
>
> store_location_category(4000000, 12500, "Medium" ,"Low", "Close")
[1] "Location Category:  Poor location"
```

The variables from the video are population, income_levels, traffic, competitor presence, and proximity to other starbucks locations.

By creating a function using if else statements for the variables above, we can categorize the level of location.

From the example output above, we have taken in the values:

Population: 4000000

Income levels: 12500

Traffic: Medium

Competitor presence: Low

Proximity to other Starbucks locations: Close

The location category output is “Poor Location”.

Question 3

```
> #Q3
> #Two local factors: weather, time_of_day
> products_promotion = function(weather, time_of_day){
+   if (weather == "Cold"){
+     promotion = "Promote hot beverages"
+   }else if (weather == "Sunny"){
+     promotion = "Promote cold beverages"
+   }else if (weather == "Normal"){
+     if (time_of_day == "Morning"){
+       promotion == "Promote hot beverages"
+     }else if (time_of_day == "Afternoon"){
+       promotion == "Promote cold beverages"
+     }else if (time_of_day == "Evening"){
+       promotion == "Promote pastries"
+     }
+   }else{
+     promotion == "No promotion"
+   }
+
+   paste("Promotion: ", promotion)
+ }
>
> products_promotion("Sunny", "Morning")
[1] "Promotion:  Promote cold beverages"
```

The two local factors mentioned in the video are weather and time of the day.

A nested if else conditional statement is created to promote a specific Starbucks product. Firstly, based on the weather conditions, we can choose to promote either hot

or cold beverages. Then, if the weather conditions if normal, we can choose to promote either hot or cold beverages, or to promote pastries. If the weather conditions and time of day do not fall in these categories, then there is no promotion.

From the example above, we can see that on a Sunny weather and Morning time of day, the promotion is to promote cold beverages.