Project 1 Markdown

Sean Hersee, Gregory Miller, Sree Prabhav Bandakavi, Michael Puchalski

2025-03-23

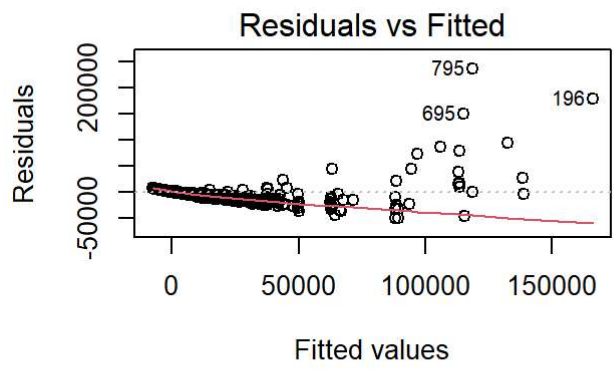
Price against Carat Linear Regression

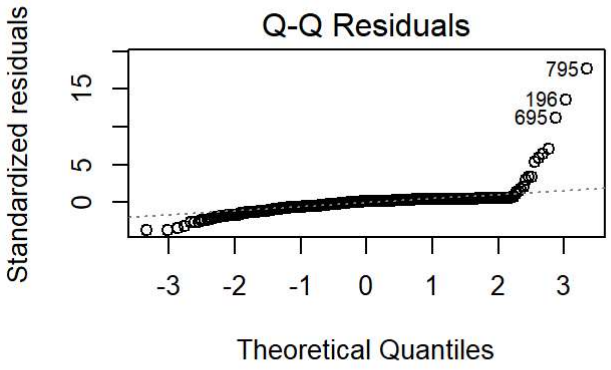
Step 1: Simple Linear Regression & Plot

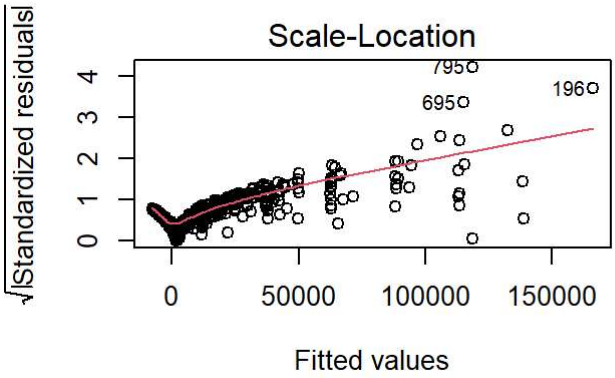
```
slr<-lm(data$price~data$carat,data=data)
ggplot2::ggplot(data, aes(x=carat, y=price))+
  geom_point()+
  geom_smooth(method=lm)+
  labs(x="Carat Weight",y="Price",title="Scatterplot of Sales Price against Carat Weight")+
  scale_y_continuous(labels = dollar_format())
```

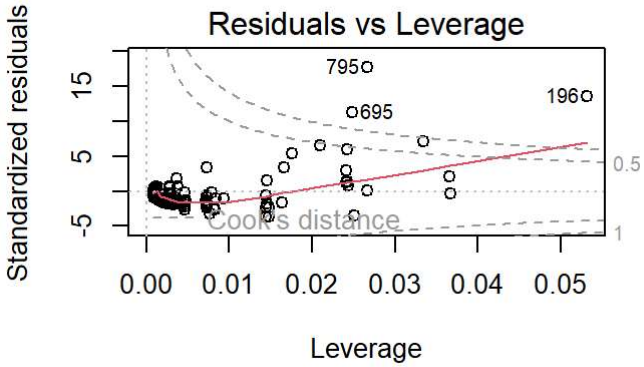
```
## `geom_smooth()` using formula = 'y ~ x'
```


```
par(mfrow = c(2, 2))
plot(slr)
```





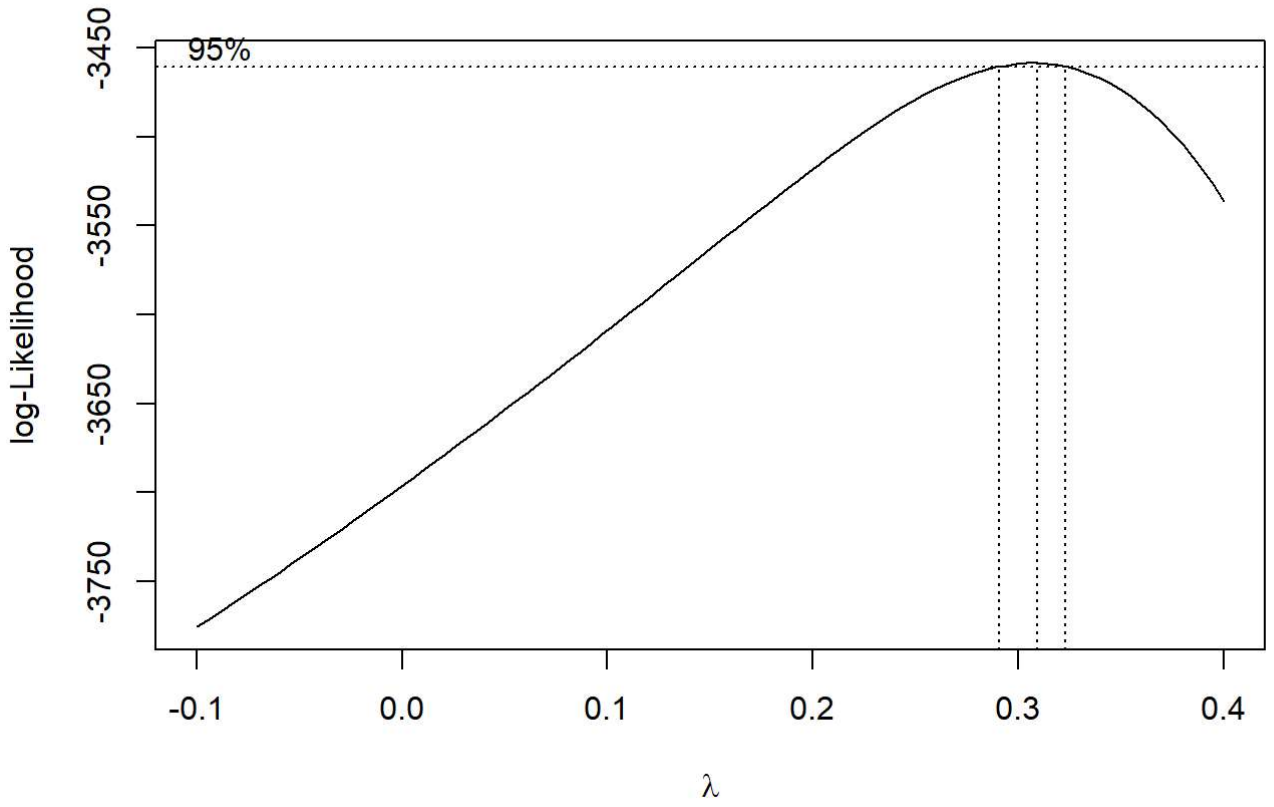




Assumption 1 is not met: Points are not evenly scattered on either side of the line on the scatterplot
Assumption 2 is not met: The average value of residuals (red) line on the Residuals Plot is not along the horizontal axis 0 and the vertical spread of the data points varies as we move left to right on the chart

Step 2: Determine Transformation of the response variable

```
MASS::boxcox(slr, lambda = seq(-.1, .4, 1/10))
```

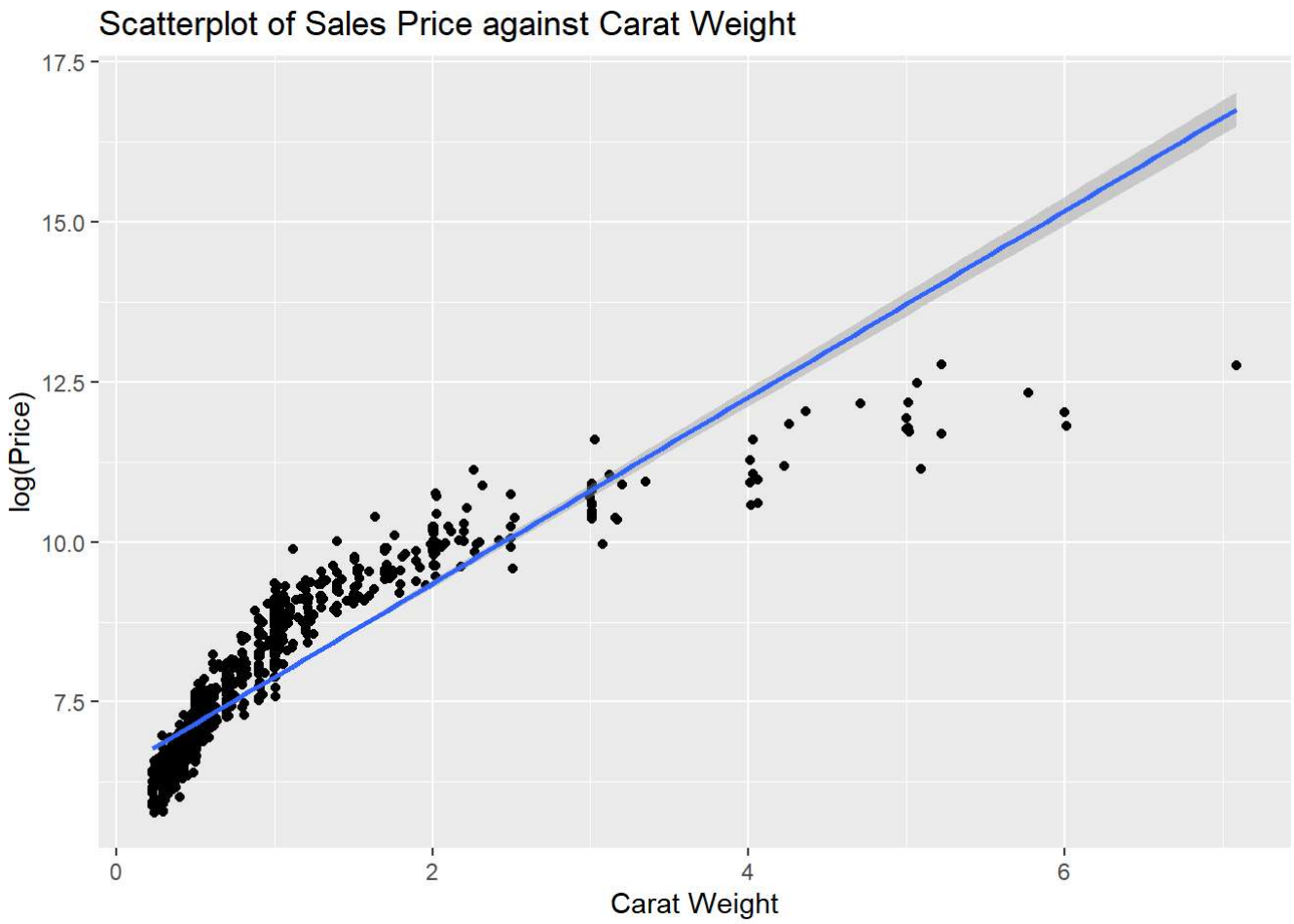


Although Lambda of 0 is not within the bounds of the confidence interval, I will choose a Lambda of 0 as that will allow for a log transformation which will lead to comprehensible results

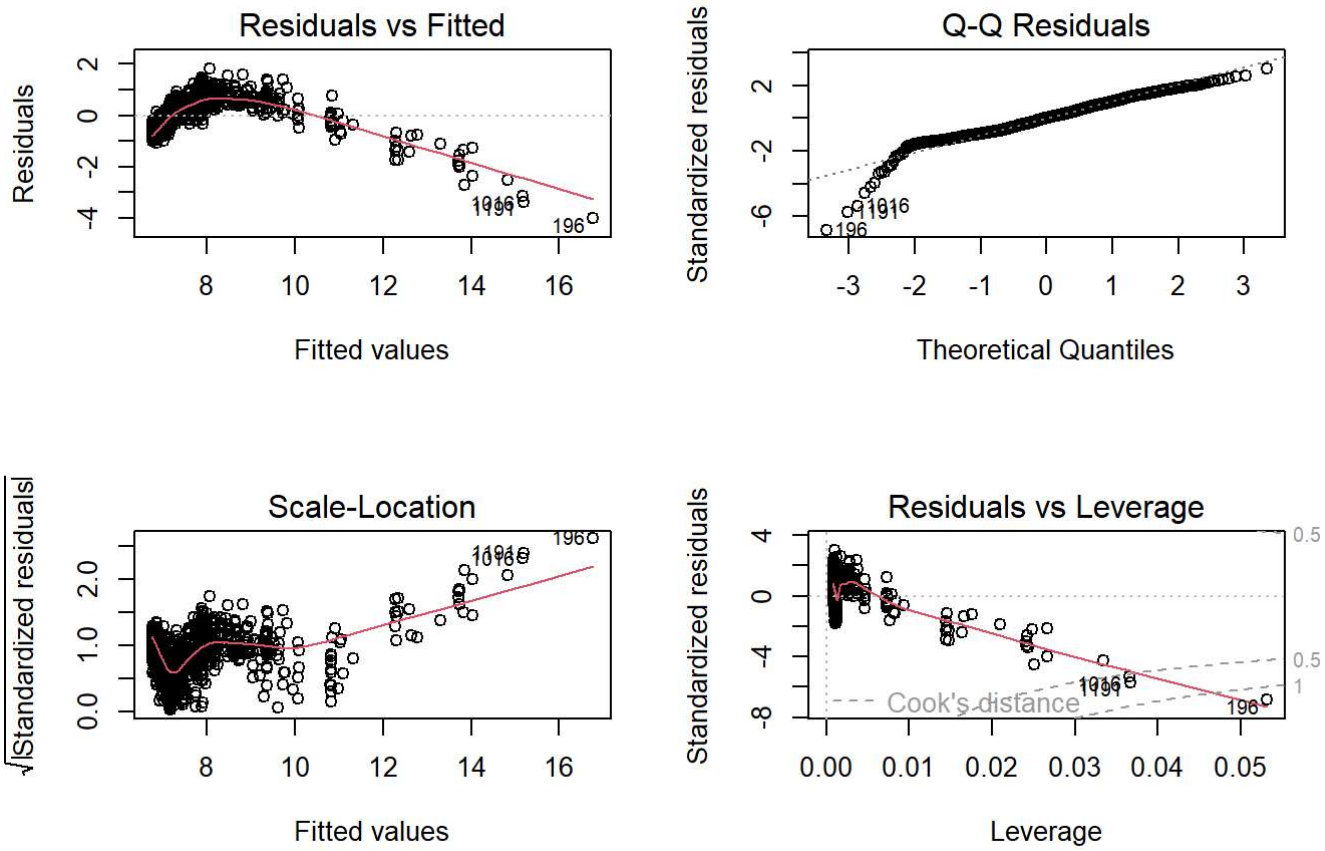
Step #3: Apply Log to Response Variable and view results

```
ystar<-log(data$price)
data<-data.frame(data,ystar)
slr.trans1<-lm(ystar~carat,data=data)
ggplot2::ggplot(data, aes(x=carat, y=ystar))+
  geom_point()+
  geom_smooth(method=lm)+
  labs(x="Carat Weight",y="log(Price)",title="Scatterplot of Sales Price against Carat Weight")
```

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




```
par(mfrow = c(2, 2))
plot(slr.trans1)
```

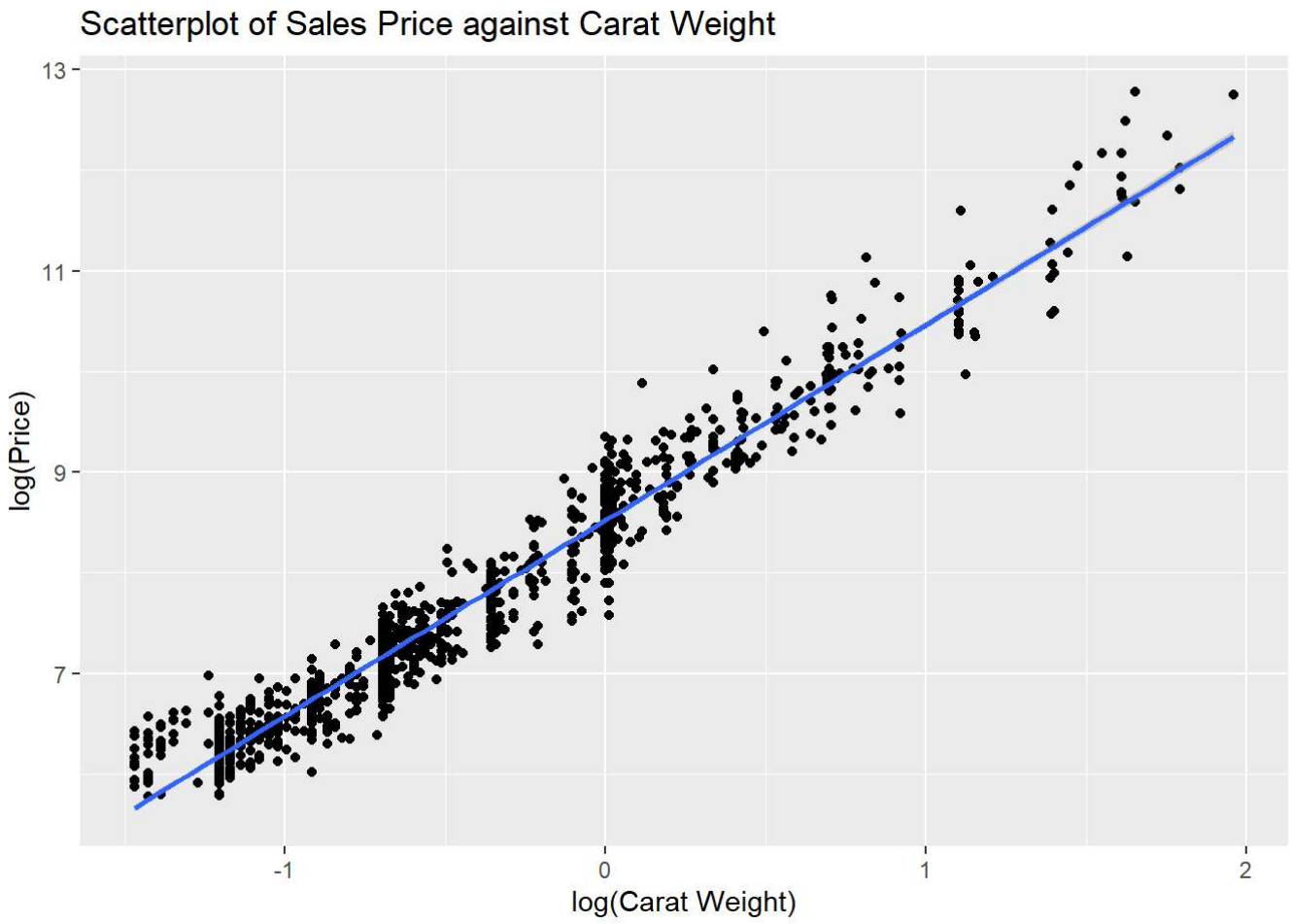


These results appear logarithmic annd indicate that the predictor variable also needs to be transformed using a log transformation

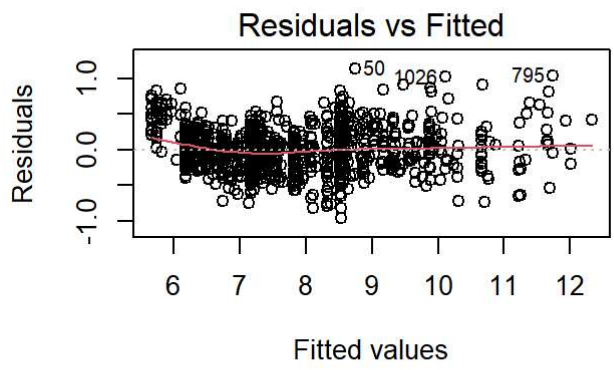
Step #4: Apply Log to the Predictor and view results

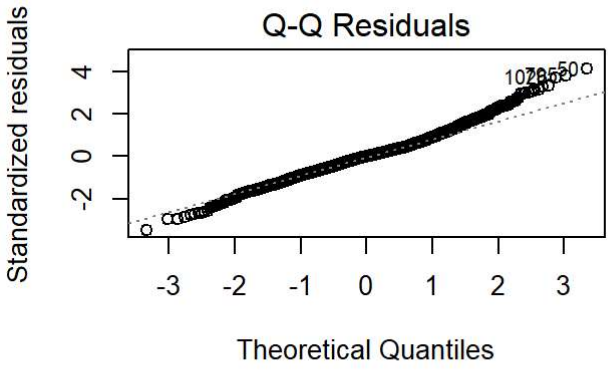
```
xstar<-log(data$carat)
data<-data.frame(data,xstar)
slr.trans2<-lm(ystar~xstar,data=data)
ggplot2::ggplot(data, aes(x=xstar, y=ystar))+
  geom_point()+
  geom_smooth(method=lm)+
  labs(x="log(Carat Weight)",y="log(Price)",title="Scatterplot of Sales Price against Carat Weig
ht")
```

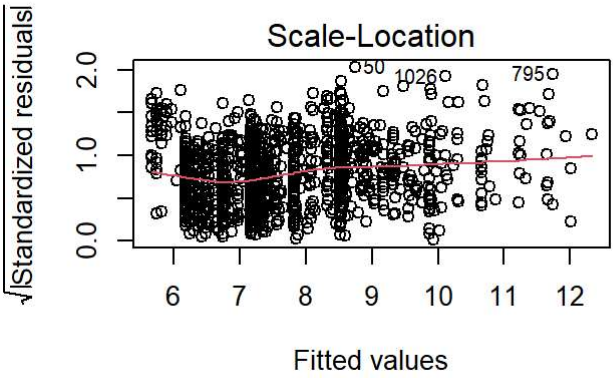
```
## `geom_smooth()` using formula = 'y ~ x'
```

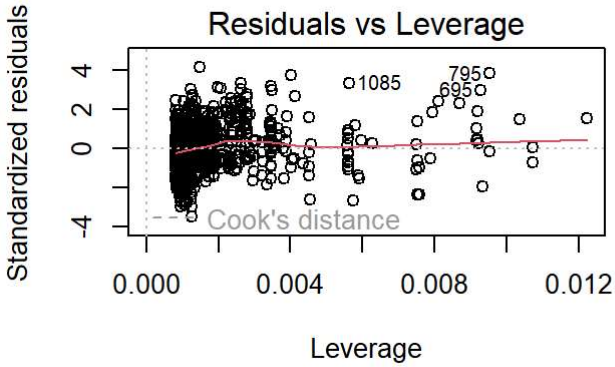



```
par(mfrow = c(2, 2))
plot(slr.trans2)
```










```
summary(slr.trans2)
```

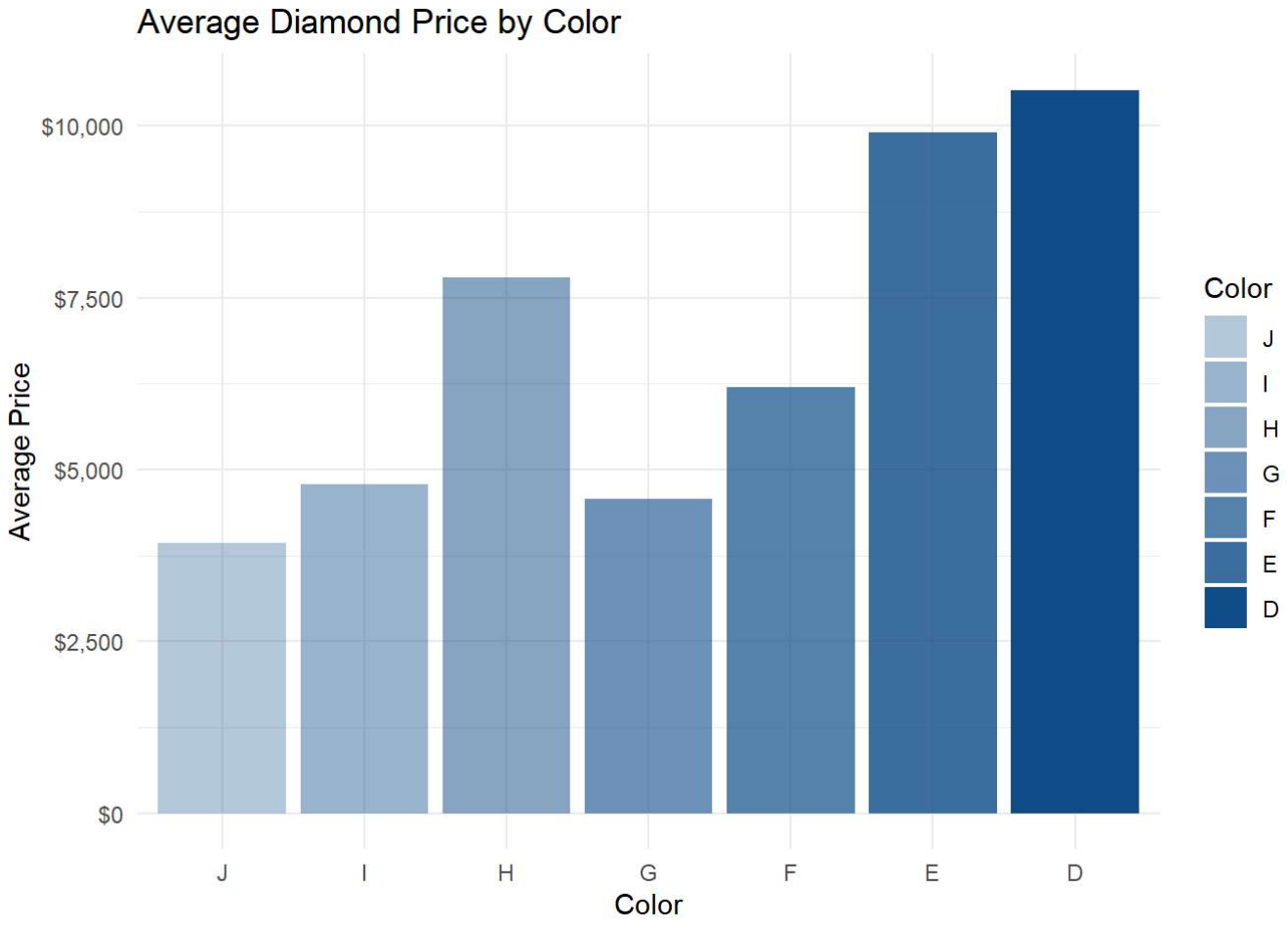
```
##
## Call:
## lm(formula = ystar ~ xstar, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.96394 -0.17231 -0.00252  0.14742  1.14095
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept) 8.521208   0.009734   875.4 <0.0000000000000002 ***
## xstar       1.944020   0.012166   159.8 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2761 on 1212 degrees of freedom
## Multiple R-squared:  0.9547, Adjusted R-squared:  0.9546
## F-statistic: 2.553e+04 on 1 and 1212 DF,  p-value: < 0.00000000000000022
```

This has resulted in a useful data set where all assumptions of linear regressions are met

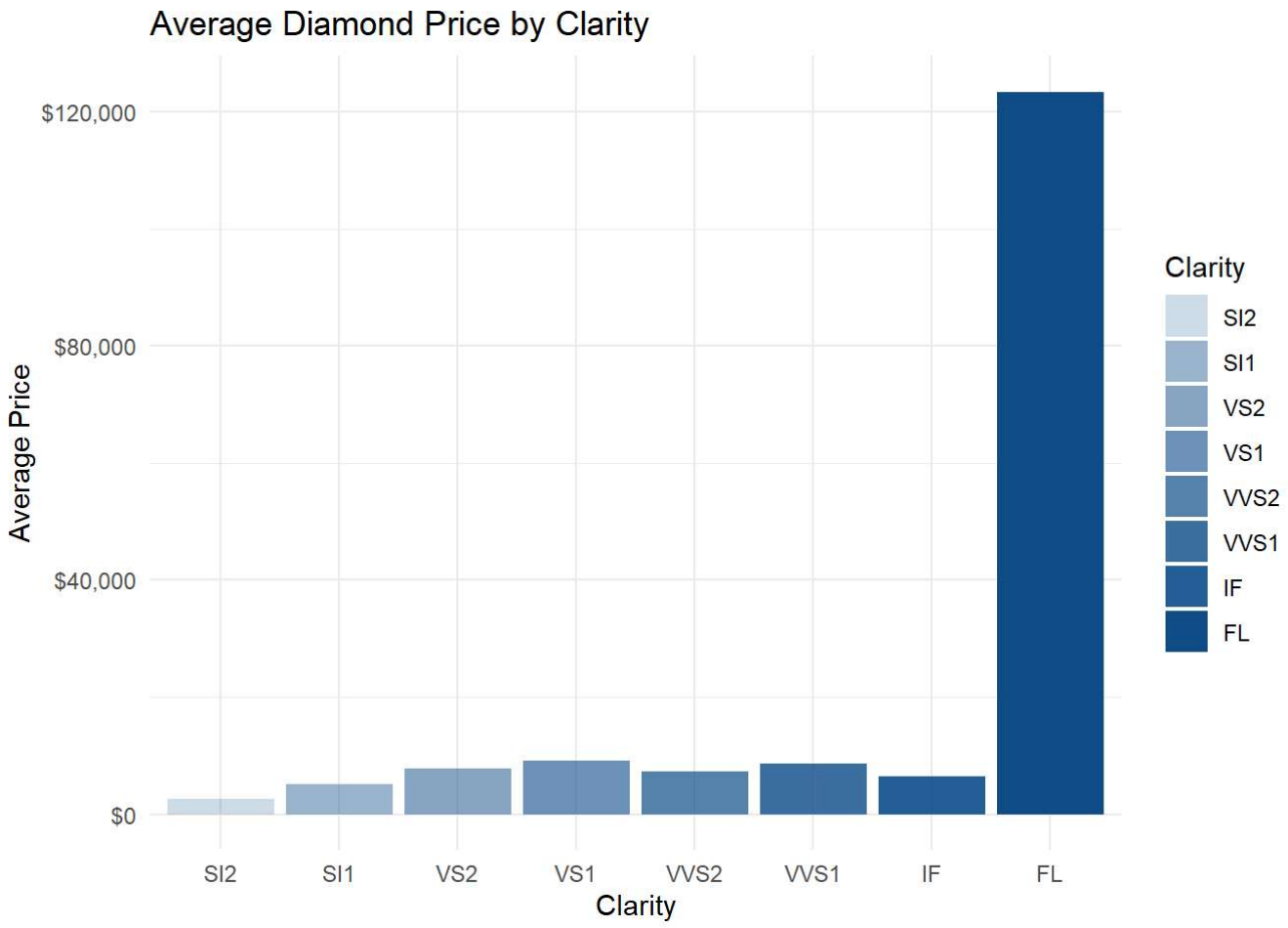
$y^* = 8.521 + 1.944x^*$

Effect of price on carat, clarity, color, and cut

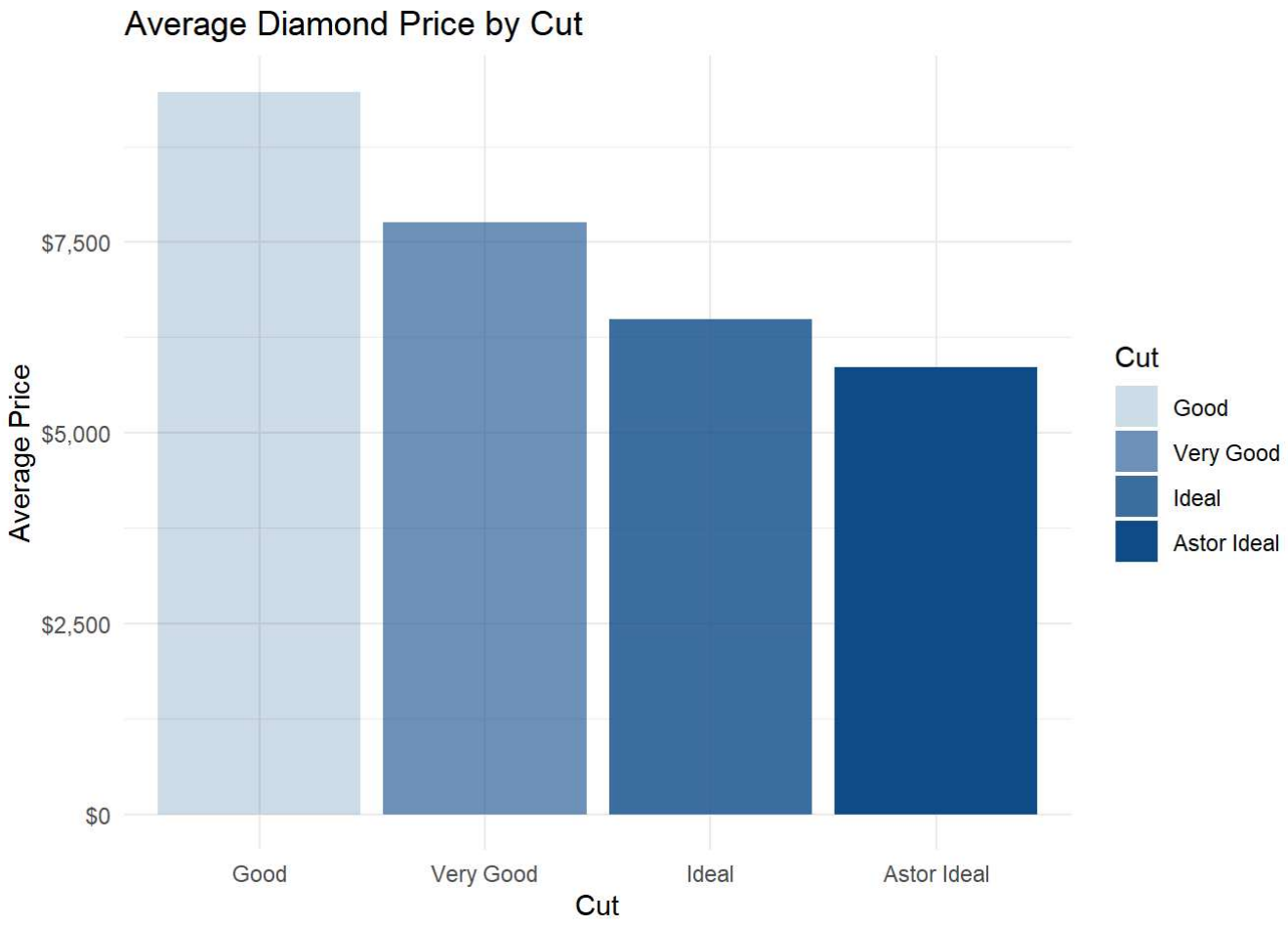
```
alpha_values3 <- c("J" = 0.3, "I" = 0.4, "H" = 0.5, "G" = 0.6, "F" = 0.7, "E" = 0.8, "D" = 1.0)
data %>%
  group_by(color) %>%
  summarise(avg_price = mean(price, na.rm = TRUE)) %>%
  ggplot(aes(x = factor(color, levels = names(alpha_values3)),
             y = avg_price,
             alpha = factor(color, levels = names(alpha_values3)))) +
  geom_col(fill = "dodgerblue4") +
  scale_y_continuous(labels = dollar_format()) +
  scale_alpha_manual(values = alpha_values3) +
  labs(title = "Average Diamond Price by Color",
       x = "Color",
       y = "Average Price",
       alpha = "Color") +
  theme_minimal()
```



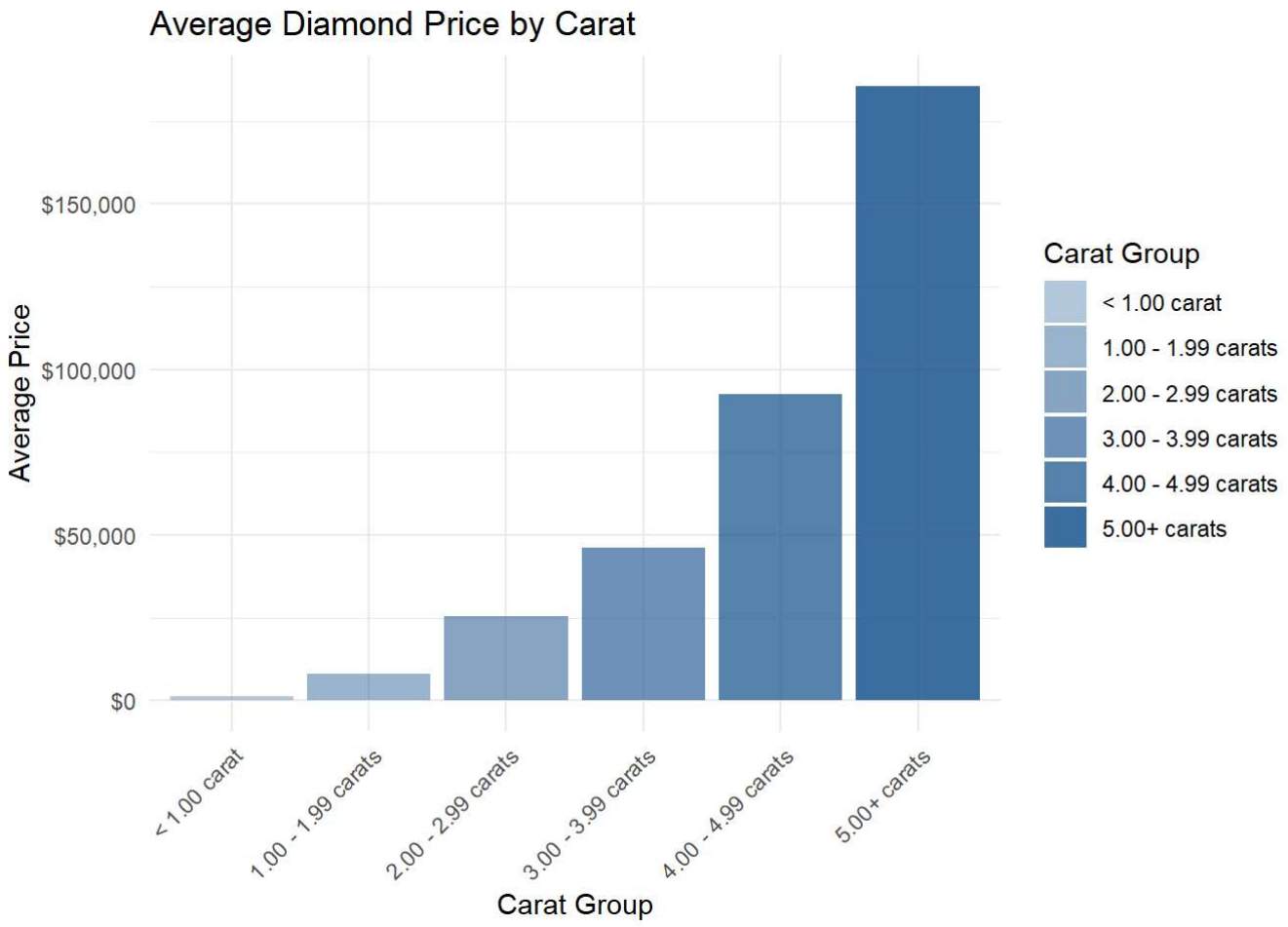
```
alpha_values <- c("SI2" = .2, "SI1" = 0.4, "VS2" = 0.5, "VS1" = 0.6, "VVS2" = 0.7, "VVS1" = 0.8,
"IF" = 0.9, "FL" = 1)
data %>%
  group_by(clarity) %>%
  summarise(avg_price = mean(price, na.rm = TRUE)) %>%
  ggplot(aes(x = factor(clarity, levels = names(alpha_values)),
             y = avg_price, alpha = factor(clarity, levels = names(alpha_values)))) +
  geom_col(fill = "dodgerblue4") +
  scale_alpha_manual(values = alpha_values) +
  scale_y_continuous(labels = dollar_format()) +
  labs(title = "Average Diamond Price by Clarity",
       x = "Clarity",
       y = "Average Price",
       alpha = "Clarity") +  # Proper legend title
  theme_minimal()
```



```
alpha_values2 <- c("Good" = 0.2, "Very Good" = 0.6, "Ideal" = 0.8, "Astor Ideal" = 1.0)
data %>%
  group_by(cut) %>%
  summarise(avg_price = mean(price, na.rm = TRUE)) %>%
  ggplot(aes(x = factor(cut, levels = names(alpha_values2)),
             y = avg_price,
             alpha = factor(cut, levels = names(alpha_values2)))) +
  geom_col(fill = "dodgerblue4") +
  scale_alpha_manual(values = alpha_values2) +
  scale_y_continuous(labels = dollar_format()) +
  labs(title = "Average Diamond Price by Cut",
       x = "Cut",
       y = "Average Price",
       alpha = "Cut") +  # Proper legend title
  theme_minimal()
```



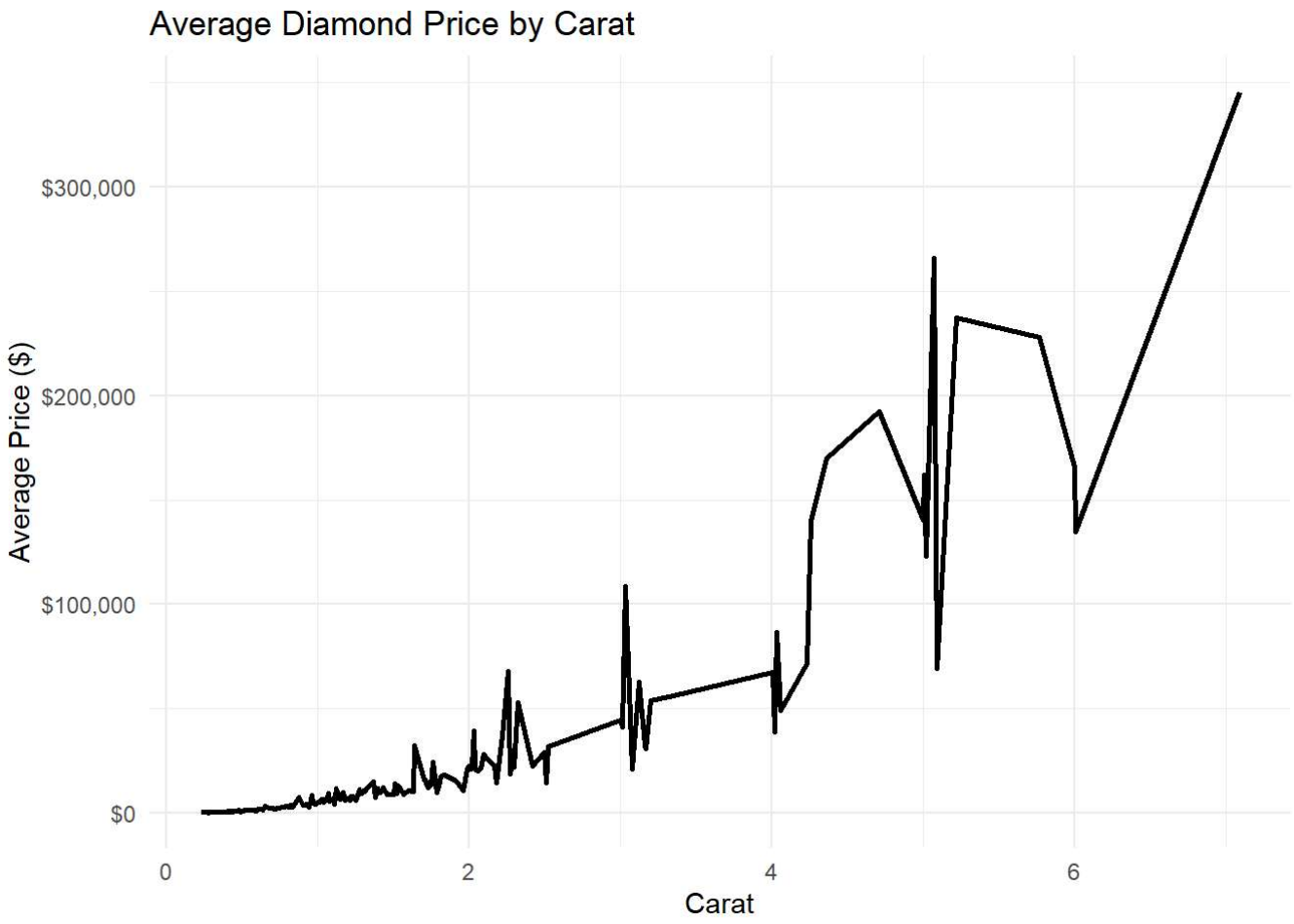
```r
data <- data %>%
  mutate(carat_group = case_when(
    carat < 1 ~ "< 1.00 carat",
    carat >= 1 & carat < 2 ~ "1.00 - 1.99 carats",
    carat >= 2 & carat < 3 ~ "2.00 - 2.99 carats",
    carat >= 3 & carat < 4 ~ "3.00 - 3.99 carats",
    carat >= 4 & carat < 5 ~ "4.00 - 4.99 carats",
    carat >= 5 ~ "5.00+ carats"
  ))

alpha_values4 <- c("< 1.00 carat" = 0.3,
                   "1.00 - 1.99 carats" = 0.4,
                   "2.00 - 2.99 carats" = 0.5,
                   "3.00 - 3.99 carats" = 0.6,
                   "4.00 - 4.99 carats" = 0.7,
                   "5.00+ carats" = 0.8)

data %>%
  group_by(carat_group) %>%
  summarise(avg_price = mean(price, na.rm = TRUE)) %>%
  ggplot(aes(x = carat_group, y = avg_price, alpha = carat_group)) +
  geom_col(fill = "dodgerblue4") +  # Keep all bars the same color
  scale_y_continuous(labels = dollar_format()) +
  scale_alpha_manual(values = alpha_values4) +
  labs(title = "Average Diamond Price by Carat",
       x = "Carat Group",
       y = "Average Price",
       alpha = "Carat Group") +  # Proper legend title
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



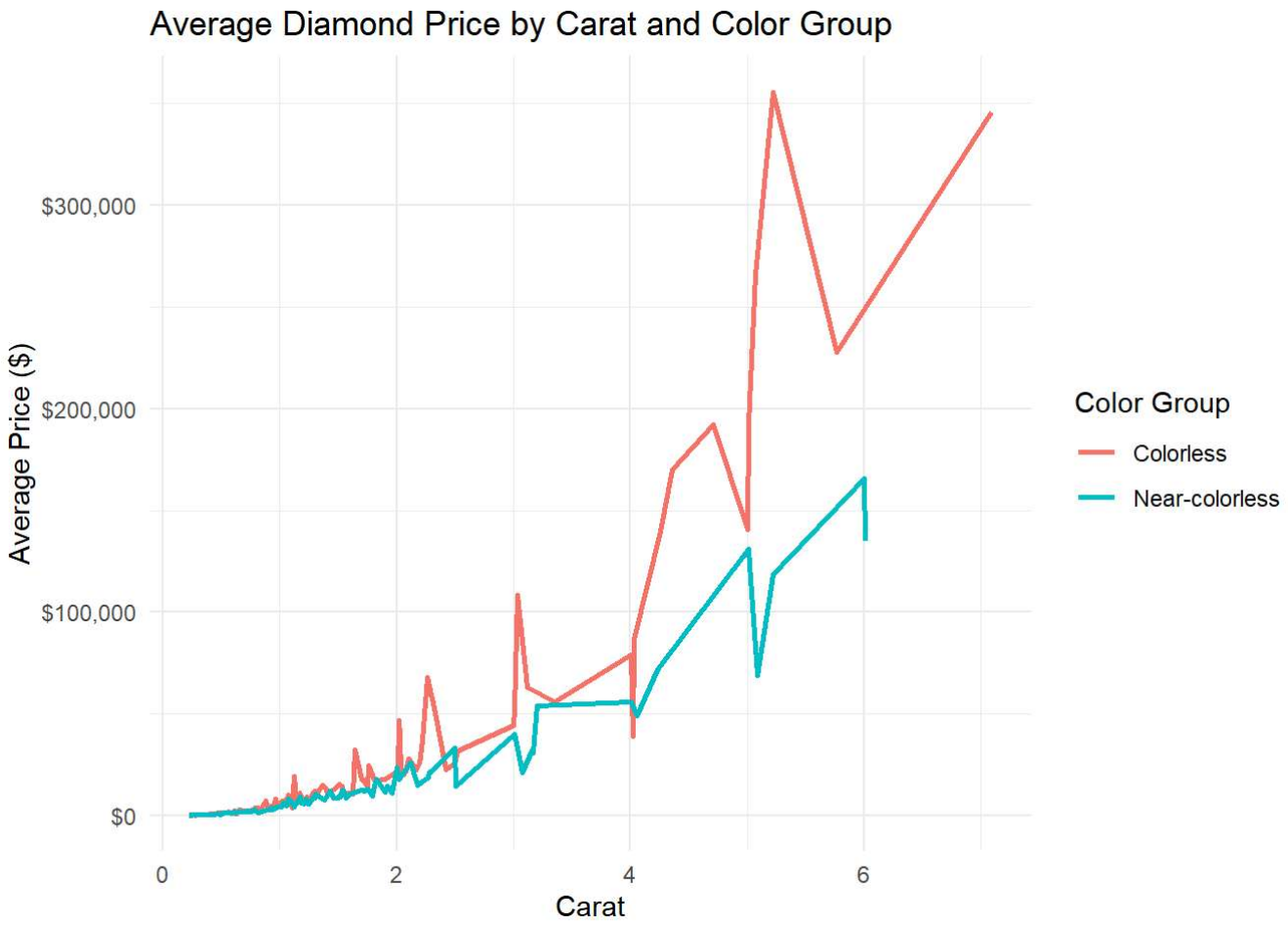
##Buy shy to save money - Diamond Price by Carat/Diamond Price by Carat and Color Group

```
ggplot(data, aes(x = carat, y = price)) +
  geom_line(stat = "summary", fun = mean, size = 1) +
  labs(title = "Average Diamond Price by Carat",
       x = "Carat",
       y = "Average Price ($)") +
  scale_y_continuous(labels = dollar_format()) +
  theme_minimal()
```

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



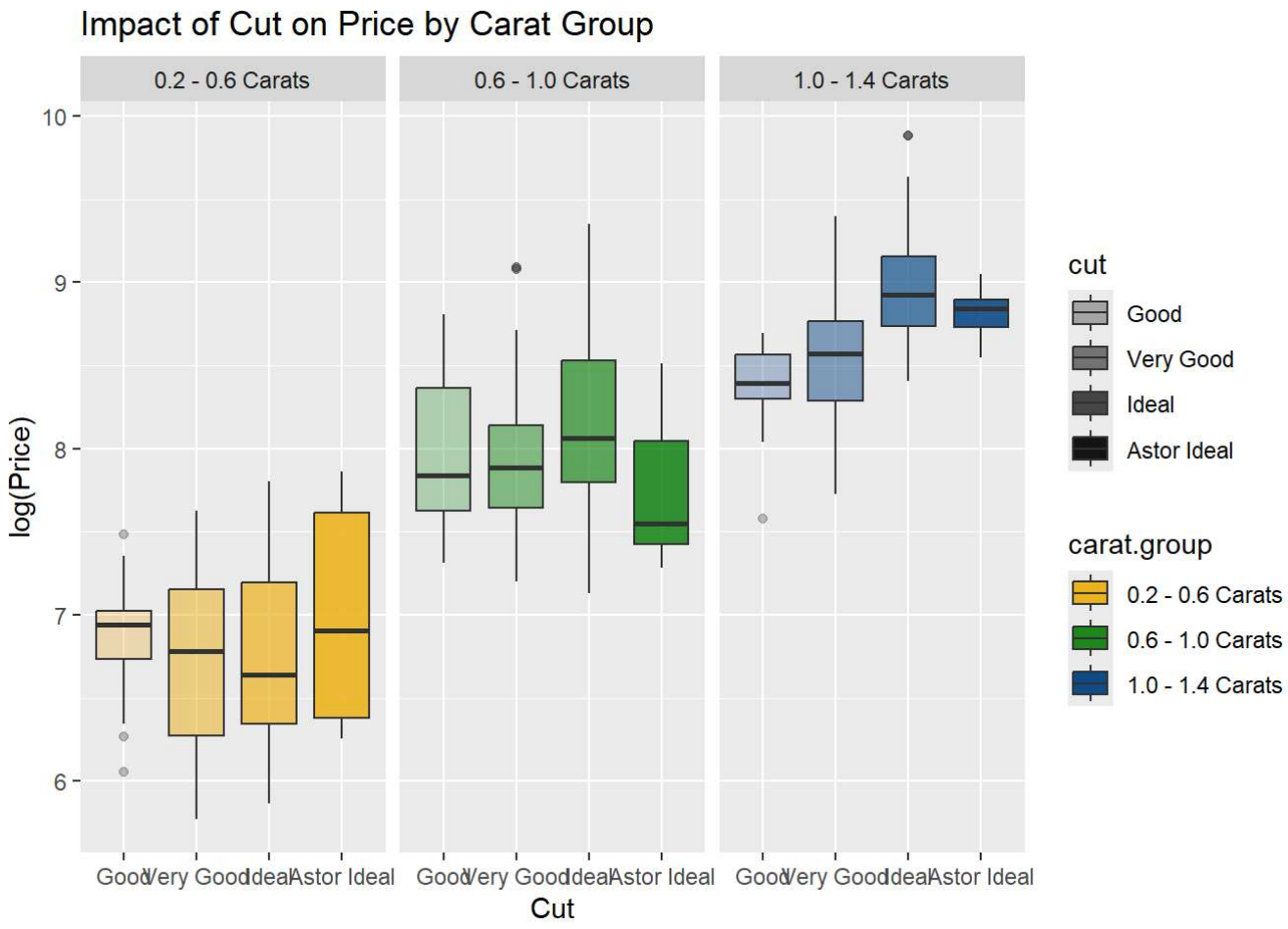

```
data <- data %>%
  mutate(color_group = case_when(
    color %in% c('D', 'E', 'F') ~ "Colorless",
    color %in% c('G', 'H', 'I', 'J') ~ "Near-colorless",
    color %in% c('K') ~ "Faint color",
    TRUE ~ as.character(color)
  ))

ggplot(data, aes(x = carat, y = price, color = color_group)) +
  geom_line(stat = "summary", fun = mean, size = 1) +
  labs(title = "Average Diamond Price by Carat and Color Group",
       x = "Carat",
       y = "Average Price ($)",
       color = "Color Group") +
  scale_y_continuous(labels = dollar_format()) +
  theme_minimal()
```



Analyzing the Impact of Cut across Carat Group

```
data.sub<-data %>%
  filter(carat<1.4)

data.sub<-data.sub %>%
  mutate(carat.group=cut(x=data.sub$carat,breaks=c(0.2,.6,1,1.4),labels=c("0.2 - 0.6 Carats","0.
6 - 1.0 Carats","1.0 - 1.4 Carats")))
data.sub$cut<-factor(data.sub$cut,levels=c("Good","Very Good","Ideal","Astor Ideal"))

ggplot(data.sub,aes(x=cut,y=ystar))+
  geom_boxplot(aes(fill=carat.group,alpha=cut))+
  facet_wrap(~carat.group)+
  labs(title="Impact of Cut on Price by Carat Group",y='log(Price)',x="Cut")+
   scale_alpha_discrete(
     range = c(0.3, 0.9),
     guide = guide_legend(override.aes = list(fill = "black")))+
  scale_fill_manual(values = c("goldenrod2", "forestgreen", "dodgerblue4"))
```

```
## Warning: Using alpha for a discrete variable is not advised.
```



The summary for Color

```
model <- lm(formula = price ~ color, data = data)
summary(model)
```

```
##
## Call:
## lm(formula = price ~ color, data = data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -10203  -6909  -4070  -2083 344878
##
## Coefficients:
##             Estimate Std. Error t value       Pr(>|t|)
## (Intercept)  10525.2     1672.2   6.294 0.000000000432 ***
## colorE        -619.3     2448.3  -0.253         0.8004
## colorF       -4320.6     2322.1  -1.861         0.0630 .
## colorG       -5953.5     2391.6  -2.489         0.0129 *
## colorH       -2726.6     2589.9  -1.053         0.2926
## colorI       -5745.8     2502.5  -2.296         0.0218 *
## colorJ       -6591.1     3037.8  -2.170         0.0302 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 24060 on 1207 degrees of freedom
## Multiple R-squared:  0.01016,    Adjusted R-squared:  0.005237
## F-statistic: 2.064 on 6 and 1207 DF,  p-value: 0.05474
```

```
model2 <- lm(formula = price ~ cut, data = diamonds)
summary(model2)
```

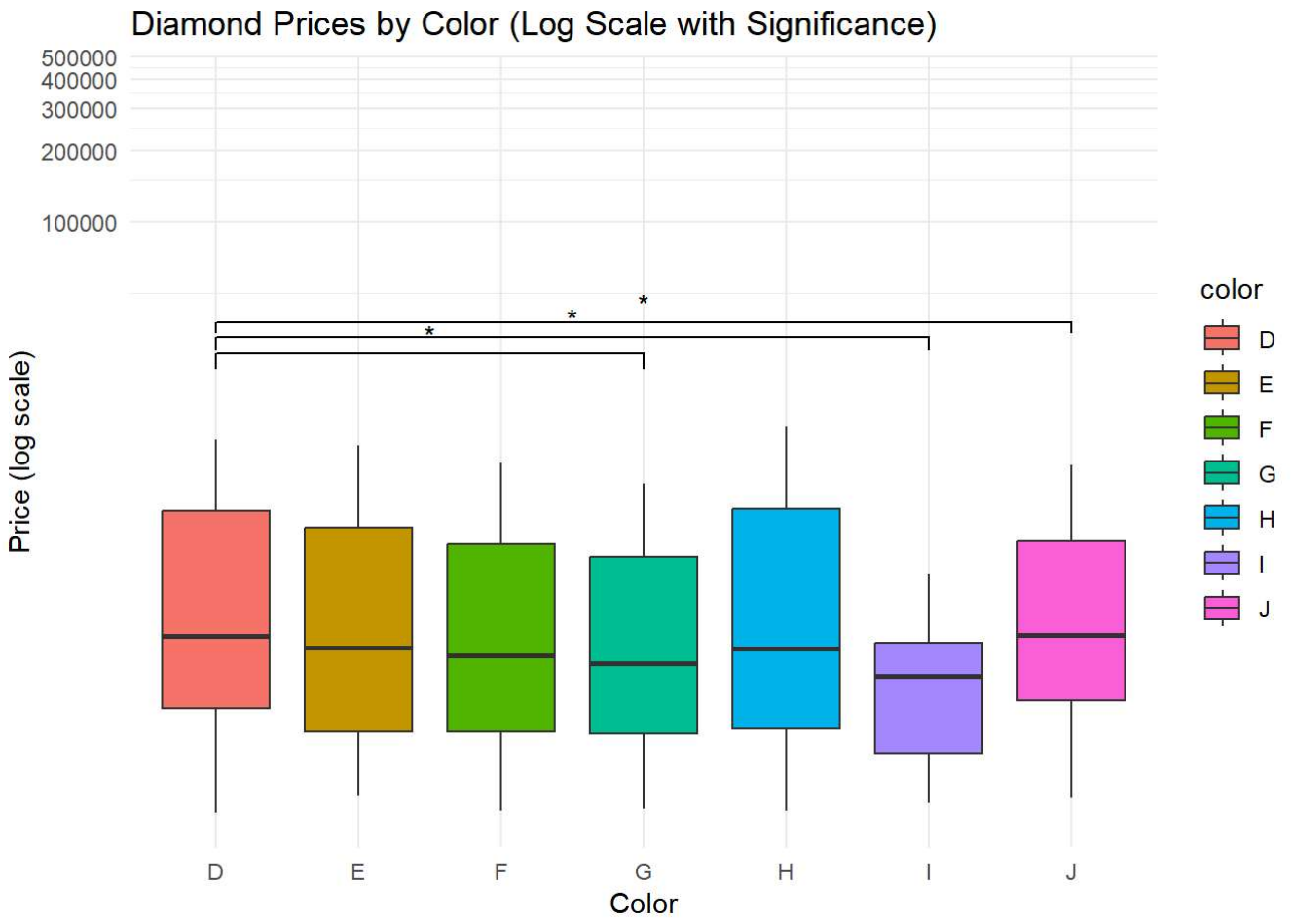
```
##
## Call:
## lm(formula = price ~ cut, data = diamonds)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -4258  -2741  -1494   1360  15348
##
## Coefficients:
##             Estimate Std. Error t value             Pr(>|t|)
## (Intercept)  4062.24      25.40 159.923 < 0.0000000000000002 ***
## cut.L        -362.73      68.04  -5.331       0.000000098033 ***
## cut.Q        -225.58      60.65  -3.719               0.0002 ***
## cut.C        -699.50      52.78 -13.253 < 0.0000000000000002 ***
## cut^4        -280.36      42.56  -6.588       0.000000000045 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3964 on 53935 degrees of freedom
## Multiple R-squared:  0.01286,    Adjusted R-squared:  0.01279
## F-statistic: 175.7 on 4 and 53935 DF,  p-value: < 0.00000000000000022
```

Table format:

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	10525.2	1672.2	6.294	4.32e-10	***
colorE	-619.3	2448.3	-0.253	0.8004	
colorF	-4320.6	2322.1	-1.861	0.0630	.
colorG	-5953.5	2391.6	-2.489	0.0129	*
colorH	-2726.6	2589.9	-1.053	0.2926	
colorI	-5745.8	2502.5	-2.296	0.0218	*
colorJ	-6591.1	3037.8	-2.170	0.0302	*

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	5851.6	5397.3	1.084	0.279
cutGood	3615.7	6091.9	0.594	0.553
cutIdeal	637.4	5469.8	0.117	0.907
cutVery Good	1906.1	5536.8	0.344	0.731

```
ggplot(data, aes(x = color, y = price, fill = color)) +
  geom_boxplot(outlier.shape = NA) +
  coord_trans(y = "log10") +
  geom_signif(comparisons = list(c("D", "G")), y_position = 10000, annotation = "*", tip_length
= 0.01) +
  geom_signif(comparisons = list(c("D", "I")), y_position = 15000, annotation = "*", tip_length
= 0.01) +
  geom_signif(comparisons = list(c("D", "J")), y_position = 20000, annotation = "*", tip_length
= 0.01) +
  labs(title = "Diamond Prices by Color (Log Scale with Significance)",
       y = "Price (log scale)",
       x = "Color") +
  theme_minimal()
```

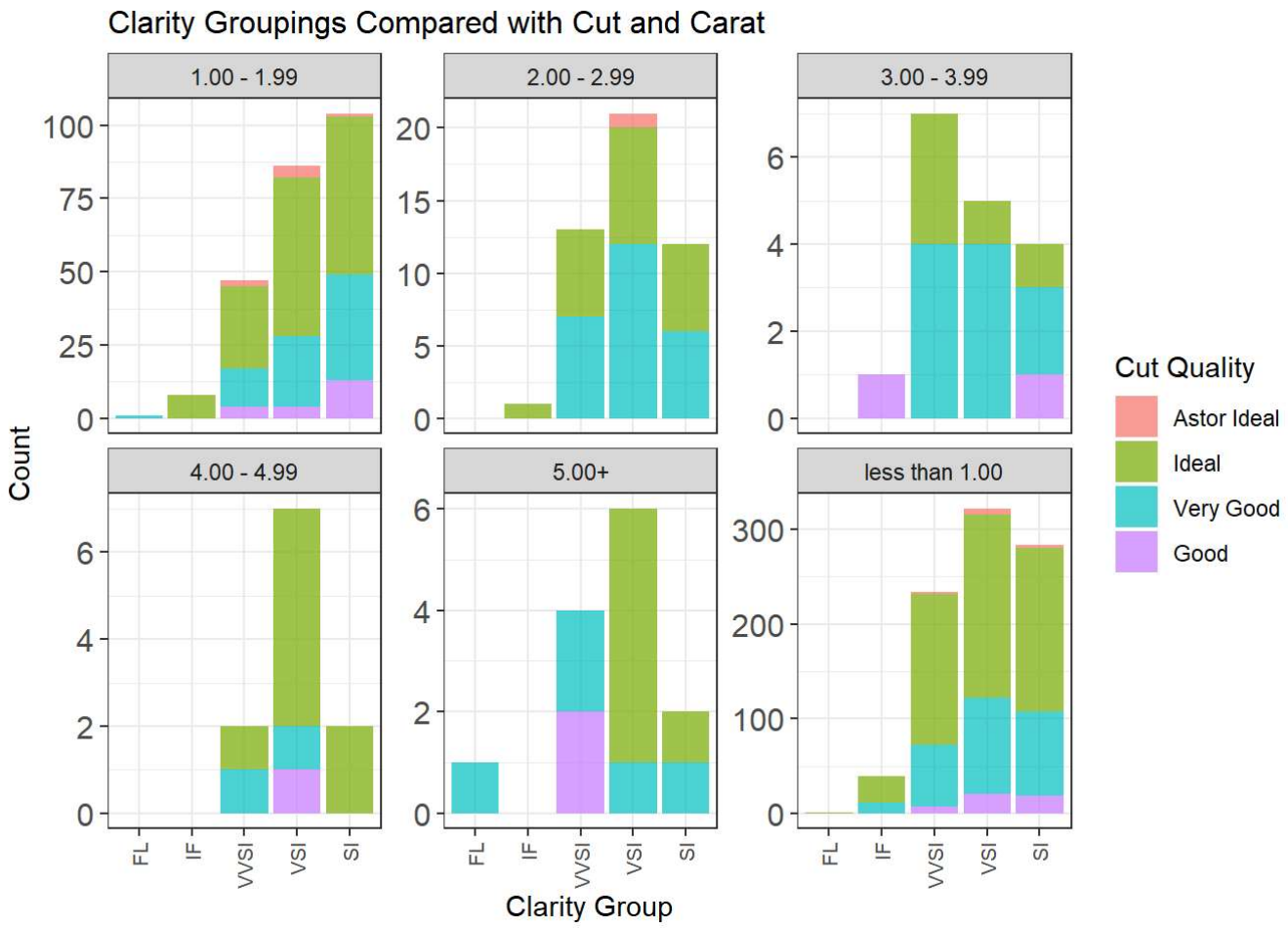



```
data<-data %>% # used to make binning of data more uniform
mutate(clarity_group = case_when(
    clarity %in% c('SI1', 'SI2')~"SI",
    clarity %in% c('VS1', 'VS2')~"VSI",
    clarity %in% c('VVS1', 'VVS2')~"VVSI",
    clarity %in% c('IF')~"IF",
    clarity %in% c('FL')~"FL",
    TRUE ~ as.character(clarity)
  ))
data<- data %>% # pulled from code format from Greg's data manipulation
  mutate(carat_group = case_when(
    carat < 1 ~ "less than 1.00",
    carat >= 1 & carat < 2 ~ "1.00 - 1.99",
    carat >= 2 & carat < 3 ~ "2.00 - 2.99",
    carat >= 3 & carat < 4 ~ "3.00 - 3.99",
    carat >= 4 & carat < 5 ~ "4.00 - 4.99",
    carat >= 5 ~ "5.00+"
  ))
```

Clarity compared with Cut and Carat

```
data<-data %>% # used to make binning of data more uniform
  mutate(clarity_group = case_when(
    clarity %in% c('SI1', 'SI2')~"SI",
    clarity %in% c('VS1', 'VS2')~"VSI",
    clarity %in% c('VVS1', 'VVS2')~"VVSI",
    clarity %in% c('IF')~"IF",
    clarity %in% c('FL')~"FL",
    TRUE ~ as.character(clarity)
  ))
```

```
ggplot(data, aes(x = factor(clarity_group, levels=c("FL","IF", "VVSI", "VSI", "SI")), fill = fac
tor(cut, levels = c("Astor Ideal", "Ideal", "Very Good", "Good")))) +
  geom_bar(alpha = 0.7, position = "stack") +
  facet_wrap(~ carat_group, scales = "free_y")+
  labs(title = "Clarity Groupings Compared with Cut and Carat",
       x = "Clarity Group",
       y = "Count",
       fill = "Cut Quality") +
  theme_bw() +  # theme for better readability
  theme(
      plot.title   = element_text(hjust= 0, size = 12),
      axis.text.x  = element_text(angle = 90, vjust = 0.5, hjust = 1, size = 8),
      axis.text.y  = element_text(size = 12)
    )
```




```
# A table summarizing counts for clarity_group, carat_group, and cut
summary_table <- as.data.frame(
  table(data$clarity_group, data$carat_group, data$cut)
)

# Renamimg the columns of the table
colnames(summary_table) <- c("Clarity Group", "Carat Group", "Cut", "Count")
```

Tables to correspond to the facet wrapped

```
# Filtered and grouped by carat_group, then clarity_group
filtered_data <- data %>%
  group_by(carat_group, clarity_group, cut) %>%
  summarise(Count = n(), .groups = "drop") %>%
  filter(Count != 0)

# Data by carat_group
facet_tables <- split(filtered_data, filtered_data$carat_group)

#summary tables for each carat_group
summary_tables <- lapply(facet_tables, function(group_data) {
  group_data <- group_data %>%
    group_by(clarity_group, cut) %>%
    summarise(Count = sum(Count), .groups = "drop") %>%
    mutate(Percentage = round((Count / sum(Count) * 100), 2)) %>%
    rename(
      Clarity = clarity_group,
      Cut = cut
    )
  return(group_data)
})
# exportable tables

for (carat in names(summary_tables)) {
  output_file <- paste0("table_", carat, ".html")
  cat("\n### Carat Group:", carat, "\n")

  summary_tables[[carat]] %>%
    kable(
      caption = paste("Table for Carat Group:", carat),
      format = "html"
    ) %>%
    kable_styling(
      bootstrap_options = c("striped", "hover", "condensed"),
      full_width = FALSE,
      position = "center"
    ) %>%
    add_header_above(c(" " = 2, "Summary" = 2))%>%
    save_kable(file = output_file)
}
```

```
##
## ### Carat Group: 1.00 - 1.99
##
## ### Carat Group: 2.00 - 2.99
##
## ### Carat Group: 3.00 - 3.99
##
## ### Carat Group: 4.00 - 4.99
##
## ### Carat Group: 5.00+
##
## ### Carat Group: less than 1.00
```