```r
# Demonstrates loading data from an .RData file

load("temps.RData")
mean(temps)
```

```r
# Demonstrates identifying outliers by index

load("temps.RData")

temps[2]
temps[4]
temps[7]

temps[c(2, 4, 7)]
```

```r
# Demonstrates removing outliers by index

load("temps.RData")
no_outliers <- temps[-c(2, 4, 7)]

mean(no_outliers)
mean(temps)
```

```r
# Demonstrates identifying outliers with logical expressions

load("temps.RData")

temps[1] < 0
temps[2] < 0
temps[3] < 0
```

```r
# Demonstrates comparison operators are vectorized

load("temps.RData")

temps < 0
```

```r
# Demonstrates `which` to return indices for which a logical expression is TRUE

load("temps.RData")

which(temps < 0)
```

```r
# Demonstrates identifying outliers with compound logical expressions

load("temps.RData")
temps < 0 | temps > 60
```

```r
# Demonstrates `any` and `all` to test for outliers

load("temps.RData")

any(temps < 0 | temps > 60)
all(temps < 0 | temps > 60)
```

```r
# Demonstrates subsetting a vector with a logical vector

load("temps.RData")
filter <- temps < 0 | temps > 60
temps[filter]
```

```r
# Demonstrates negating a logical expression with !

load("temps.RData")
filter <- !(temps < 0 | temps > 60)
temps[filter]
```

```r
# Demonstrates removing outliers
load("temps.RData")

no_outliers <- temps[!(temps < 0 | temps > 60)]
save(no_outliers, file = "no_outliers.RData")

outliers <- temps[temps < 0 | temps > 60]
save(outliers, file = "outliers.RData")
```

```r
# Reads a CSV of data

chicks <- read.csv("chicks.csv")
View(chicks)
```

```r
# Demonstrates `mean` calculation with NA values

chicks <- read.csv("chicks.csv")
average_weight <- mean(chicks$weight)
average_weight
```

```r
# Demonstrates na.rm to remove NA values from mean calculation

chicks <- read.csv("chicks.csv")
average_weight <- mean(chicks$weight, na.rm = TRUE)
average_weight
```

```r
# Demonstrates computing casein average with explicit indexes

chicks <- read.csv("chicks.csv")
casein_chicks <- chicks[c(1, 2, 3), ]
mean(casein_chicks$weight)
```

```r
# Demonstrates constructing sequential vector with :

chicks <- read.csv("chicks.csv")
casein_chicks <- chicks[1:3, ]
mean(casein_chicks$weight)
```

```r
# Demonstrates logical expression to identify rows with casein feed

chicks <- read.csv("chicks.csv")

chicks$feed == "casein"
```

```r
# Demonstrates subsetting data frame with logical vector

chicks <- read.csv("chicks.csv")

filter <- chicks$feed == "casein"
casein_chicks <- chicks[filter, ]
mean(casein_chicks$weight)
```

```r
# Demonstrates subsetting with `subset`

chicks <- read.csv("chicks.csv")

casein_chicks <- subset(chicks, feed == "casein")
mean(casein_chicks$weight, na.rm = TRUE)
```

```r
# Demonstrates failing to remove NA values

chicks <- read.csv("chicks.csv")
chicks$weight != NA
```

```r
# Demonstrates identifying NA values with `is.na`

chicks <- read.csv("chicks.csv")

is.na(chicks$weight)
!is.na(chicks$weight)

chicks$chick[is.na(chicks$weight)]
```

```r
# Demonstrates removing NA values and resetting row names

chicks <- read.csv("chicks.csv")

chicks <- subset(chicks, !is.na(weight))
rownames(chicks)

rownames(chicks) <- NULL
rownames(chicks)
```

```r
# Demonstrates interactive program to view data by feed type

# Read and clean data
chicks <- read.csv("chicks.csv")
chicks <- subset(chicks, !is.na(weight))

# Determine feed options
feed_options <- unique(chicks$feed)

# Prompt user with options
cat("1.", feed_options[1])
cat("2.", feed_options[2])
cat("3.", feed_options[3])
cat("4.", feed_options[4])
cat("5.", feed_options[5])
cat("6.", feed_options[6])
feed_choice <- as.integer(readline("Feed type: "))
```

```r
# Demonstrates \n

# Read and clean data
chicks <- read.csv("chicks.csv")
chicks <- subset(chicks, !is.na(weight))

# Determine feed options
feed_options <- unique(chicks$feed)

# Prompt user with options
cat("1.", feed_options[1], "\n")
cat("2.", feed_options[2], "\n")
cat("3.", feed_options[3], "\n")
cat("4.", feed_options[4], "\n")
cat("5.", feed_options[5], "\n")
cat("6.", feed_options[6], "\n")
feed_choice <- as.integer(readline("Feed type: "))
```

```r
# Demonstrates interactive program to view data by feed type

# Read and clean data
chicks <- read.csv("chicks.csv")
chicks <- subset(chicks, !is.na(weight))

# Determine feed options
feed_options <- unique(chicks$feed)

# Format feed options
formatted_options <- paste0(1:length(feed_options), ". ", feed_options)

# Prompt user with options
cat(formatted_options, sep = "\n")
feed_choice <- as.integer(readline("Feed type: "))
```

```r
# Demonstrates interactive program to view data by feed type

# Read and clean data
chicks <- read.csv("chicks.csv")
chicks <- subset(chicks, !is.na(weight))

# Determine feed options
feed_options <- unique(chicks$feed)

# Format feed options
formatted_options <- paste0(1:length(feed_options), ". ", feed_options)

# Prompt user with options
cat(formatted_options, sep = "\n")
feed_choice <- as.integer(readline("Feed type: "))

# Print selected option
selected_feed <- feed_options[feed_choice]
print(subset(chicks, feed == selected_feed))
```

```r
# Demonstrates interactive program to view data by feed type

# Read and clean data
chicks <- read.csv("chicks.csv")
chicks <- subset(chicks, !is.na(weight))

# Determine feed options
feed_options <- unique(chicks$feed)

# Format feed options
formatted_options <- paste0(1:length(feed_options), ". ", feed_options)

# Prompt user with options
cat(formatted_options, sep = "\n")
feed_choice <- as.integer(readline("Feed type: "))

# Invalid choice?
if (feed_choice < 1 || feed_choice > length(feed_options)) {
  cat("Invalid choice.")
}

selected_feed <- feed_options[feed_choice]
print(subset(chicks, feed == selected_feed))
```

```r
# Demonstrates interactive program to view data by feed type

# Read and clean data
chicks <- read.csv("chicks.csv")
chicks <- subset(chicks, !is.na(weight))

# Determine feed options
feed_options <- unique(chicks$feed)

# Format feed options
formatted_options <- paste0(1:length(feed_options), ". ", feed_options)

# Prompt user with options
cat(formatted_options, sep = "\n")
feed_choice <- as.integer(readline("Feed type: "))

# Invalid choice?
if (feed_choice < 1 || feed_choice > length(feed_options)) {
  cat("Invalid choice.")
} else {
  selected_feed <- feed_options[feed_choice]
  print(subset(chicks, feed == selected_feed))
}
```

```r
# Implements same functionality with `menu`

# Read and clean data
chicks <- read.csv("chicks.csv")
chicks <- subset(chicks, !is.na(weight))

# Prompt user for input
feed_options <- unique(chicks$feed)
feed_choice <- menu(
  feed_options,
  title = "Feed type:"
)

# Show subset
selected_feed = feed_options[feed_choice]
print(subset(chicks, feed == selected_feed))
```

```r
# Reads 4 separate CSVs

Q1 <- read.csv("Q1.csv")
Q2 <- read.csv("Q2.csv")
Q3 <- read.csv("Q3.csv")
Q4 <- read.csv("Q4.csv")
```

```r
# Combines data frames with `rbind`

Q1 <- read.csv("Q1.csv")
Q2 <- read.csv("Q2.csv")
Q3 <- read.csv("Q3.csv")
Q4 <- read.csv("Q4.csv")

sales <- rbind(Q1, Q2, Q3, Q4)
```

```r
# Adds quarter column to data frames

Q1 <- read.csv("Q1.csv")
Q1$quarter <- "Q1"

Q2 <- read.csv("Q2.csv")
Q2$quarter <- "Q2"

Q3 <- read.csv("Q3.csv")
Q3$quarter <- "Q3"

Q4 <- read.csv("Q4.csv")
Q4$quarter <- "Q4"

sales <- rbind(Q1, Q2, Q3, Q4)
```

```r
# Demonstrates flagging sales as high value

Q1 <- read.csv("Q1.csv")
Q1$quarter <- "Q1"

Q2 <- read.csv("Q2.csv")
Q2$quarter <- "Q2"

Q3 <- read.csv("Q3.csv")
Q3$quarter <- "Q3"

Q4 <- read.csv("Q4.csv")
Q4$quarter <- "Q4"

sales <- rbind(Q1, Q2, Q3, Q4)

sales$value <- ifelse(sales$sale_amount > 100, "High Value", "Regular")
```