IMAGE 1: Airbnbs density by County

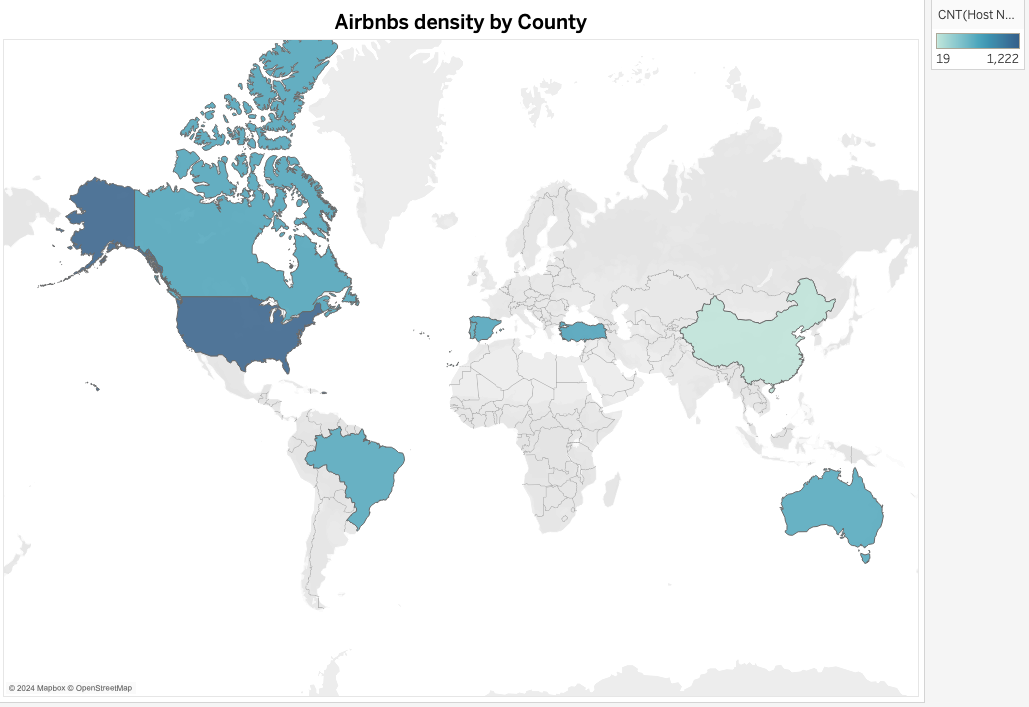


IMAGE 2: Median Price Distribution by Room Type and Location

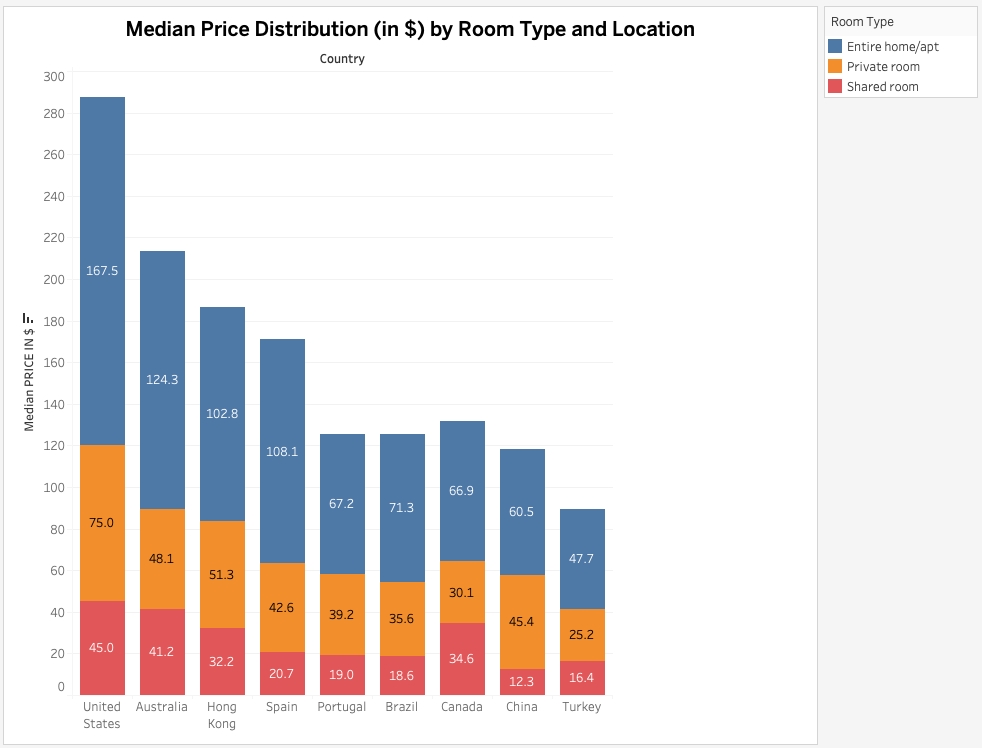


IMAGE : 3 TOP QUALITY HOSTS

| **TOP QUALITY HOSTS** | | |
| --- | --- | --- |
| **Host Name** | **Number Of Reviews** | **Avg. Review Scores Rating** |
| David | 1,244 | 93 |
| Sarah | 905 | 91 |
| Maria | 900 | 95 |
| Paul | 786 | 96 |
| Ana | 758 | 93 |
| Sam | 735 | 91 |
| Luis | 728 | 91 |
| Cristina | 709 | 93 |
| Koni | 667 | 93 |
| João | 662 | 95 |
| Jose | 645 | 91 |
| Dana | 644 | 97 |
| Michael | 616 | 94 |
| Jason | 612 | 95 |
| Susan | 604 | 91 |
| Carlos | 578 | 91 |
| Miguel | 563 | 91 |
| Paula | 557 | 98 |
| Francisco | 555 | 87 |

IMAGE 4 : Host Response Rate Analysis

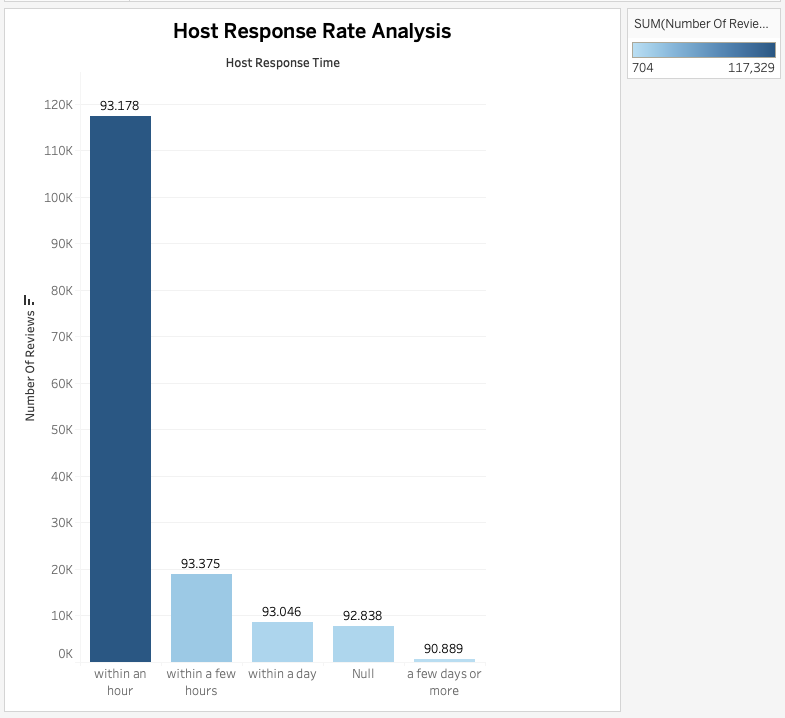


IMAGE 5.1 : FREQUENCY DISTRIBUTION OF ROOM TYPES

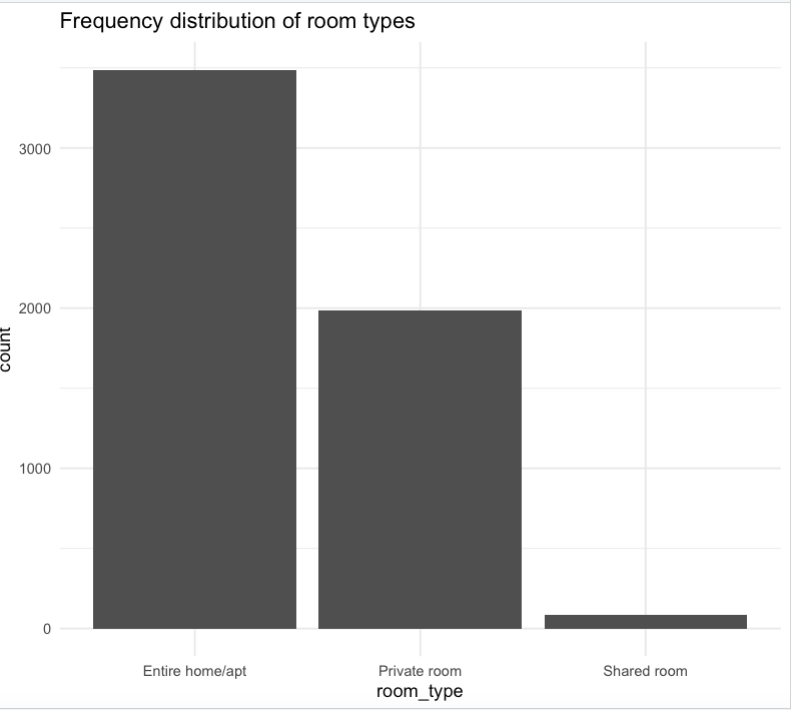


IMAGE 5.2 : FREQUENCY DISTRIBUTION OF BED TYPES

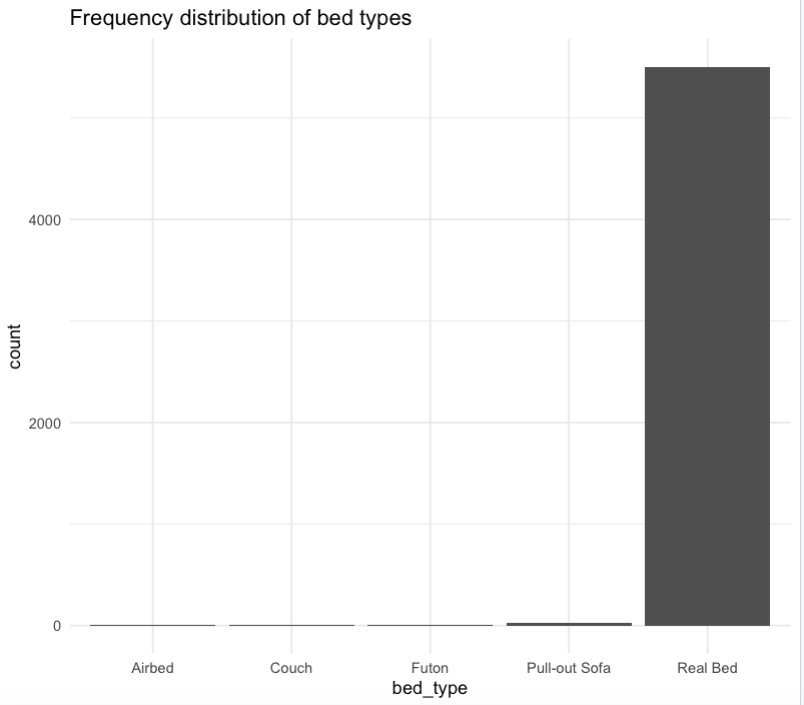


IMAGE 6 : FREQUENCY DISTRIBUTION OF CANCELLATION POLICIES

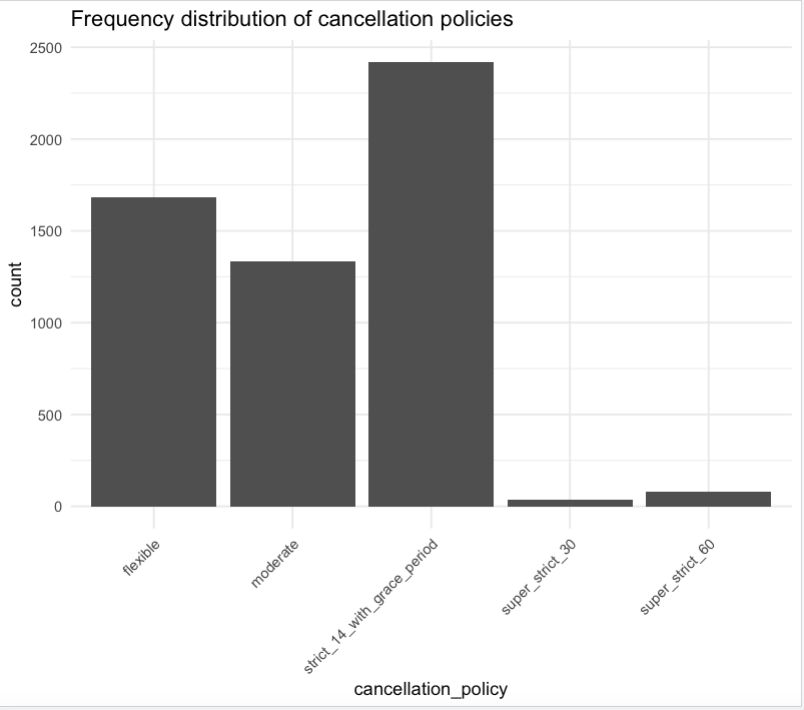


IMAGE 7 : TOP 20 WORDS IN AIRBNB LISTINGS

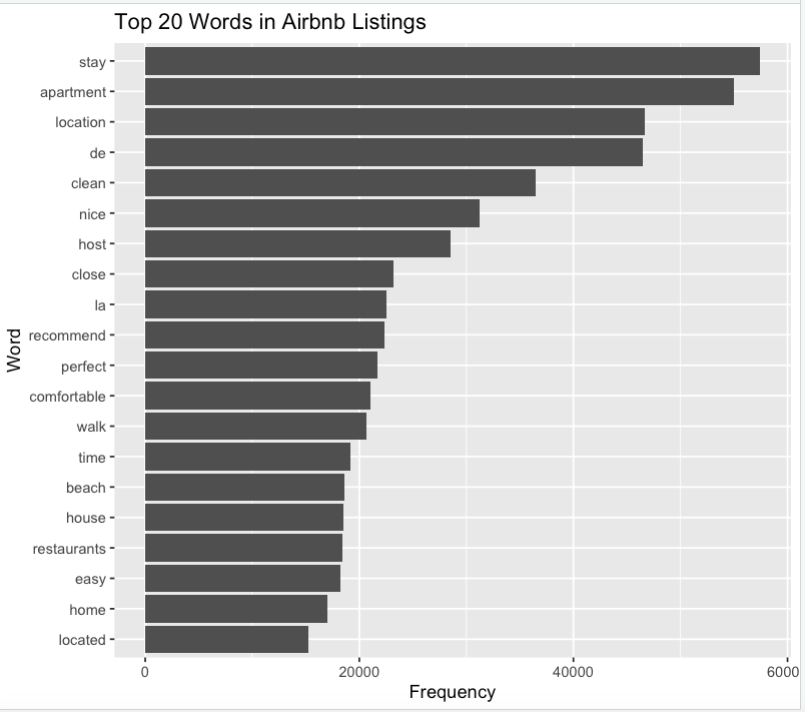


IMAGE 8 : TOP 20 BIGRAMS IN AIRBNB LISTINGS

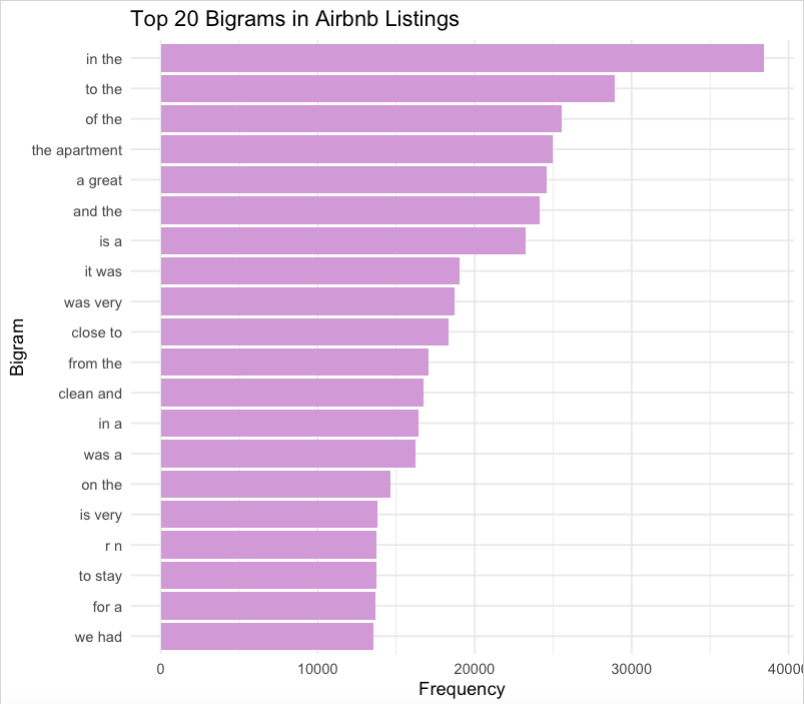


IMAGE 9 : DISTRIBUTION OF EMOTIONS IN AIRBNB LISTINGS

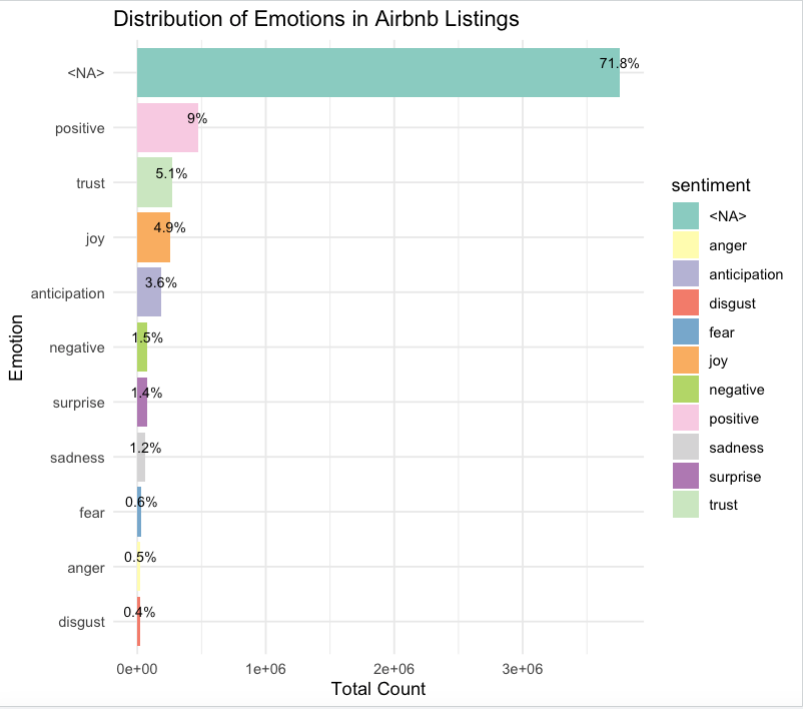


IMAGE 10 : WORD LENGTH VS TF-IDF SCORE

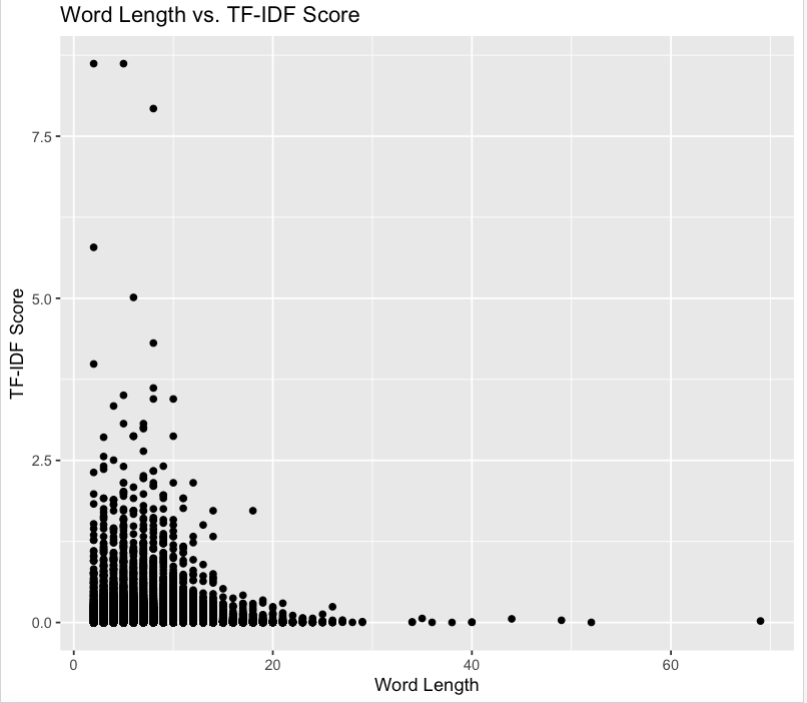
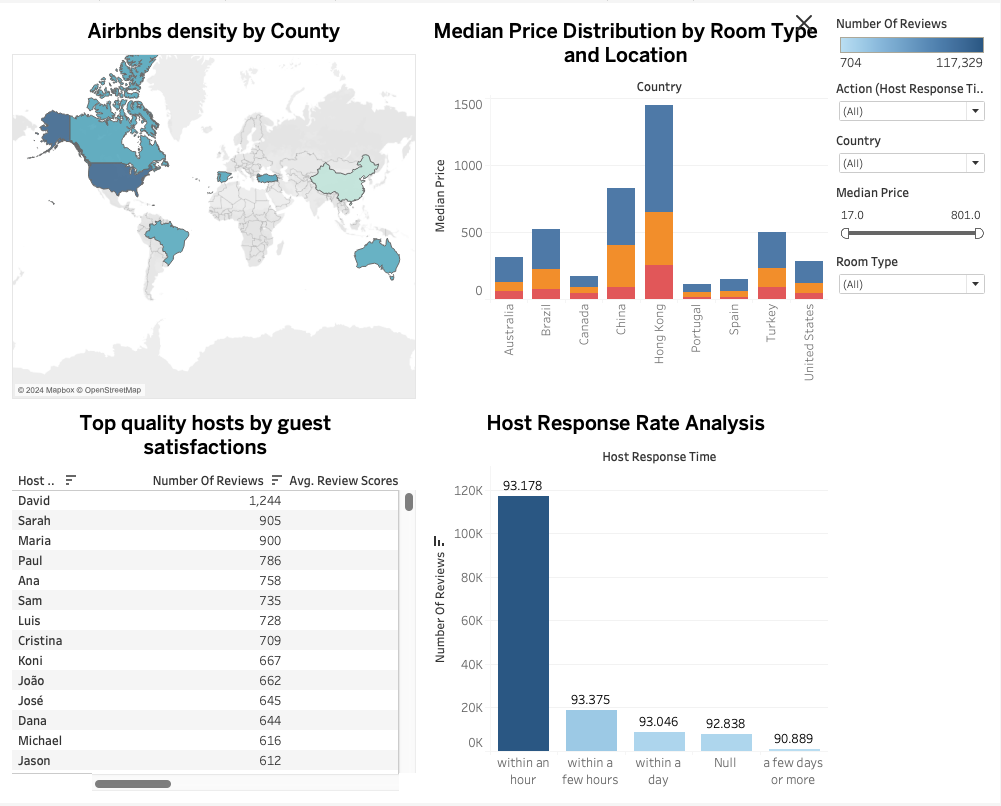


IMAGE 11 : DASHBOARD

R CODE :

#installing and loading the mongolite library to download the Airbnb data

#install.packages("mongolite") #need to run this line of code only once and then you can comment out

install.packages("tm")

install.packages("dplyr")

install.packages("syuzhet")

install.packages("wordcloud")

install.packages("igraph")

install.packages("ggraph")

install.packages("textdata")

install.packages("writexl")

install.packages("stringr")

library(mongolite)

library(dplyr)

library(tidyr)

library(tidytext)

library(ggplot2)

library(tm)

library(textdata)

library(topicmodels)

library(writexl)

library(stringr)

# This is the connection\_string. You can get the exact url from your MongoDB cluster screen

#replace the <<user>> with your Mongo user name and <<password>> with the mongo password

#lastly, replace the <<server\_name>> with your MongoDB server name

connection\_string <- 'mongodb+srv://scompagnone:i7cEQ7YpcO0JPQ3j@cluster0.uew8phc.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0'

airbnb\_collection <- mongo(collection="listingsAndReviews", db="sample\_airbnb", url=connection\_string)

#Here's how you can download all the Airbnb data from Mongo

## keep in mind that this is huge and you need a ton of RAM memory

airbnb\_all <- airbnb\_collection$find()

# Print the columns

print(names(airbnb\_all))

# Printing column types

for (column\_name in colnames(airbnb\_all)) {

column\_type <- class(airbnb\_all[[column\_name]])

print(paste(column\_name, ":" ,column\_type))

}

############################

###### Data cleaning ######

############################

# Selecting columns based on their data types EXCEPT LISTS AND DFs

selected\_columns <- airbnb\_all[, !sapply(airbnb\_all, function(x) any(class(x) %in% c("list", "data.frame")))]

#####amenities#######

# converting into lists COLUMNS amenities AND reviews into strings

selected\_columns$amenities <- sapply(airbnb\_all$amenities, function(x) paste(x, collapse=", "))

selected\_columns$reviews <- sapply(airbnb\_all$reviews, function(x) paste(x, collapse=", "))

#####HOST#######

# Convert the 'host' column into a data frame where each property is prefixed with 'host\_'

host\_df <- do.call(rbind, lapply(airbnb\_all$host, function(x) {

as.data.frame(t(x))

})) %>%

# This renames the columns by adding 'host\_' prefix

setNames(paste0('host\_', names(.)))

# Transpose host\_df

transposed\_host\_df <- t(host\_df)

# Append transposed\_host\_df to selected\_columns

selected\_columns <- cbind(selected\_columns, transposed\_host\_df)

# convert list into string "host\_verifications"

selected\_columns$host\_verifications\_f <- sapply(selected\_columns$host\_verifications, function(x) paste(x, collapse=", "))

# Drop the column "host\_verifications"

selected\_columns <- subset(selected\_columns, select = -c(host\_verifications))

#####review\_scores#######

# Convert the 'review\_scores' column into a data frame where each property is prefixed with 'review\_scores\_'

review\_scores\_df <- do.call(rbind, lapply(airbnb\_all$review\_scores, function(x) {

as.data.frame(t(x))

})) %>%

# This renames the columns by adding 'review\_scores\_' prefix

setNames(paste0('review\_scores\_', names(.)))

# Transpose review\_scores\_df

transposed\_review\_scores\_df <- t(review\_scores\_df)

# Append transposed\_review\_scores\_df to selected\_columns

selected\_columns <- cbind(selected\_columns, transposed\_review\_scores\_df)

#####availability#######

# Convert the 'availability' column into a data frame where each property is prefixed with 'availability\_'

availability\_df <- do.call(rbind, lapply(airbnb\_all$availability, function(x) {

as.data.frame(t(x))

})) %>%

# This renames the columns by adding 'availability\_' prefix

setNames(paste0('availability\_', names(.)))

# Transpose availability\_df

transposed\_availability\_df <- t(availability\_df)

# Append transposed\_availability\_df to selected\_columns

selected\_columns <- cbind(selected\_columns, transposed\_availability\_df)

#####airbnb\_all$address$location$coordinates#######

selected\_columns$coordinates <- sapply(airbnb\_all$address$location$coordinates, function(x) paste(x, collapse=", "))

#####airbnb\_all$address$location#######

# Convert the 'location' column into a data frame where each property is prefixed with 'location\_'

location\_df <- do.call(rbind, lapply(airbnb\_all$address$location, function(x) {

as.data.frame(t(x))

})) %>%

# This renames the columns by adding 'location\_' prefix

setNames(paste0('location\_', names(.)))

# Transpose location\_df

transposed\_location\_df <- t(location\_df)

# Append transposed\_location\_df to selected\_columns

selected\_columns <- cbind(selected\_columns, transposed\_location\_df)

#####airbnb\_all$address#######

# Select the columns from airbnb\_all$address

selected\_from\_address <- airbnb\_all$address %>%

select(street, suburb, government\_area, market, country, country\_code)

# Check if selected\_columns is not null and has rows to ensure compatibility

if(!is.null(selected\_columns) && nrow(selected\_columns) > 0) {

# Bind the columns to selected\_columns

selected\_columns <- bind\_cols(selected\_columns, selected\_from\_address)

} else {

# If selected\_columns is null or empty, simply assign the selected columns to it

selected\_columns <- selected\_from\_address

}

selected\_columns$coordinates\_l <- sapply(selected\_columns$coordinates...66, function(x) paste(x, collapse=", "))

selected\_columns$is\_location\_exact\_l <- sapply(selected\_columns$is\_location\_exact, function(x) paste(x, collapse=", "))

selected\_columns <- selected\_columns %>%

select(-is\_location\_exact, -`coordinates...66`,-type)

# Print the columns

print(names(selected\_columns))

# Find indices of columns containing "url"

url\_columns <- grep("url", names(selected\_columns))

# Drop columns containing "url"

selected\_columns <- selected\_columns[, -url\_columns]

# Define the column types

column\_types <- list(

"host\_id" = "integer",

"host\_name" = "character",

"host\_about" = "character",

"host\_response\_time" = "character",

"host\_neighbourhood" = "character",

"host\_response\_rate" = "character",

"host\_identity\_verified" = "logical",

"host\_listings\_count" = "integer",

"host\_total\_listings\_count" = "integer",

"host\_verifications\_f" = "character",

"host\_has\_profile\_pic"= "logical",

"host\_is\_superhost" = "logical",

"host\_location" = "character"

)

# Convert each column to the specified type

for (column\_name in names(column\_types)) {

selected\_columns[[column\_name]] <- as(selected\_columns[[column\_name]], column\_types[[column\_name]])

}

# Print the data types of the columns after conversion

for (column\_name in colnames(selected\_columns)) {

column\_type <- class(selected\_columns[[column\_name]])

print(paste(column\_name, ":" ,column\_type))

}

# Selecting specific columns

cleaned\_for\_tableau\_columns <- selected\_columns[, c(

"name",

"property\_type",

"room\_type",

"bed\_type",

"minimum\_nights",

"maximum\_nights",

"cancellation\_policy",

"last\_scraped",

"calendar\_last\_scraped",

"first\_review",

"last\_review",

"accommodates",

"bedrooms",

"beds",

"number\_of\_reviews",

"bathrooms",

"price",

"security\_deposit",

"cleaning\_fee",

"extra\_people",

"guests\_included",

"weekly\_price",

"monthly\_price",

"reviews\_per\_month",

"amenities",

"host\_id",

"host\_name",

"host\_location",

"host\_response\_time",

"host\_neighbourhood",

"host\_response\_rate",

"host\_is\_superhost",

"host\_has\_profile\_pic",

"host\_identity\_verified",

"host\_listings\_count",

"host\_total\_listings\_count",

"host\_verifications\_f",

"review\_scores\_accuracy",

"review\_scores\_cleanliness",

"review\_scores\_checkin",

"review\_scores\_communication",

"review\_scores\_location",

"review\_scores\_value",

"review\_scores\_rating",

"availability\_30",

"availability\_60",

"availability\_90",

"availability\_365",

"coordinates...64",

"street",

"suburb",

"government\_area",

"market",

"country",

"country\_code",

"coordinates\_l",

"is\_location\_exact\_l"

)]

# Writing to Excel

write\_xlsx(cleaned\_for\_tableau\_columns, path = "selected\_columns.xlsx")

#################

###### EDA ######

#################

# Print remaining column names

print(names(selected\_columns))

# Summary statistics for numerical columns

summary(selected\_columns[, c("minimum\_nights", "maximum\_nights", "accommodates", "bedrooms", "beds", "number\_of\_reviews", "bathrooms", "price")])

# Frequency distributions for categorical columns4

table(selected\_columns$room\_type)

table(selected\_columns$bed\_type)

table(selected\_columns$cancellation\_policy)

# Plot for room\_type

ggplot(selected\_columns, aes(x = room\_type)) +

geom\_bar() +

labs(title = "Frequency distribution of room types") +

theme\_minimal()

# Plot for bed\_type

ggplot(selected\_columns, aes(x = bed\_type)) +

geom\_bar() +

labs(title = "Frequency distribution of bed types") +

theme\_minimal()

# Plot for cancellation\_policy with rotated x-axis labels

ggplot(selected\_columns, aes(x = cancellation\_policy)) +

geom\_bar() +

labs(title = "Frequency distribution of cancellation policies") +

theme\_minimal() +

theme(axis.text.x = element\_text(angle = 45, hjust = 1))

################################

###### DATA PREPROCESSING ######

################################

# 1. Text Importing and Concatenation

data <- selected\_columns

# Combine relevant text columns into one for analysis

data$combined\_text <- paste(data$name, data$summary, data$description, data$neighborhood\_overview, data$notes, data$transit, data$reviews, sep = " ")

# 2. Text Tokenization and Cleaning

library(tidytext)

library(stringr)

library(dplyr)

tokens <- data %>%

unnest\_tokens(word, combined\_text) %>%

anti\_join(stop\_words, by = "word") %>%

filter(str\_detect(word, "[a-z]"))

# Adjusting tokenization to include a document identifier for further analyses

tokens <- data %>%

mutate(listing\_id = row\_number()) %>%

unnest\_tokens(word, combined\_text) %>%

anti\_join(stop\_words, by = "word") %>%

filter(str\_detect(word, "[a-z]")) %>%

select(listing\_id, word)

################################

####### FREQUENCY ANALYSIS #####

################################

# 3. Word Frequency Analysis

word\_freq <- tokens %>%

count(word, sort = TRUE)

# 4. Visualization of Word Frequency

library(ggplot2)

ggplot(word\_freq %>%

top\_n(20), aes(x = reorder(word, n), y = n)) +

geom\_col() +

coord\_flip() +

labs(x = "Word", y = "Frequency", title = "Top 20 Words in Airbnb Listings")

################################

######## SENTIMENT ANALYSIS ####

################################

# 5. Sentiment Analysis

# Load AFINN sentiments

sentiments <- get\_sentiments("afinn")

# Join tokens with sentiments for overall sentiment analysis

token\_sentiment <- tokens %>%

inner\_join(sentiments, by = "word")

# Calculate overall sentiment score

overall\_sentiment\_score <- token\_sentiment %>%

summarise(total\_sentiment = sum(value))

# Bing Sentiment Analysis

bing\_sentiments <- get\_sentiments("bing")

bing\_score <- tokens %>%

inner\_join(bing\_sentiments, by = "word") %>%

count(listing\_id, sentiment, sort = TRUE) %>%

spread(sentiment, n, fill = 0) %>%

mutate(bing\_sentiment\_score = positive - negative)

################################

########### TF-IDF #############

################################

# 6. TF-IDF Framework

# Now recalculating term frequency with the correct document identifier

tf <- tokens %>%

count(listing\_id, word, sort = TRUE) %>%

ungroup()

# Calculate TF-IDF with the correct structure

tf\_idf <- tf %>%

bind\_tf\_idf(word, listing\_id, n)

# Visualization of words with the highest TF-IDF scores

tf\_idf %>%

arrange(desc(tf\_idf)) %>%

mutate(word\_length = str\_length(word)) %>%

ggplot(aes(x = word\_length, y = tf\_idf)) +

geom\_point() +

labs(x = "Word Length", y = "TF-IDF Score", title = "Word Length vs. TF-IDF Score")

################################

########## N-GRAMS #############

################################

# 7. N-grams and Tokenization

bigrams <- data %>%

unnest\_tokens(bigram, combined\_text, token = "ngrams", n = 2)

# Frequency analysis of bigrams

bigram\_freq <- bigrams %>%

count(bigram, sort = TRUE)

# Visualization of Top Bigrams

ggplot(bigram\_freq %>%

arrange(desc(n)) %>%

head(20), aes(x = reorder(bigram, n), y = n)) +

geom\_col(fill = "plum") +

coord\_flip() +

labs(title = "Top 20 Bigrams in Airbnb Listings",

x = "Bigram",

y = "Frequency") +

theme\_minimal()

################################

#### TOPIC MODELING & EMOTIONS ##

################################

# 8. Topic Modeling with LDA

library(topicmodels)

# For LDA, you first need to create a Document-Term Matrix (DTM)

dtm <- tokens %>%

count(listing\_id, word) %>%

cast\_dtm(listing\_id, word, n)

# Running LDA

lda\_model <- LDA(dtm, k = 2) # Adjust the number of topics (k) as needed

# 9. NRC Emotion Analysis

nrc\_sentiments <- get\_sentiments("nrc") %>%

filter(sentiment %in% c("positive", "negative", "anger", "anticipation", "disgust", "fear", "joy", "sadness", "surprise", "trust"))

nrc\_emotion\_score <- tokens %>%

left\_join(nrc\_sentiments, by = "word") %>%

count(listing\_id, sentiment) %>%

spread(sentiment, n, fill = 0)

# Summarize and visualize emotions

nrc\_emotion\_summary <- nrc\_emotion\_score %>%

gather(sentiment, count, -listing\_id) %>%

group\_by(sentiment) %>%

summarize(total = sum(count)) %>%

mutate(percentage = total / sum(total) \* 100) %>%

arrange(desc(total))

# Plot

ggplot(nrc\_emotion\_summary, aes(x = reorder(sentiment, total), y = total, fill = sentiment)) +

geom\_bar(stat = "identity") +

geom\_text(aes(label = paste0(round(percentage, 1), "%"), y = total + 1),

vjust = -0.5, size = 3) +

coord\_flip() +

labs(title = "Distribution of Emotions in Airbnb Listings",

x = "Emotion",

y = "Total Count") +

scale\_fill\_brewer(palette = "Set3") +

theme\_minimal()