````
# =========== Restaurant Reviews Sentiment Analysis ==========
# Author: Sharad Thing
# Description: This script performs data cleaning, exploration,
#              and sentiment analysis on restaurant review data using caret ML model
# ============================================================


# ----------------------
# 1. Loading Required Libraries
# ----------------------

```{r}

# This section loads all necessary packages for data manipulation,
# text processing, visualization, and machine learning

library(readr)      # For reading CSV files
library(tidyr)      # For data tidying
library(dplyr)      # For data manipulation
library(tm)         # For text mining
library(SnowballC)  # For text stemming
library(stringr)    # For string operations
library(caret)      # For machine learning
library(lubridate)  # For date/time handling
library(skimr)      # For data summary
library(ggplot2)    # For data visualization



```

# ----------------------
# 2. Data Loading & Initial Inspection
# ----------------------


```{r}

# Read the raw restaurant reviews data
reviews <- read_csv("Restaurant_reviews.csv")


```


```{r}
# This Checks for any parsing problems in the data

problems(reviews)

```



```{r}
# Examine column names to understand data structure
colnames(reviews)

```
# ----------------------
# 3. Data Cleaning & Inhancing
# ----------------------

```{r}
#Remove unnecessary column
reviews <- reviews %>% select(-`7514`)
colnames(reviews)
# Remove rows with missing values and drop Pictures column since we dont need picture to analysis of sentiment
reviews <- reviews %>%
  drop_na() %>%
  select(-Pictures)

```

```{r}
# Check for missing values in the dataset
any(is.na(reviews))
colSums(is.na(reviews))
reviews_with_na <- reviews %>% filter(if_any(everything(), is.na))

# Number of rows with any missing values
nrow(reviews_with_na)

```



```{r}
# Remove rows with missing values and drop Pictures column since we dont need picture to analysis of sentiment
reviews <- reviews %>%
  drop_na() %>%
  select(-Pictures)


```


```{r}
# Verify cleaning results
any(is.na(reviews))
nrow(reviews_with_na)            # Shows how many were removed
````
````

````
dim(reviews)                     # New dimensions of the cleaned data

```


```{r}
# Saves cleaned data
write_csv(reviews, "Cleaned_Restaurant_Reviews.csv")

```

# ----------------------
# 4.  Descriptive Data Analysis
# ----------------------

```{r}

# Create restaurant summary statistics
restaurant_summary <- reviews %>%
  group_by(Restaurant) %>%
  summarise(
    TotalReviews = n(),
    AvgRating = round(mean(Rating, na.rm = TRUE), 2),
    MedianRating = median(Rating, na.rm = TRUE),
    AvgLength = round(mean(nchar(Review), na.rm = TRUE), 2)
  ) %>%
  arrange(desc(TotalReviews))

View(restaurant_summary)

```

# ----------------------
# 5.  Exploratory Data Analysis (EDA)
# ----------------------

```{r}

# Analyze review timing patterns
reviews$Time <- mdy_hm(reviews$Time)  # Converts character to datetime
reviews$Hour <- hour(reviews$Time)    # Extracts hour (0-23)

```

```{r}
# Plot review frequency by hour

reviews$Weekday <- wday(reviews$Time, label = TRUE)


ggplot(reviews, aes(x = Hour)) +
  geom_bar(fill = "steelblue") +
  labs(title = "Number of Customer's Reviews by Hour of Day (Busy Times)",
       x = "Hour (0-23)",
       y = "Number of Customers Reviews") +
  theme_minimal()



```



--
# Create Visual categorical satisfaction levels based on ratings

```{r}
# Create SatisfactionLevel column based on Rating
reviews <- reviews %>%
  mutate(SatisfactionLevel = case_when(
    Rating <= 2 ~ "Bad",
    Rating == 3 ~ "Moderate",
    Rating == 4 ~ "Good",
    Rating == 5 ~ "Excellent",
    TRUE ~ NA_character_
  ))

```



```{r}
# Top 5 Excellent Restaurants
top_excellent <- reviews %>%
  filter(SatisfactionLevel == "Excellent") %>%
  count(Restaurant, sort = TRUE) %>%
  slice_max(n, n = 5)

# Top 5 Bad Restaurants
top_bad <- reviews %>%
  filter(SatisfactionLevel == "Bad") %>%
  count(Restaurant, sort = TRUE) %>%
  slice_max(n, n = 5)



top_combined <- bind_rows(
  top_excellent %>% mutate(Level = "Excellent"),
  top_bad %>% mutate(Level = "Bad")
)
````

````
library(ggplot2)

ggplot(top_combined, aes(x = reorder(Restaurant, n), y = n, fill = Level)) +
  geom_col() +
  coord_flip() +
  facet_wrap(~ Level, scales = "free_y") +
  labs(
    title = "Top 5 Restaurants with Highest Excellent & Bad Ratings",
    x = "Restaurant",
    y = "Number of Reviews"
  ) +
  theme_minimal()


```



#  Top Reviewer/loyal customer Analysis

```{r}

# Show top 15 reviewers
top_loyal_customers <- loyalty_data %>% slice_max(TotalReviews, n = 15)

ggplot(top_loyal_customers, aes(x = reorder(Reviewer, TotalReviews), y = TotalReviews)) +
  geom_col(fill = "orange") +
  coord_flip() +
  labs(title = "Top Customers to give Reviews",
       x = "Customer",
       y = "Total Reviews") +
  theme_minimal()

```

# Rating Distribution Analysis


```{r}
# Visualize rating distribution

ggplot(reviews, aes(x = Rating)) +
  geom_density(fill = "skyblue", alpha = 0.5) +
  labs(title = "Density of Ratings",
       x = "Rating",
       y = "Density") +
  theme_minimal()

```



# ============================================
# PART 2: SENTIMENT ANALYSIS USING MACHINE LEARNING
# ============================================

# ----------------------
# 6. Text Preprocessing
# ----------------------

```{r}
# Initialize sentiment column
reviews$Sentiments <- NA

# Confirm
colnames(reviews)

```



```{r}
# View structure and summary
str(reviews)
summary(reviews)
head(reviews)
```



```{r}
# Create text corpus from reviews
corpus <- VCorpus(VectorSource(reviews$Review))

```

```{r}
# Clean text data through multiple steps:

# Step 1: Convert to lowercase
corpus_clean <- tm_map(corpus, content_transformer(tolower))
# Step 2: Remove numbers
corpus_clean <- tm_map(corpus_clean, removeNumbers)
# Step 3: Remove punctuation
corpus_clean <- tm_map(corpus_clean, removePunctuation)
corpus_clean <- tm_map(corpus_clean, content_transformer(function(x) gsub("’|‘|`|'", "", x)))
# Step 4: Remove common stopwords like "the", "is", etc.
corpus_clean <- tm_map(corpus_clean, removeWords, stopwords("english"))
# Step 5: Remove extra spaces
corpus_clean <- tm_map(corpus_clean, stripWhitespace)
# Step 6: Apply stemming (e.g., "amazing", "amazingly" → "amaz")
#corpus_clean <- tm_map(corpus_clean, stemDocument)

```



```{r}
````

````
# Inspect cleaned text
inspect(corpus_clean[[1]])

```

```{r}
# Create the Document-Term Matrix (DTM)
dtm <- DocumentTermMatrix(corpus_clean)

```

```{r}
# Examine DTM
dtm

```

```{r}
# Removes the last 10 less frequent words

dtm_sparse <- removeSparseTerms(dtm, 0.99)

# Find removed terms
full_terms <- Terms(dtm)
sparse_terms <- Terms(dtm_sparse)

removed_terms <- setdiff(full_terms, sparse_terms)
head(removed_terms, 10)

```

```{r}
# Convert DTM to a matrix
dtm_matrix <- as.matrix(dtm)

# Sum each word’s frequency across all documents
word_freq <- colSums(dtm_matrix)

# Sort by frequency (highest to lowest)
word_freq <- sort(word_freq, decreasing = TRUE)

# View the top 20 most frequent words for training training set
head(word_freq, 200)

```


# ----------------------
# 7. Initial Sentiment Labeling
# ----------------------

```{r}
# Define keyword lists for sentiment classification
positive_keywords <- c( "polite","wonderful", "good", "excellent", "awesome", "quick")
negative_keywords <- c("bad", "worst", "didnt", "poor", "rude", "tasteless")

```


```{r}
# Label sentiments based on rating and keywords

reviews$Sentiments <- case_when(
  str_detect(tolower(reviews$Review), paste(positive_keywords, collapse = "|")) ~ "Positive Review",
  str_detect(tolower(reviews$Review), paste(negative_keywords, collapse = "|")) ~ "Negative Review",
  TRUE ~ NA_character_
)


```



```{r}

# Save partially labeled dataset
#na for rest
table(reviews$Sentiments, useNA = "ifany")
write_csv(reviews, "Test_Dataset_Sentiments_Restaurant_Reviews.csv")

```




#............#Machine Learning Setup................




```{r}
#This will load the test dataset and leave na Sentiments
train_reviews <- reviews %>% filter(!is.na(Sentiments))

# Split into labeled (train/test) and unlabeled data

library(caret)
set.seed(123)  # For reproducibility (takes same values)

# Create a split index (80% train, 20% test)
train_index <- createDataPartition(train_reviews$Sentiments, p = 0.8, list = FALSE)
````

````
# Split the data
train_split <- train_reviews[train_index, ]
test_split  <- train_reviews[-train_index, ]

```



```{r}
# Preprocess training text data
train_corpus <- VCorpus(VectorSource(train_split$Review))

train_corpus_clean <- train_corpus %>%
  tm_map(content_transformer(tolower)) %>%
  tm_map(content_transformer(function(x) gsub("’|‘|`|'", "", x))) %>%
  tm_map(removeNumbers) %>%
  tm_map(removePunctuation) %>%
  tm_map(removeWords, stopwords("english")) %>%
  tm_map(stripWhitespace)

```

```{r}

# Preprocess test text data (using same transformations)

test_corpus <- VCorpus(VectorSource(test_split$Review))

test_corpus_clean <- test_corpus %>%
  tm_map(content_transformer(tolower)) %>%
  tm_map(content_transformer(function(x) gsub("’|‘|`|'", "", x))) %>%
  tm_map(removeNumbers) %>%
  tm_map(removePunctuation) %>%
  tm_map(removeWords, stopwords("english")) %>%
  tm_map(stripWhitespace)

```

```{r}
# Training DTM
train_dtm <- DocumentTermMatrix(train_corpus_clean)
train_dtm <- removeSparseTerms(train_dtm, 0.99)  # Removes rarely used words

# Test DTM - use same terms as training
test_dtm <- DocumentTermMatrix(test_corpus_clean, control = list(dictionary = Terms(train_dtm)))

```

```{r}
# Training set
train_data <- as.data.frame(as.matrix(train_dtm))
train_data$Sentiments <- train_split$Sentiments

# Test set
test_data <- as.data.frame(as.matrix(test_dtm))
test_labels <- test_split$Sentiments

```



# ----------------------
# 12. Model Training & Evaluation
# ----------------------

```{r}
#1library(e1071)

#nb_model <- naiveBayes(Label ~ ., data = train_data)


# Set up training control
ctrl <- trainControl(method = "cv", number = 10 ,verboseIter = TRUE)  # 5-fold cross-validation

# Train model using caret with, say, Random Forest (rf)
model <- train(
  Sentiments ~ .,
  data = train_data,
  method = "svmLinear" ,
  trControl = ctrl
)

# Predict on test data
predictions <- predict(model, newdata = test_data)


```



```{r}
# Ensure same levels and factor type
predictions <- factor(predictions, levels = c("Positive Review", "Negative Review"))
test_labels <- factor(test_labels, levels = c("Positive Review", "Negative Review"))

```

```{r}
# Evaluate model performance

confusionMatrix(predictions, test_labels)

```
````
````

````
```{r}

# Sentiment distribution

ggplot(final_reviews, aes(x = Sentiments, fill = Sentiments)) +
  geom_bar() +
  theme_minimal() +
  labs(
    title = "Sentiment Distribution of All Reviews",
    x = "Sentiment",
    y = "Number of Reviews"
  )

```


```{r}
# Sentiment pie chart

final_reviews %>%
  count(Sentiments) %>%
  ggplot(aes(x = "", y = n, fill = Sentiments)) +
  geom_col(width = 1) +
  coord_polar("y") +
  theme_void() +
  labs(title = "Sentiment Breakdown of All Reviews")

```


```{r}

# Sentiment by rating

ggplot(final_reviews, aes(x = as.factor(Rating), fill = Sentiments)) +
  geom_bar(position = "dodge") +
  labs(title = "Sentiment by Rating",
       x = "Star Rating",
       y = "Number of Reviews") +
  theme_minimal()

```
# ----------------------
# 13. Labeling Unlabeled Data in CSV
# ----------------------


```{r}
#This will load the na label reviews
unlabeled_reviews <- reviews %>% filter(is.na(Sentiments))


```


```{r}

# Preprocess unlabeled data (same as training)
unlabeled_corpus <- VCorpus(VectorSource(unlabeled_reviews$Review))

unlabeled_corpus_clean <- unlabeled_corpus %>%
  tm_map(content_transformer(tolower)) %>%
  tm_map(content_transformer(function(x) gsub("’|‘|`|'", "", x))) %>%
  tm_map(removeNumbers) %>%
  tm_map(removePunctuation) %>%
  tm_map(removeWords, stopwords("english")) %>%
  tm_map(stripWhitespace)


```

```{r}
# Create DTM using training vocabulary
unlabeled_dtm <- DocumentTermMatrix(unlabeled_corpus_clean, control = list(dictionary = Terms(train_dtm)))
unlabeled_data <- as.data.frame(as.matrix(unlabeled_dtm))
unlabeled_predictions <- predict(model, newdata = unlabeled_data)
unlabeled_reviews$Sentiments <- unlabeled_predictions



```



```{r}
# Combine all reviews into final dataset
final_reviews <- bind_rows(
  reviews %>% filter(!is.na(Sentiments)),  # Already labeled (train + test)
  unlabeled_reviews                   # Now machine-labeled
)


# Save final results
write_csv(final_reviews, "Full_Final_Labeled_Restaurant_Reviews.csv")
```


# ============================================
# PART 3: Aspect-Based Sentiment Classification (Food, Service, Price,Other)
# ============================================
````

````
```{r}

# Load the already labeled dataset
show_col_types = FALSE
reviews <- read_csv("Full_Final_Labeled_Restaurant_Reviews.csv")



```


```{r}
# Filter positive reviews only
positive_reviews <- reviews %>%
  filter(Sentiments == "Positive Review")

data("stop_words")

positive_words <- positive_reviews %>%
  unnest_tokens(word, Review) %>%
  anti_join(stop_words, by = "word")  # Remove common stopwords

positive_word_counts <- positive_words %>%
  count(word, sort = TRUE)

head(positive_word_counts, 100)  # Top 20 words

```
```{r}
negative_reviews <- reviews %>%
  filter(Sentiments == "Negative Review")


data("stop_words")

negative_words <- negative_reviews %>%
  unnest_tokens(word, Review) %>%
  anti_join(stop_words, by = "word")  # Remove common stopwords

negative_word_counts <- negative_words %>%
  count(word, sort = TRUE)

head(negative_word_counts, 100)  # View top 100 negative keywords

```

```{r}
reviews$Sentiment_Description <- NA

```
````

#Add Descriptive Categories to Positive and Negative Reviews based on frequent words from the previous trained set (positive and Negative respectively)

````
```{r}

reviews <- reviews %>%
  mutate(Sentiment_Description = case_when(
    # Positive Review mappings
    Sentiments == "Positive Review" & str_detect(tolower(Review),
"food|chicken|taste|veg|biryani|starters|rice|tasty|menu|paneer|buffet|delicious|dishes|spicy|lunch|fish|tasted|drinks|fried|pizza|di
~ "Good Food",
    Sentiments == "Positive Review" & str_detect(tolower(Review),
"service|ambience|experience|people|serve|time|zomato|friedns|friendly|staff|quick|polite|service|seating|polite") ~ "Good Service",
    Sentiments == "Positive Review" & str_detect(tolower(Review),
"affordable|price|cheap|worth|reasonable|value|varity|quantity|options") ~ "Affordable",
    Sentiments == "Positive Review" & str_detect(tolower(Review), "5|4|3|music") ~ "Other",

    # Negative Review mappings
    Sentiments == "Negative Review" & str_detect(tolower(Review),
"food|chicken|taste|veg|biryani|starters|rice|tasty|menu|paneer|buffet|delicious|dishes|spicy|lunch|fish|tasted|drinks|fried|pizza|di
food|smelly|spoiled|burnt|not tasty|not fresh|worst food") ~ "Bad Food",
    Sentiments == "Negative Review" & str_detect(tolower(Review),
"service|ambience|experience|people|serve|time|zomato|friedns|friendly|staff|quick|polite|service|seating|polite|rude|slow|late|unpro
service|careless|ignore|worst service|not helpful") ~ "Bad Service",
    Sentiments == "Negative Review" & str_detect(tolower(Review),
"affordable|price|cheap|worth|reasonable|value|varity|quantity|options|expensive|overpriced|costly|too much|not worth|high
price|waste of money") ~ "Overpriced",
    Sentiments == "Positive Review" & str_detect(tolower(Review), "5|4|3|music") ~ "Other",
    TRUE ~ Sentiment_Description  # preserve previous matches
  ))

```


```{r}

table(reviews$Sentiment_Description, useNA = "ifany")

```


```{r}
library(ggplot2)

reviews %>%
  filter(!is.na(Sentiment_Description)) %>%
  count(Sentiment_Description) %>%
````
````

````
  ggplot(aes(x = reorder(Sentiment_Description, n), y = n)) +
  geom_col(fill = "tomato") +
  coord_flip() +
  labs(
    title = "Review Categories by Sentiment Description",
    x = "Sentiment Category",
    y = "Number of Reviews"
  )


```


```{r}
# Filter reviews with category
ml_data <- reviews %>% filter(!is.na(Sentiment_Description))

# Split into train/test
set.seed(123)
train_index <- createDataPartition(ml_data$Sentiment_Description, p = 0.8, list = FALSE)
train_split <- ml_data[train_index, ]
test_split  <- ml_data[-train_index, ]



```


```{r}
write_csv(reviews, "/Users/nicktamang/UniversityY3/DATA-ANALYSIS/CW2/Training_Sentiment_Descriptions.csv")

```


```{r}

# ===============================================
# PART 3: MULTI-CLASS SENTIMENT CATEGORY CLASSIFICATION
# ===============================================



# ----------------------
# 3. Preprocess Text Data (Train)
# ----------------------
train_corpus <- VCorpus(VectorSource(train_split$Review))

train_corpus_clean <- train_corpus %>%
  tm_map(content_transformer(tolower)) %>%
  tm_map(content_transformer(function(x) gsub("'", "", x))) %>% # Simplified
  tm_map(removeNumbers) %>%
  tm_map(removePunctuation) %>%
  tm_map(removeWords, stopwords("english")) %>%
  tm_map(stripWhitespace)

# ----------------------
# 4. Preprocess Text Data (Test)
# ----------------------
test_corpus <- VCorpus(VectorSource(test_split$Review))

test_corpus_clean <- test_corpus %>%
  tm_map(content_transformer(tolower)) %>%
  tm_map(content_transformer(function(x) gsub("'", "", x))) %>%
  tm_map(removeNumbers) %>%
  tm_map(removePunctuation) %>%
  tm_map(removeWords, stopwords("english")) %>%
  tm_map(stripWhitespace)

# ----------------------
# 5. Create Document-Term Matrices
# ----------------------
train_dtm <- DocumentTermMatrix(train_corpus_clean)
train_dtm <- removeSparseTerms(train_dtm, 0.99) # Keep most frequent terms

test_dtm <- DocumentTermMatrix(test_corpus_clean,
                             control = list(dictionary = Terms(train_dtm))) # Use same features

# ----------------------
# 6. Convert to DataFrames
# ----------------------
train_data <- as.data.frame(as.matrix(train_dtm))
train_data$Sentiment_Description <- train_split$Sentiment_Description

test_data <- as.data.frame(as.matrix(test_dtm))
test_labels <- factor(test_split$Sentiment_Description)

# Ensure same factor levels
train_data$Sentiment_Description <- factor(train_data$Sentiment_Description,
                                          levels = levels(test_labels))

# ----------------------
# 7. Train Multi-Class Classifier (SVM)
# ----------------------
ctrl <- trainControl(method = "cv", number = 10, verboseIter = TRUE)

model <- train(
  Sentiment_Description ~ .,
  data = train_data,
  method = "svmLinear", # Or try "naive_bayes" for comparison
  trControl = ctrl
)
````

````
# ----------------------
# 8. Predict & Evaluate
# ----------------------
predictions <- predict(model, newdata = test_data)
conf_matrix <- confusionMatrix(predictions, test_labels)


```

```{r}
# Ensure same factor levels
predictions <- factor(predictions, levels = levels(test_labels))
test_labels <- factor(test_labels, levels = levels(test_labels))

```

```{r}
# Confusion matrix for multi-class classification
conf_matrix <- confusionMatrix(predictions, test_labels)
print(conf_matrix)

```
# ----------------------
# 14. Final Visualizations
# ----------------------


```{r}


#To get the confusion matrix results
conf_matrix <- confusionMatrix(predictions, test_labels)

# Extract overall accuracy and format as percentage
accuracy <- conf_matrix$overall['Accuracy']
accuracy_percent <- round(accuracy * 100, 2)  # Rounds to 2 decimal places

# Prints the accuracy
cat(paste0("\nModel Accuracy: ", accuracy_percent, "%\n"))

# For more detailed metrics:
metrics <- data.frame(
  "Metric" = c("Accuracy", "Kappa", "Precision", "Recall", "F1"),
  "Value" = c(
    round(conf_matrix$overall['Accuracy'] * 100, 2),
    round(conf_matrix$overall['Kappa'] * 100, 2),
    round(mean(conf_matrix$byClass[,'Precision']) * 100, 2),
    round(mean(conf_matrix$byClass[,'Recall']) * 100, 2),
    round(mean(conf_matrix$byClass[,'F1']) * 100, 2)
  )
)

# Prints formatted metrics
print(metrics)

# Visual representation (simple bar plot)
ggplot(metrics, aes(x = Metric, y = Value, fill = Metric)) +
  geom_col() +
  geom_text(aes(label = paste0(Value, "%")), vjust = -0.5) +
  labs(title = "Model Performance Metrics",
       subtitle = paste("Overall Accuracy:", accuracy_percent, "%"),
       y = "Percentage") +
  ylim(0, 100) +
  theme_minimal() +
  theme(legend.position = "none")

```
```{r}
# 1. Ensures factor levels match between predictions and actuals
predictions <- factor(predictions,
                      levels = c("Good Food", "Good Service", "Affordable",
                                 "Bad Food", "Bad Service", "Overpriced", "Other"))
test_labels <- factor(test_labels,
                      levels = levels(predictions))  # Use same levels as predictions

# 2. Generates and print the confusion matrix
conf_matrix <- confusionMatrix(predictions, test_labels)
print(conf_matrix)

# 3. Extract and display accuracy metrics
accuracy <- conf_matrix$overall['Accuracy']
cat(sprintf("\nModel Accuracy: %.2f%%\n", accuracy*100))

# 4. Visualises the confusion matrix
library(ggplot2)
confusion_df <- as.data.frame(conf_matrix$table)

ggplot(confusion_df, aes(x = Reference, y = Prediction, fill = Freq)) +
  geom_tile(color = "white") +
  geom_text(aes(label = Freq), color = "black", size = 4) +
  scale_fill_gradient(low = "white", high = "steelblue") +
  labs(title = "Confusion Matrix",
       subtitle = paste("Overall Accuracy:", round(accuracy*100, 2), "%"),
       x = "Actual Category",
       y = "Predicted Category") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# 5. Calculates and displays per-class metrics
class_metrics <- data.frame(
  Class = names(conf_matrix$byClass[, "Recall"]),
  Precision = conf_matrix$byClass[, "Precision"],
````

````
  Recall = conf_matrix$byClass[, "Recall"],
  F1 = conf_matrix$byClass[, "F1"]
)

print(class_metrics)

# 6. Visualises per-class performance
library(tidyr)
class_metrics_long <- pivot_longer(class_metrics, cols = -Class, names_to = "Metric")

ggplot(class_metrics_long, aes(x = Class, y = value, fill = Metric)) +
  geom_bar(stat = "identity", position = "dodge") +
  scale_y_continuous(labels = scales::percent) +
  labs(title = "Per-Class Performance Metrics",
       y = "Score") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```
```{r}


```


#Sentiment_classification visulisation
```{r}
# Top 5 positive reviewers
top_positive_reviewers <- final_reviews %>%
  filter(Sentiments == "Positive Review") %>%
  count(Reviewer, sort = TRUE) %>%
  slice_max(n, n = 20) %>%
  pull(Reviewer)

# Get their reviews along with sentiment descriptions
top_positive_reviews <- final_reviews %>%
  filter(Reviewer %in% top_positive_reviewers, Sentiments == "Positive Review") %>%
  select(Reviewer, Restaurant, Rating, Review, Sentiments, Sentiment_Description)

# View the result
View(top_positive_reviews)

```

```{r}
top_negative_reviewers <- final_reviews %>%
  filter(Sentiments == "Negative Review") %>%
  count(Reviewer, sort = TRUE) %>%
  slice_max(n, n = 5) %>%
  pull(Reviewer)

top_negative_reviews <- final_reviews %>%
  filter(Reviewer %in% top_negative_reviewers, Sentiments == "Negative Review") %>%
  select(Reviewer, Restaurant, Rating, Review, Sentiments, Sentiment_Description)

View(top_negative_reviews)

```

# Step 8: Conclusion
#This sentiment analysis reveals key insights into customer perceptions of restaurants. Establishments with consistently positive
reviews often excel in areas like [food quality, service speed, or ambiance], while negative feedback commonly highlights issues
such as [long wait times, pricing, tasteless foods or cleanliness]. These findings help restaurant managers identify strengths to
maintain and weaknesses to address, ultimately improving customer satisfaction and business performance.




```{r}


# Step 8: Conclusion
#This sentiment analysis reveals key insights into customer perceptions of restaurants. Establishments with consistently positive
reviews often excel in areas like [food quality, service speed, or ambiance], while negative feedback commonly highlights issues
such as [long wait times, pricing, tasteless foods or cleanliness]. These findings help restaurant managers identify strengths to
maintain and weaknesses to address, ultimately improving customer satisfaction and business performance.

```

```{r}
```


```{r}

```
````
````