Data Wrangling Yelp Reviews

Brian Cervantes Alvarez
January 28, 2025

Project Overview

This report is all about taking Yelp data and making it cleaner and easier to use for analysis. The Yelp Academic Dataset has lots of JSON files, each one containing different information like reviews, businesses, user details, check-ins, and tips.

Extracting the Data

I started by downloading JSON files from Yelp, such as `yelp_academic_dataset_business.json` and `yelp_academic_dataset_review.json`, and then changed them into CSV files. This first step involved breaking down complex data so that each row in the CSV represents one specific piece of information.

Cleaning the Data

Next, I cleaned up the data by standardizing the column names. I removed prefixes like `attributes.` and split combined attributes like `Ambience` and `BusinessParking` into their own separate columns. To fix missing or incorrect data, I flagged or removed postal codes that didn't match the valid U.S. or Canadian formats and excluded any state abbreviations that weren't recognized. I also changed text values like `"true"` or `"false"` into actual Boolean values (True/False) and replaced entries like `"NA"` or `"None"` with proper missing value indicators (`NA`).

Exploring the Data

After cleaning, I took a quick look at the data to make sure it was easier to work with. The operating hours now have a consistent format (for example, `09:00:00 - 17:00:00`), and columns like `businessName`, `city`, and `state` are more straightforward. This better organization helps researchers quickly find out things like which businesses stay open late, which ones have features like wheelchair access or a live DJ, and how these features differ in different areas.

Conclusion

In addition to extracting and cleaning the data, I converted the original JSON files into CSV format using Python scripts that flatten the nested fields. Then, I turned the CSV back into JSON using R's `jsonlite` package. This back-and-forth process makes it easy to work with the dataset in different tools. I checked everything by loading the new JSON into pandas, and it all looked good—no missing columns or strange errors.

I chose the Yelp Academic Dataset because it's pretty challenging—I'd rate it about **6 out of 10**. It has nested data, a ton of rows, and there are actually **four** main JSON files total. So far, I've mostly focused on cleaning up just the `business.json` file, and that alone had over 90 columns once flattened. Even with just one file, there was a lot to tackle!

After the data was cleaned, I loaded the resulting CSV into R for a quick look. I made some histograms to check star ratings and plotted the business locations on a map. Everything worked smoothly, which means the dataset is now ready for more detailed analysis.

For my final project, I plan to **bring in the reviews** (from `review.json`) and do some **joins** between the business data and the review data. That way, I can dig deeper into how different factors—like having live music or certain pricing—might affect star ratings. I also want to see if certain comments in the text reviews match up with the ratings customers give. These steps will set the stage for more interesting questions and possibly even building a small recommendation system.

Yelp Business Reviews JSON Format (Initial Data)

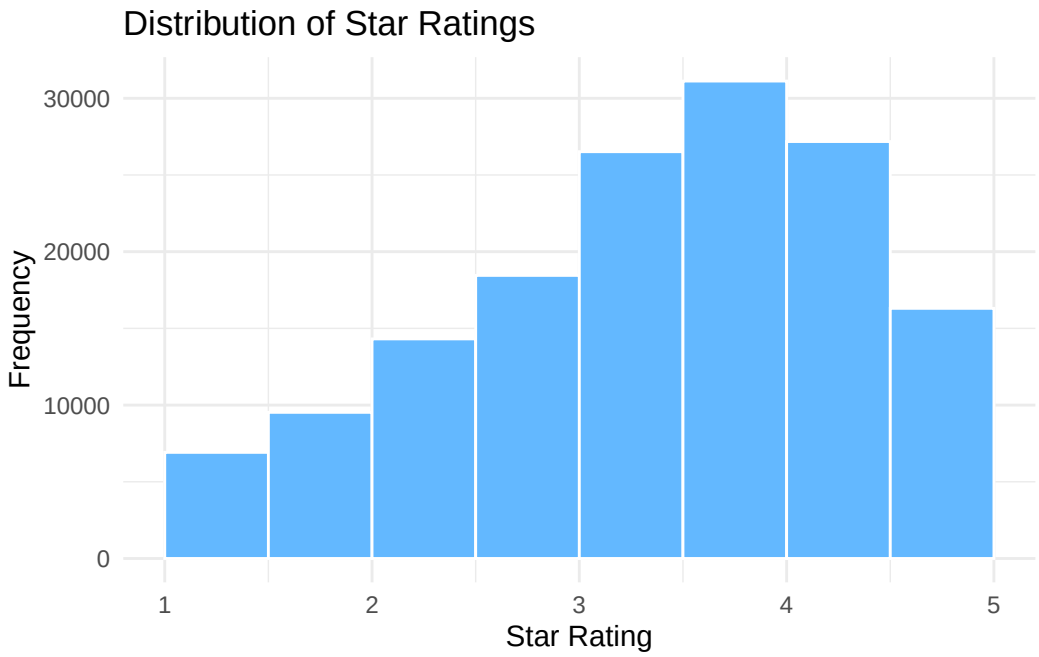

```
{
  "business_id": "Pns2l4eNsfO8kk83dixA6A",
  "name": "Abby Rappoport, LAC, CMQ",
  "address": "1616 Chapala St, Ste 2",
  "city": "Santa Barbara",
  "state": "CA",
  "postal_code": "93101",
  "latitude": 34.4266787,
  "longitude": -119.7111968,
  "stars": 5.0,
  "review_count": 7,
  "is_open": 0,
  "attributes": {
    "ByAppointmentOnly": "True"
  },
  "categories": "Doctors, Traditional Chinese Medicine,
      Naturopathic/Holistic, Acupuncture, Health & Medical,
      Nutritionists",
  "hours": null
}
{
  "business_id": "mpf3x-BjTdTEA3yCZrAYPw",
  "name": "The UPS Store",
  "address": "87 Grasso Plaza Shopping Center",
  "city": "Affton",
  "state": "MO",
  "postal_code": "63123",
  "latitude": 38.551126,
  "longitude": -90.335695,
  "stars": 3.0,
  "review_count": 15,
  "is_open": 1,
  "attributes": {
    "BusinessAcceptsCreditCards": "True"
  },
  "categories": "Shipping Centers, Local Services, Notaries, Mailbox
      Centers, Printing Services",
  "hours": {
    "Monday": "0:0-0:0",
    "Tuesday": "8:0-18:30",
```

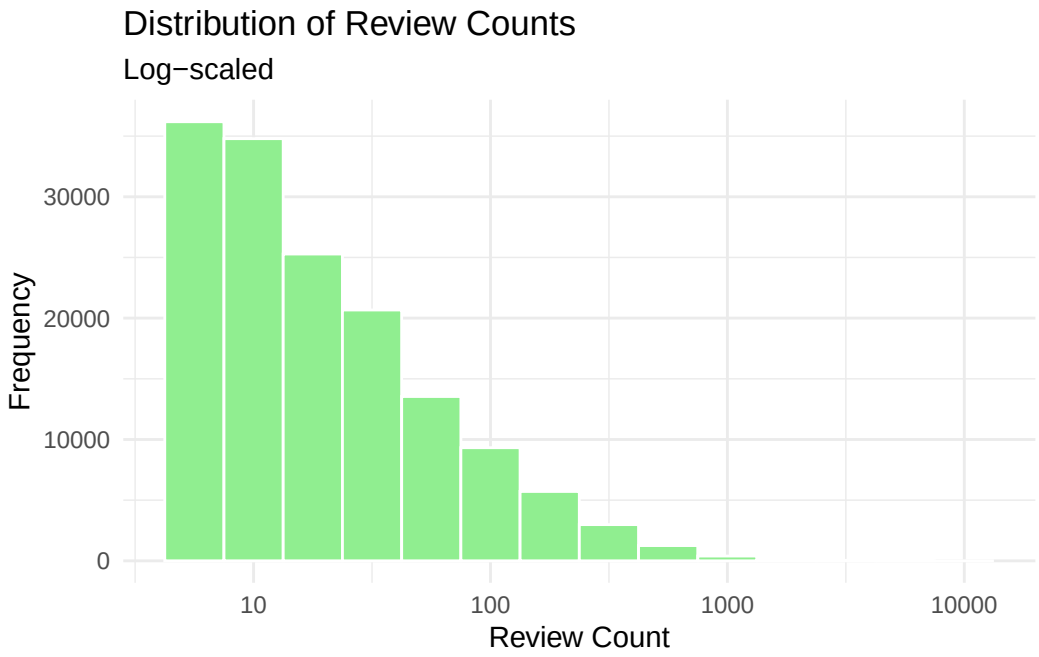
```
    "Wednesday": "8:0-18:30",
    "Thursday": "8:0-18:30",
    "Friday": "8:0-18:30",
    "Saturday": "8:0-14:0"
  }
}
```

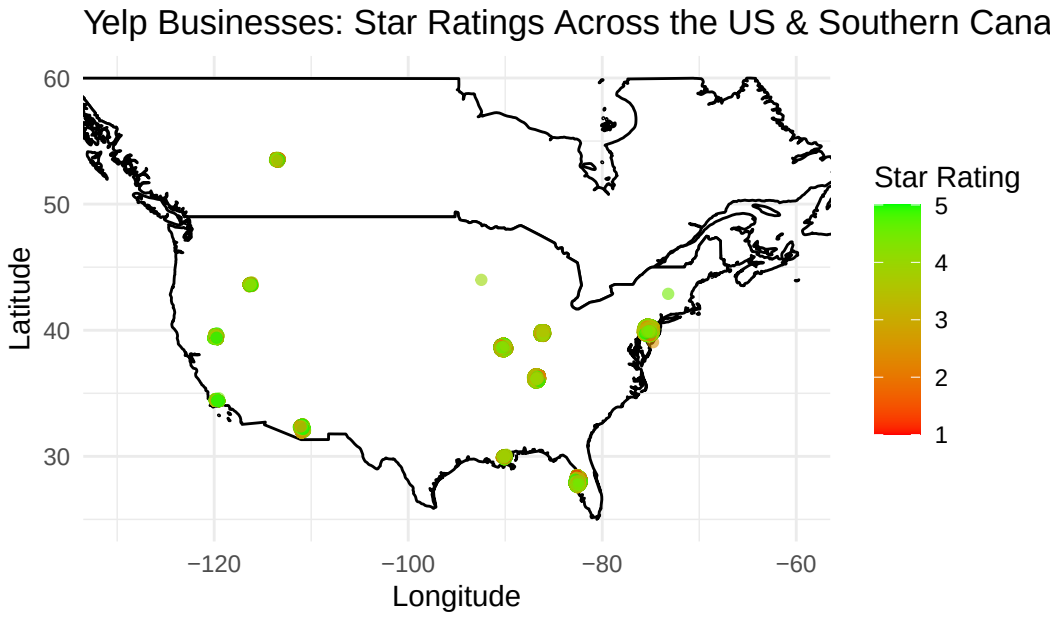
Yelp Business Reviews CSV Format (Cleaned Data)

```
Rows: 2
Columns: 15

$ businessId         <chr> "Pns2l4eNsfO8kk83dixA6A",
    "mpf3x-BjTdTEA3yCZrAYPw"
$ businessName       <chr> "Abby Rappoport, LAC, CMQ", "The UPS Store"
$ address            <chr> "1616 Chapala St, Ste 2", "87 Grasso Plaza
    Shopping Center"
$ city               <chr> "Santa Barbara", "Affton"
$ state              <chr> "CA", "MO"
$ usPostalCode       <chr> "93101", "63123"
$ latitude           <dbl> 34.42668, 38.55113
$ longitude          <dbl> -119.71120, -90.33570
$ starRating         <dbl> 5, 3
$ reviewCount        <dbl> 7, 15
$ isOpen             <dbl> 0, 1
$ categories         <chr> "Doctors, Traditional Chinese Medicine,
    Naturopathic/Holistic, Acupuncture, Health & Medical,
    Nutritionists",
                           "Shipping Centers, Local Services, Notaries,
                               Mailbox Centers, Printing Services"
$ bestNightMonday    <lgl> FALSE, FALSE
$ bestNightTuesday   <lgl> FALSE, FALSE
$ bestNightWednesday <lgl> FALSE, FALSE

(2 of 150,344 rows & 79 of 94 columns not shown)
```

Data Visualizations






Yelp Businesses: Star Ratings Across the US & Southern Cana
Latitude
60
50
40
30
−120
−100
−80
−60
Longitude
Star Rating
5
4
3
2
1

Appendix

Below is all the code I used for cleaning and organizing the data. I removed any unnecessary libraries and listed the Python imports first, followed by the R libraries. Each section of the code handles a different part of the process.

Note: I used ChatGPT to help me format the code and handle repetitive tasks like renaming columns, fixing data types, and applying regular expressions. The only thing that I used generative AI for the report above was for the table creation. Additionally, working with this complex dataset would have taken me around five to six hours of manual effort if I hadn't used the AI. Hence, ChatGPT made the process much faster and easier, allowing me to focus more on analyzing the data instead of getting stuck on repetitive coding tasks. Just wanted to be transparent!

Python Imports

```
import os
import json
import csv
import pandas as pd
import ast
import re
```

R Libraries

```
library(readr)
library(dplyr)
library(stringr)
library(jsonlite)
library(purrr)
library(tidyr)
library(ggplot2)
library(maps)
```

Python Code: JSON to CSV Conversion

```
# Paths
input_dir = "yelp_dataset"
output_dir = "yelpDatasetsCSV"

# Create the output directory if it doesn't exist
os.makedirs(output_dir, exist_ok=True)

# Function to flatten nested fields
def flatten_record(record, parent_key='', sep='.'):
    items = []
    for key, value in record.items():
        new_key = f"{parent_key}{sep}{key}" if parent_key else key
        if isinstance(value, dict):
            items.extend(flatten_record(value, new_key,
    sep=sep).items())
        else:
            items.append((new_key, value))
    return dict(items)

# Function to dynamically adjust headers and write to CSV
def json_to_csv(json_file, csv_file):
    fieldnames = set()
    rows = []

    with open(json_file, 'r', encoding='utf-8') as infile:
        for line in infile:
            record = json.loads(line.strip())
            flat_record = flatten_record(record)
            fieldnames.update(flat_record.keys())
            rows.append(flat_record)

    with open(csv_file, 'w', encoding='utf-8', newline='') as outfile:
        writer = csv.DictWriter(outfile, fieldnames=sorted(fieldnames))
        writer.writeheader()
        writer.writerows(rows)

# JSON files to process
json_files = [
    "yelp_academic_dataset_business.json",
    "yelp_academic_dataset_review.json",
```

```
    "yelp_academic_dataset_user.json",
    "yelp_academic_dataset_checkin.json",
    "yelp_academic_dataset_tip.json",
]

# Convert each JSON file to a separate CSV
for json_filename in json_files:
    json_file = os.path.join(input_dir, json_filename)
    csv_file = os.path.join(output_dir, json_filename.replace(".json",
    ".csv"))

    if os.path.exists(json_file):
        print(f"Converting {json_filename} to {csv_file}...")
        json_to_csv(json_file, csv_file)
    else:
        print(f"File {json_filename} not found in the input directory.")
```

Python Code: Initial CSV Cleaning (Flattening JSON-Like Fields)

```
# File paths
input_csv = "yelpDatasetsCSV/yelp_academic_dataset_business.csv"
output_csv = "yelpDatasetsCSV/cleaned_business.csv"

# Load CSV into a DataFrame
df = pd.read_csv(input_csv)

# Remove 'attributes.' prefix from column names
df.columns = [col.replace("attributes.", "") for col in df.columns]

# Flatten JSON-like values in cells
def parse_json_columns(row, columns):
    for col in columns:
        if pd.notna(row[col]):
            try:
                parsed_data = ast.literal_eval(row[col])
                if isinstance(parsed_data, dict):
                    for key, value in parsed_data.items():
                        row[f"{col}_{key}"] = value
                row[col] = None  # Drop original JSON-like column
            except (ValueError, SyntaxError):
```

```
                pass
    return row

json_columns = ["Ambience", "BusinessParking", "GoodForMeal"]
df = df.apply(lambda row: parse_json_columns(row, json_columns), axis=1)

# Remove u' and extra quotes, standardize 'NA'/'None'
def clean_text(value):
    if pd.isna(value):
        return None
    if isinstance(value, str):
        value = re.sub(r"^u?'|'$", "", value.strip())
        if value.lower() in ["none", "na"]:
            return None
    return value

df = df.applymap(clean_text)

# Convert string booleans to actual booleans
def convert_booleans(value):
    if pd.isna(value):
        return None
    if isinstance(value, str):
        val = value.strip().lower()
        if val in ["true", "yes"]:
            return True
        elif val in ["false", "no"]:
            return False
    return value

df = df.applymap(convert_booleans)

# Save cleaned DataFrame to a new CSV file
df.to_csv(output_csv, index=False)
print(f"Cleaned data saved to: {output_csv}")
```

R Code: Additional Cleanup & Feature Extraction

```r
businessReviews <- read_csv("yelpDatasetsCSV/cleaned_business.csv")

# Drop columns that are fully NA
businessReviews <- businessReviews %>%
  select(where(~ !all(is.na(.))))

# Identify columns containing business hours
time_columns <- grep("^hours\\.", colnames(businessReviews), value =
    TRUE)

# Function to format time strings consistently
format_time_range <- function(time_ranges) {
  sapply(time_ranges, function(time_range) {
    if (is.na(time_range)) {
      return("Closed")
    }
    parts <- strsplit(time_range, "-")[[1]]
    if (length(parts) != 2) {
      return(NA)
    }
    # Format as HH:MM:SS
    start_split <- strsplit(parts[1], ":")[[1]]
    end_split <- strsplit(parts[2], ":")[[1]]
    formatted_start <- sprintf("%02d:%02d:00",
                               as.integer(start_split[1]),
                               as.integer(start_split[2]))
    formatted_end <- sprintf("%02d:%02d:00",
                             as.integer(end_split[1]),
                             as.integer(end_split[2]))
    paste(formatted_start, "-", formatted_end)
  })
}

# Clean and rename hour columns
businessReviews <- businessReviews %>%
  rename_with(~ gsub("^hours\\.", "", .x)) %>%
  rename_with(~ paste0(., "_Operating_Hours"),
   matches("Monday|Tuesday|Wednesday|Thursday|Friday|Saturday|Sunday"))
       %>%
  mutate(across(ends_with("_Operating_Hours"), format_time_range))
```

```r
# Clean up `BestNights` and create day columns
businessReviews <- businessReviews %>%
  mutate(
    BestNights = gsub("u'", "'", BestNights),
    BestNights = gsub('"', "'", BestNights),
    BestNight_Monday    = ifelse(grepl("'monday': true", BestNights,
        ignore.case = TRUE), TRUE, FALSE),
    BestNight_Tuesday   = ifelse(grepl("'tuesday': true", BestNights,
        ignore.case = TRUE), TRUE, FALSE),
    BestNight_Wednesday = ifelse(grepl("'wednesday': true", BestNights,
        ignore.case = TRUE), TRUE, FALSE),
    BestNight_Thursday  = ifelse(grepl("'thursday': true", BestNights,
        ignore.case = TRUE), TRUE, FALSE),
    BestNight_Friday    = ifelse(grepl("'friday': true", BestNights,
        ignore.case = TRUE), TRUE, FALSE),
    BestNight_Saturday  = ifelse(grepl("'saturday': true", BestNights,
        ignore.case = TRUE), TRUE, FALSE),
    BestNight_Sunday    = ifelse(grepl("'sunday': true", BestNights,
        ignore.case = TRUE), TRUE, FALSE)
  ) %>%
  select(-BestNights)

# Validate states for U.S. + Canada
valid_states <- c(

      "AL","AK","AZ","AR","CA","CO","CT","DE","FL","GA","HI","ID","IL","IN","IA",

      "KS","KY","LA","ME","MD","MA","MI","MN","MS","MO","MT","NE","NV","NH","NJ",

      "NM","NY","NC","ND","OH","OK","OR","PA","RI","SC","SD","TN","TX","UT","VT",
  "VA","WA","WV","WI","WY","DC",
  "AB","BC","MB","NB","NL","NS","NT","NU","ON","PE","QC","SK","YT"
)

businessReviews <- businessReviews %>%
  mutate(
    state = toupper(state),
    state = ifelse(state %in% valid_states, state, NA)
  ) %>%
  filter(!is.na(state))
```

```r
# Separate postal codes into U.S. vs. Canadian
businessReviews <- businessReviews %>%
  mutate(
    US_PostalCode = ifelse(grepl("^[0-9]+$", postal_code),
        as.numeric(postal_code), NA),
    Canada_PostalCode = ifelse(
      grepl("^[A-Za-z]\\d[A-Za-z]( \\d[A-Za-z]\\d)?$", postal_code),
      toupper(postal_code), NA
    )
  ) %>%
  select(-postal_code)
```

R Code: Clean JSON-Like Columns (Hair Specialties, Music)

```r
clean_json_column <- function(df, column_name, new_prefix) {
  df %>%
    mutate(
      !!column_name := str_replace_all(!!sym(column_name), "u'|'",
          "\""),
      !!column_name := str_replace_all(!!sym(column_name), "'", "\""),
      !!column_name := str_replace_all(!!sym(column_name), "\\bnone\\b",
          "null"),
      parsed = map(!!sym(column_name), ~ {
        json_str <- .x
        if (!is.na(json_str)) {
          result <- tryCatch(fromJSON(json_str), error = function(e)
    NULL)
          return(result)
        } else {
          return(NA)
        }
      })
    ) %>%
    unnest_wider(parsed, names_sep = "_") %>%
    rename_with(~ str_replace(., "parsed_", new_prefix)) %>%
    select(-!!sym(column_name))
}

businessReviews <- businessReviews %>%
  clean_json_column("HairSpecializesIn", "HairSpecializesIn") %>%
```

```
  clean_json_column("Music", "Music")

# Remove extraneous auto-generated columns
businessReviews <- businessReviews %>%
  select(-Music1, -HairSpecializesIn1)

# Rename columns and convert to lowercase
businessReviews <- businessReviews %>%
  rename_with(~ str_replace_all(.,
    c(
      "HairSpecializesIn" = "hair_specializes_in_",
      "Music" = "music_",
      "(?<=[a-z])([A-Z])" = "_\\1"
    )
  ) %>% str_to_lower())
```

R Code: Final Column Renaming & Export

```
ds <- businessReviews

# Dictionary for renaming
column_renames <- c(
  "accepts_insurance" = "acceptsInsurance",
  "ages_allowed" = "agesAllowed",
  "alcohol" = "alcoholType",
  "ambience_casual" = "hasCasualAmbience",
  "ambience_classy" = "hasClassyAmbience",
  "ambience_divey" = "hasDiveyAmbience",
  "ambience_hipster" = "hasHipsterAmbience",
  "ambience_intimate" = "hasIntimateAmbience",
  "ambience_romantic" = "hasRomanticAmbience",
  "ambience_touristy" = "hasTouristyAmbience",
  "ambience_trendy" = "hasTrendyAmbience",
  "ambience_upscale" = "hasUpscaleAmbience",
  "byob" = "allowsByob",
  "byobcorkage" = "byobCorkageFee",
  "bike_parking" = "hasBikeParking",
  "business_accepts_bitcoin" = "acceptsBitcoin",
  "business_accepts_credit_cards" = "acceptsCreditCards",
  "business_parking_garage" = "hasGarageParking",
```

```
"business_parking_lot" = "hasLotParking",
"business_parking_street" = "hasStreetParking",
"business_parking_valet" = "hasValetParking",
"business_parking_validated" = "hasValidatedParking",
"by_appointment_only" = "byAppointmentOnly",
"caters" = "offersCatering",
"coat_check" = "hasCoatCheck",
"corkage" = "hasCorkageFee",
"dogs_allowed" = "allowsDogs",
"drive_thru" = "hasDriveThru",
"good_for_dancing" = "goodForDancing",
"good_for_kids" = "goodForKids",
"good_for_meal_breakfast" = "goodForBreakfast",
"good_for_meal_brunch" = "goodForBrunch",
"good_for_meal_dessert" = "goodForDessert",
"good_for_meal_dinner" = "goodForDinner",
"good_for_meal_latenight" = "goodForLateNight",
"good_for_meal_lunch" = "goodForLunch",
"happy_hour" = "hasHappyHour",
"has_tv" = "hasTv",
"noise_level" = "noiseLevel",
"open24hours" = "open24Hours",
"outdoor_seating" = "hasOutdoorSeating",
"restaurants_attire" = "restaurantAttire",
"restaurants_counter_service" = "hasCounterService",
"restaurants_delivery" = "offersDelivery",
"restaurants_good_for_groups" = "goodForGroups",
"restaurants_price_range2" = "priceRange",
"restaurants_reservations" = "acceptsReservations",
"restaurants_table_service" = "hasTableService",
"restaurants_take_out" = "offersTakeout",
"smoking" = "smokingPolicy",
"wheelchair_accessible" = "isWheelchairAccessible",
"wi_fi" = "hasWifi",
"address" = "address",
"business_id" = "businessId",
"categories" = "categories",
"city" = "city",
"friday_operating_hours" = "fridayHours",
"monday_operating_hours" = "mondayHours",
"saturday_operating_hours" = "saturdayHours",
"sunday_operating_hours" = "sundayHours",
```

```
  "thursday_operating_hours" = "thursdayHours",
  "tuesday_operating_hours" = "tuesdayHours",
  "wednesday_operating_hours" = "wednesdayHours",
  "is_open" = "isOpen",
  "latitude" = "latitude",
  "longitude" = "longitude",
  "name" = "businessName",
  "review_count" = "reviewCount",
  "stars" = "starRating",
  "state" = "state",
  "best_night_monday" = "bestNightMonday",
  "best_night_tuesday" = "bestNightTuesday",
  "best_night_wednesday" = "bestNightWednesday",
  "best_night_thursday" = "bestNightThursday",
  "best_night_friday" = "bestNightFriday",
  "best_night_saturday" = "bestNightSaturday",
  "best_night_sunday" = "bestNightSunday",
  "us_postal_code" = "usPostalCode",
  "canada_postal_code" = "canadaPostalCode",
  "hair_specializes_in_straightperms" = "specializesInStraightPerms",
  "hair_specializes_in_coloring" = "specializesInColoring",
  "hair_specializes_in_extensions" = "specializesInExtensions",
  "hair_specializes_in_africanamerican" =
      "specializesInAfricanAmericanHair",
  "hair_specializes_in_curly" = "specializesInCurlyHair",
  "hair_specializes_in_kids" = "specializesInKidsHair",
  "hair_specializes_in_perms" = "specializesInPerms",
  "hair_specializes_in_asian" = "specializesInAsianHair",
  "music_dj" = "hasDj",
  "music_background_music" = "hasBackgroundMusic",
  "music_no_music" = "hasNoMusic",
  "music_jukebox" = "hasJukebox",
  "music_live" = "hasLiveMusic",
  "music_video" = "hasVideoMusic",
  "music_karaoke" = "hasKaraoke"
)

ds <- ds %>%
  rename_with(
    ~ column_renames[.x],
    .cols = names(column_renames)
  )
```

```
# Write final CSV
write_csv(ds, "yelpDatasetsCSV/yelpBusinessData.csv")
```

R Code: Final Glimpse

```
finalDs <- read_csv("yelpDatasetsCSV/yelpBusinessData.csv")
glimpse(finalDs)
```

R Code: Converting Final Dataset Back to JSON

```
write_dataset_to_json <- function(dataset, path) {
  # Convert to JSON with pretty formatting
  json_data <- toJSON(dataset, pretty = TRUE, na = "null")
  write(json_data, file = path)
}

write_dataset_to_json(finalDs, "yelpDatasetsCSV/yelpBusinessData.json")
```

```
# Verify that we can read json file
df = pd.read_json('yelpDatasetsCSV/yelpBusinessData.json')
# List of columns you want to view
columns_to_view = ['businessName', 'city', 'state', 'starRating',
    'reviewCount']

# Display only the selected columns
print(df[columns_to_view])
```

R Code: Data Visualizations

```
library(ggplot2)
library(maps)
library(readr)
library(dplyr)

finalDs <- read_csv("yelpDatasetsCSV/yelpBusinessData.csv")
```

```r
ggplot(finalDs, aes(x = starRating)) +
  geom_histogram(binwidth = 0.5, fill = "steelblue1", color = "white",
      boundary = 0) +
  theme_minimal() +
  labs(
    title = "Distribution of Star Ratings",
    x = "Star Rating",
    y = "Frequency"
  )
```

```r
ggplot(finalDs, aes(x = reviewCount)) +
  geom_histogram(binwidth = 0.25, fill = "lightgreen", color = "white")
      +
  scale_x_log10() +
  theme_minimal() +
  labs(
    title = "Distribution of Review Counts",
    subtitle = "Log-scaled",
    x = "Review Count",
    y = "Frequency"
  )
```

```r
world_map <- map_data("world")
north_america_map <- world_map %>%
  filter(
    region %in% c("USA", "Canada"),
    lat > 25,
    lat < 60
  )
ggplot() +
  geom_polygon(
    data = north_america_map,
    aes(x = long, y = lat, group = group),
    fill = "white",
    color = "black"
  ) +
  geom_point(
    data = finalDs,
    aes(x = longitude, y = latitude, color = starRating),
    alpha = 0.6
  ) +
```

```
  scale_color_gradient(low = "red", high = "green", na.value = "gray50")
      +
  theme_minimal() +
  labs(
    title = "Yelp Businesses: Star Ratings Across the US & Southern
        Canada",
    x = "Longitude",
    y = "Latitude",
    color = "Star Rating"
  ) +
  coord_fixed(1.3, xlim = c(-130, -60), ylim = c(25, 60))  # Show all US
      and southern Canada
```