## **Big Data in Media Technology**

**Lab 2: Use Generative AI To Support Advanced Data Analytics**



**Group 3**

**Name：Yangle Ma**

**Group Members: JingWen Qin YuTing Huang**

**Task**

1. **Exploratory Data Analysis (EDA)**
   1. **Correlation between numerical features in the dataset**

**Prompt:**

1. **I have a wine dataset, the features are Name Variety Origin Attr\_1 Attr\_2 Alcohol\_vol Alcohol\_percentage Winemaker\_notes Review Avg\_rating N\_ratings Price\_Out-of-stock Price 13 features all. I wanna visualize Correlation between numerical features in the dataset**
2. **could not convert string to float: '1,199.99'**
3. **AttributeError: Can only use .str accessor with string values!**
4. **ValueError: could not convert string to float: ''**

import seaborn as sns

import matplotlib.pyplot as plt

import numpy as np

# Ensure the 'Price' and 'Price (Out-of-stock)' columns are strings

df['Price'] = df['Price'].astype(str).str.replace(',', '').astype(float)

df['Price\_Out-of-stock'] = df['Price\_Out-of-stock'].replace('', np.nan).astype(str).str.replace(',', '').astype(float)

# Select numerical features

numerical\_features = df[['Alcohol\_vol', 'Alcohol\_percentage', 'Avg\_rating', 'N\_ratings', 'Price', 'Price\_Out-of-stock']]

# Calculate the correlation matrix

correlation\_matrix = numerical\_features.corr()

# Set up the matplotlib figure

plt.figure(figsize=(10, 8))

# Create a heatmap

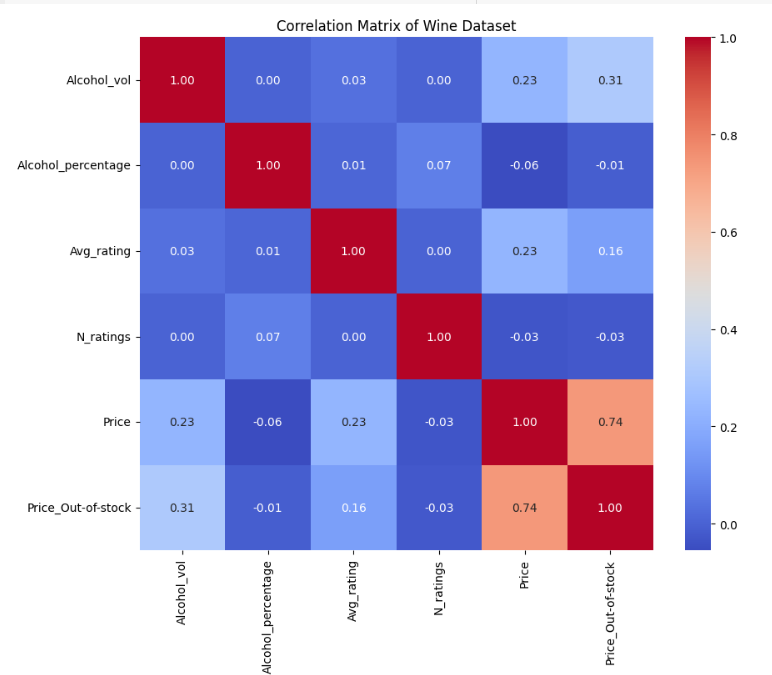
sns.heatmap(correlation\_matrix, annot=True, fmt=".2f", cmap='coolwarm', square=True)

# Add title

plt.title('Correlation Matrix of Wine Dataset')

# Show the plot

plt.show()

****

* 1. **Top ten countries with the most wines in the dataset**

**Prompt:**

1. **Visualize Top ten countries with the most wines in the dataset**

# Count wines by country

country\_counts = df['Origin'].value\_counts()

# Select the top 10 countries

top\_countries = country\_counts.head(10)

# Create a bar plot

plt.figure(figsize=(12, 6))

sns.barplot(x=top\_countries.index, y=top\_countries.values, palette='viridis')

# Add labels and title

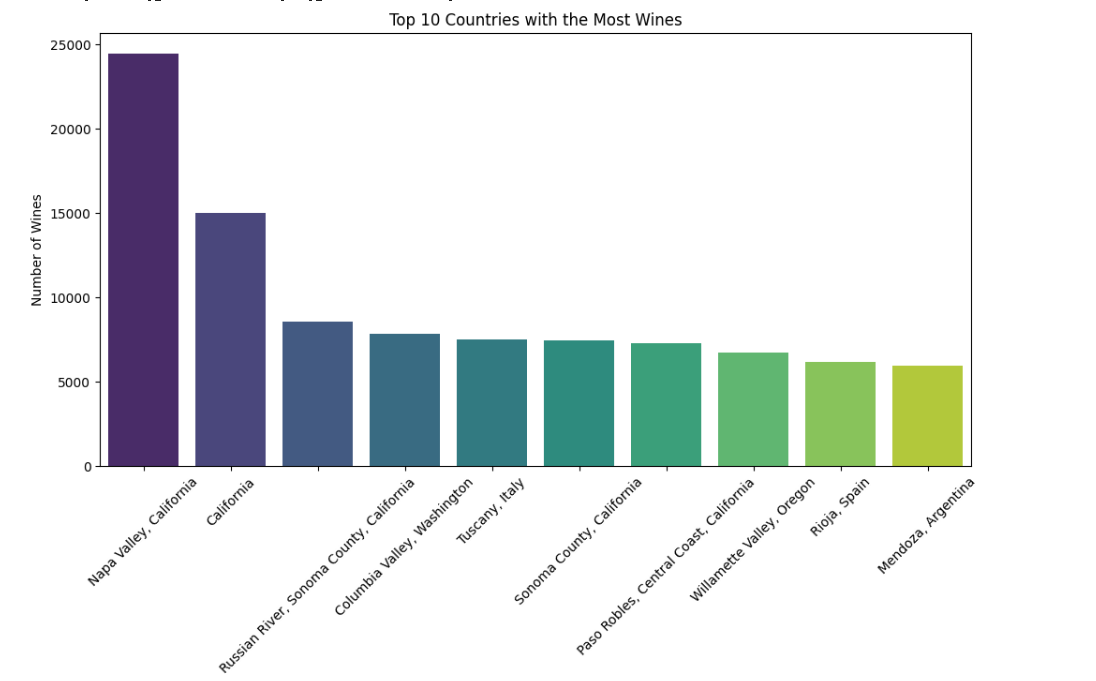
plt.xlabel('Country')

plt.ylabel('Number of Wines')

plt.title('Top 10 Countries with the Most Wines')

plt.xticks(rotation=45)

plt.show()

****

* 1. **Top ten grape varieties with the most entries in the dataset**

**Prompt:**

1. **Top ten grape varieties with the most entries in the dataset**

# Count entries by grape variety

variety\_counts = df['Variety'].value\_counts()

# Select the top 10 grape varieties

top\_varieties = variety\_counts.head(10)

# Create a bar plot

plt.figure(figsize=(12, 6))

sns.barplot(x=top\_varieties.index, y=top\_varieties.values, palette='magma')

# Add labels and title

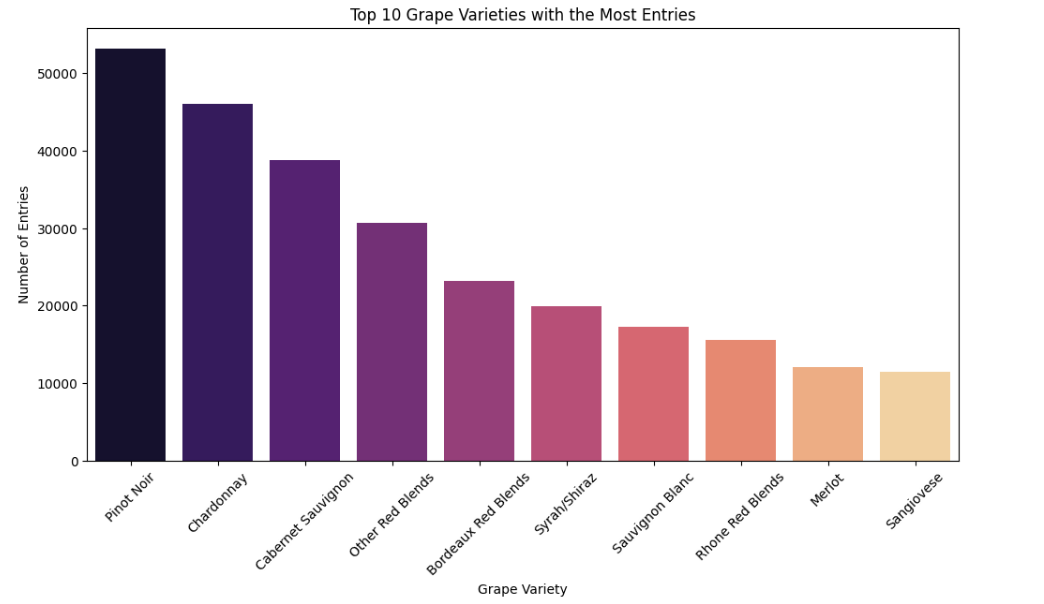
plt.xlabel('Grape Variety')

plt.ylabel('Number of Entries')

plt.title('Top 10 Grape Varieties with the Most Entries')

plt.xticks(rotation=45)

plt.show()

****

* 1. **Distribution of unique values among user review scores**

**Prompt:**

1. **Visualize Distribution of unique values among user review scores**

plt.figure(figsize=(12, 6))

# Using a count plot for unique review scores

sns.countplot(x='Avg\_rating', data=df, palette='coolwarm')

# Add labels and title

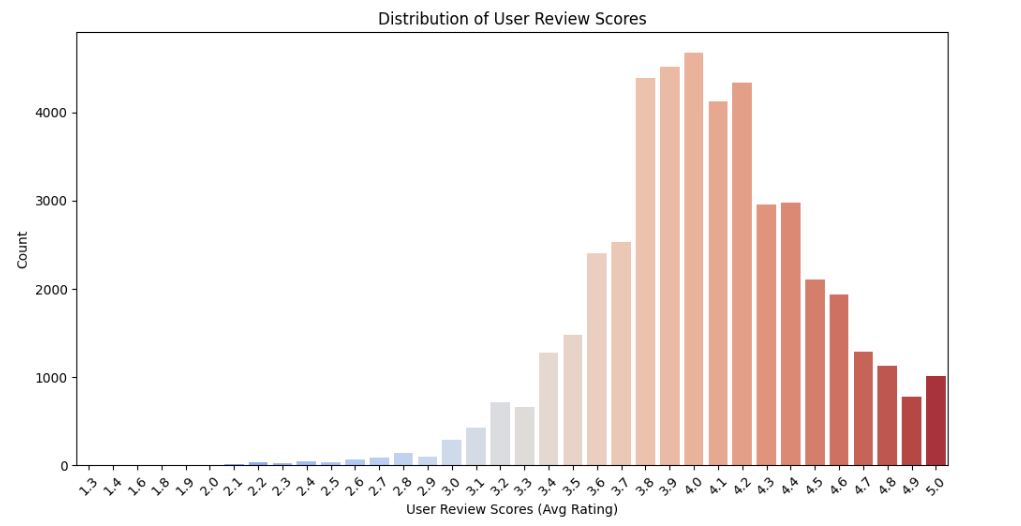
plt.xlabel('User Review Scores (Avg Rating)')

plt.ylabel('Count')

plt.title('Distribution of User Review Scores')

plt.xticks(rotation=45)

plt.show()

****

* 1. **Average user rating from the top ten countries**

**Prompt:**

1. **Visualize Average user rating from the top ten countries**

# Calculate average rating by country

average\_rating = df.groupby('Origin')['Avg\_rating'].mean()

# Sort and select the top 10 countries by average rating

top\_countries = average\_rating.sort\_values(ascending=False).head(10)

# Create a bar plot

plt.figure(figsize=(12, 6))

sns.barplot(x=top\_countries.index, y=top\_countries.values, palette='viridis')

# Add labels and title

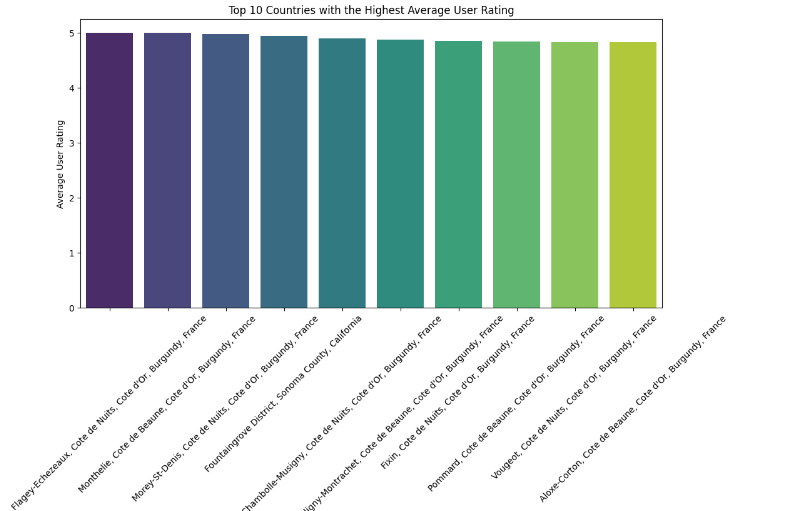
plt.xlabel('Country')

plt.ylabel('Average User Rating')

plt.title('Top 10 Countries with the Highest Average User Rating')

plt.xticks(rotation=45)

plt.show()

****

**It can be seen that we could’t get expected results without cleaning the raw data.**

1. **Data Cleaning and Feature Engineering**

**2.1 Remove Null Values**

**Prompt：**

1. **I have a wine dataset, the features are Name Variety Origin Attr\_1 Attr\_2 Alcohol\_vol Alcohol\_percentage Winemaker\_notes Review Avg\_rating N\_ratings Price\_Out-of-stock Price 13 features all.**

**checking for any entries that are missing a winemaker note or have null values for the wine’s provenance. discard any records that do not contain a review.**

# Step 1: Remove records with missing or null values in 'Winemaker\_notes' or 'Origin'

df = df.dropna(subset=['Winemaker\_notes', 'Origin'])

# Step 2: Discard any records that do not contain a review

df = df.dropna(subset=['Review'])

# Optional: Reset the index for the cleaned dataframe

df = df.reset\_index(drop=True)

**2.2 Remove and Build new feature**

**Prompt：**

1. **remove any non-numeric characters (in this case, "the") from the "Price-Out-of-Stock" and "Price" columns.**

df['Price\_Out-of-stock'] = df['Price\_Out-of-stock'].astype(str).str.replace(r'[^0-9.]', '', regex=True)

df['Price'] = df['Price'].astype(str).str.replace(r'[^0-9.]', '', regex=True)

# Step 4: Convert cleaned columns to numeric data types (if necessary)

df['Price\_Out-of-stock'] = pd.to\_numeric(df['Price\_Out-of-stock'], errors='coerce')

df['Price'] = pd.to\_numeric(df['Price'], errors='coerce')

1. **Convert the values to float types, then combine them into a single feature named "Price\_Feature.**

df['Price\_Feature'] = df['Price'].combine\_first(df['Price\_Out-of-stock'])

1. **‘Origin’ entries are formatted inconsistently, some only list the country,some list the state. For example "Rhone, France' Split any entries containing multiple locations in the 'Origin' column into separate columns, labeling each appropriately as 'Country' or 'Region'. For wines originating from the United States that only list the state, assign "United States" as the country of origin.**

# Step 6: Split 'Origin' into 'Country' and 'Region'

us\_states = [

    'Alabama', 'Alaska', 'Arizona', 'Arkansas', 'California', 'Colorado', 'Connecticut',

    'Delaware', 'Florida', 'Georgia', 'Hawaii', 'Idaho', 'Illinois', 'Indiana', 'Iowa',

    'Kansas', 'Kentucky', 'Louisiana', 'Maine', 'Maryland', 'Massachusetts', 'Michigan',

    'Minnesota', 'Mississippi', 'Missouri', 'Montana', 'Nebraska', 'Nevada', 'New Hampshire',

    'New Jersey', 'New Mexico', 'New York', 'North Carolina', 'North Dakota', 'Ohio',

    'Oklahoma', 'Oregon', 'Pennsylvania', 'Rhode Island', 'South Carolina', 'South Dakota',

    'Tennessee', 'Texas', 'Utah', 'Vermont', 'Virginia', 'Washington', 'West Virginia',

    'Wisconsin', 'Wyoming'

]

def split\_origin(origin):

    parts = [part.strip() for part in origin.split(',')]

    if len(parts) == 1:

        return pd.Series([parts[0], None])

    elif len(parts) >= 2:

        country = parts[len(parts)-1]

        region = parts[len(parts)-2]

        zone = parts[len(parts)-3] if len(parts) >= 3 else None

        # Assign "United States" if the region is a known US state

        if country in us\_states:

            country = "United States"

        return pd.Series([country, region, zone])

return pd.Series([None, None])  # Handle any unexpected cases

# Apply the split\_origin function to the 'Origin' column

df[['Country', 'Region','Zone']] = df['Origin'].apply(split\_origin)

**2.3 Extract Feature**

**Prompt:**

1. **Extract the year from the wine’s name and create a new feature, "Year," to store this information. Do not retain decimals**

import re

def extract\_year(name):

    if pd.isna(name):

        return None

    # Find a four-digit number (year) in the name

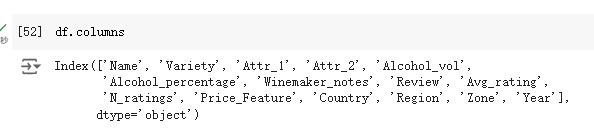
    match = re.search(r'\b(20\d{2}|19\d{2})\b', name)

    return int(match.group(0)) if match else None

# Apply the extract\_year function to the 'Name' column

df['Year'] = df['Name'].apply(extract\_year)

**For the Final Results , we can see we satisy the requirements**

****

**2.4 Create 6 Histograms**

**Prompt:**

1. **Create 6 histograms to display the distribution of each numerical feature.**

# List of numerical features to plot

numerical\_features = [

    'Alcohol\_vol',

    'Alcohol\_percentage',

    'Price\_Feature',

    'Year',

    'Avg\_rating',

    'N\_ratings'  # Ensure to include all relevant numerical features

]

# Set up the plotting area

plt.figure(figsize=(15, 15))

# Create histograms for each numerical feature

for i, feature in enumerate(numerical\_features):

    plt.subplot(3, 2, i + 1)  # Adjust the grid size as needed

    plt.hist(df[feature].dropna(), bins=30, color='skyblue', edgecolor='black')

    plt.title(f'Distribution of {feature}')

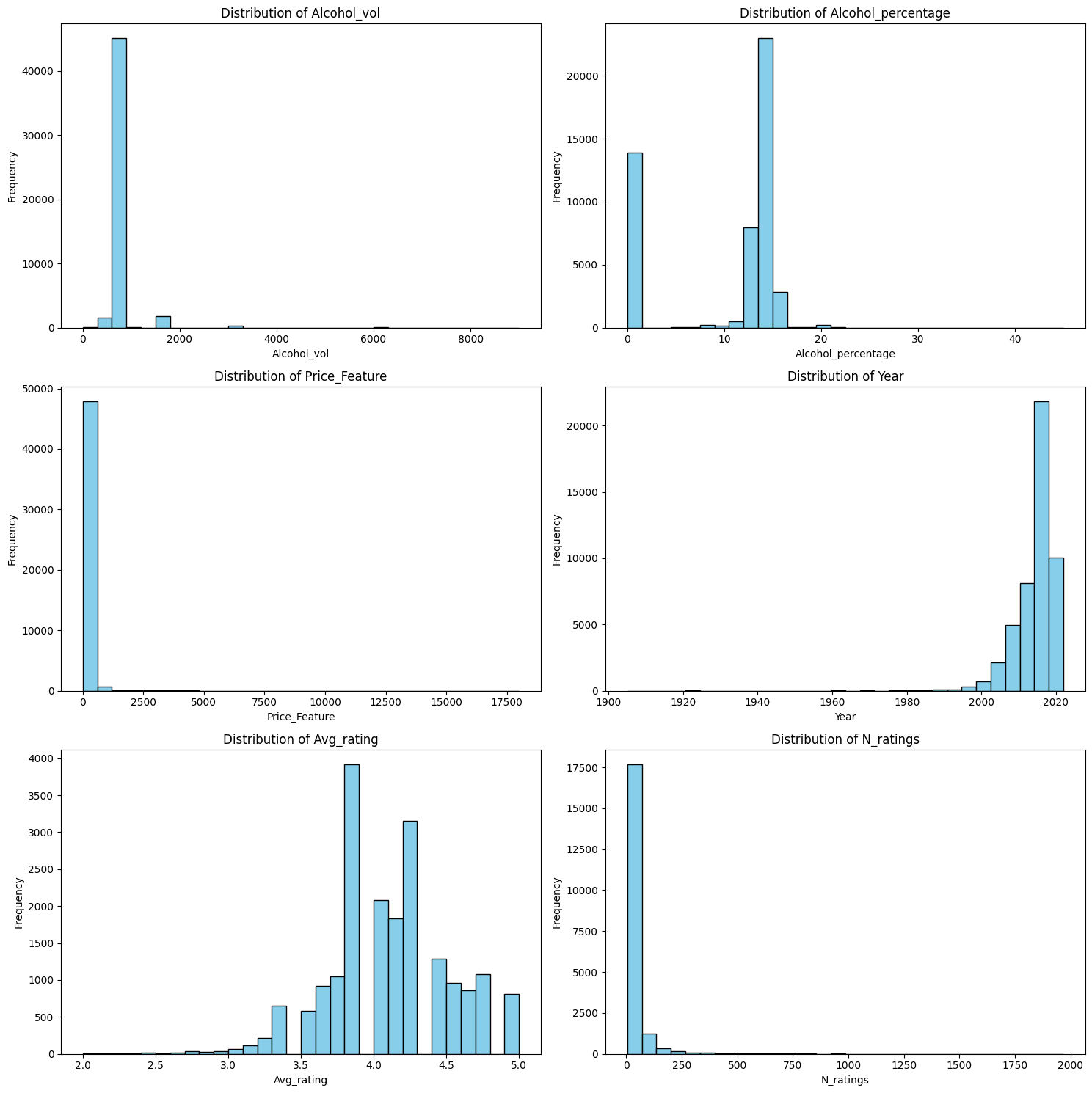
    plt.xlabel(feature)

    plt.ylabel('Frequency')

# Adjust layout for better spacing

plt.tight\_layout()

plt.show()



**2.5 Re-draw Figures 2, 3, and 5**

**Prompt:**

1. **Visualize Top ten countries with the most wines in the dataset**

# Step 1: Count the number of wines for each country

country\_counts = df['Country'].value\_counts()

# Step 2: Select the top ten countries

top\_ten\_countries = country\_counts.head(10)

# Step 3: Plot the bar chart

plt.figure(figsize=(10, 6))

top\_ten\_countries.plot(kind='bar', color='lightblue', edgecolor='black')

# Step 4: Add titles and labels

plt.title('Top Ten Countries with the Most Wines')

plt.xlabel('Country')

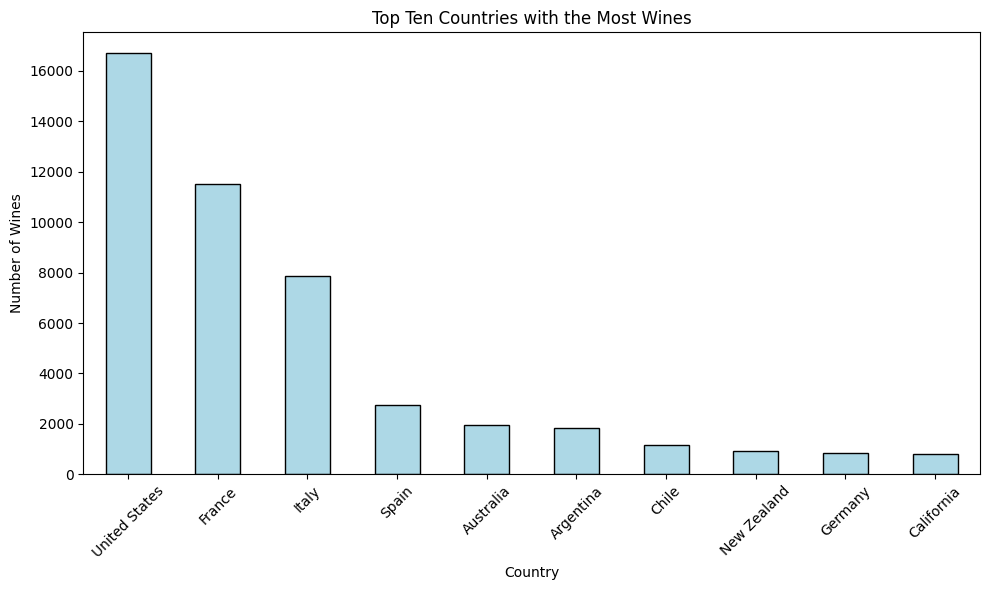
plt.ylabel('Number of Wines')

plt.xticks(rotation=45)

# Show the plot

plt.tight\_layout()

plt.show()



1. **Visualize Top ten countries with the most wines in the dataset**

# Step 1: Count the number of entries for each grape variety

variety\_counts = df['Variety'].value\_counts()

# Step 2: Select the top ten grape varieties

top\_ten\_varieties = variety\_counts.head(10)

# Step 3: Plot the bar chart

plt.figure(figsize=(10, 6))

top\_ten\_varieties.plot(kind='bar', color='lightcoral', edgecolor='black')

# Step 4: Add titles and labels

plt.title('Top Ten Grape Varieties with the Most Entries')

plt.xlabel('Grape Variety')

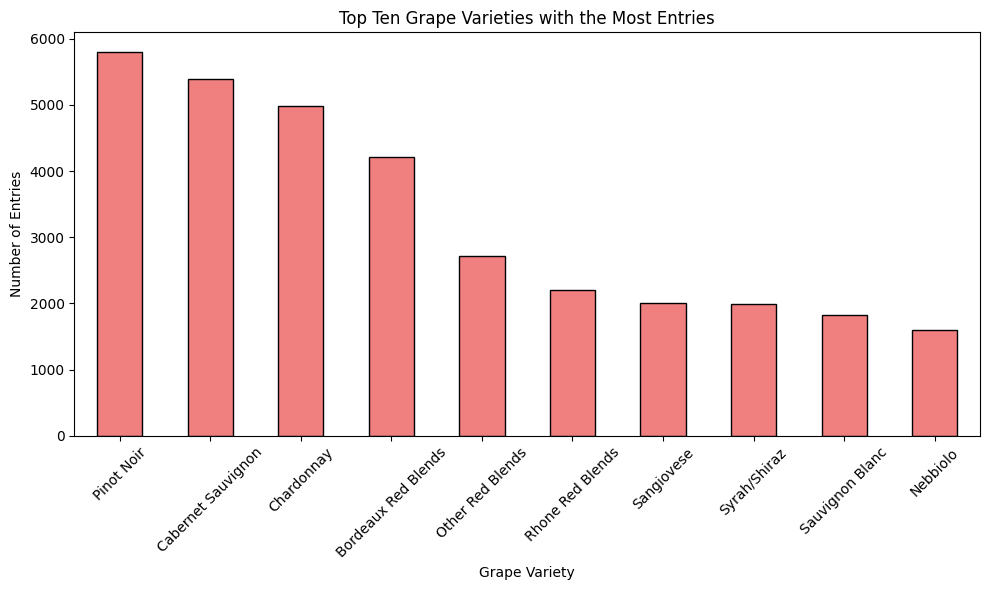
plt.ylabel('Number of Entries')

plt.xticks(rotation=45)

# Show the plot

plt.tight\_layout()

plt.show()



1. **Visualize Average user rating from the top ten countries**

# Step 1: Count the number of wines for each country

country\_counts = df['Country'].value\_counts()

# Step 2: Get the top ten countries

top\_ten\_countries = country\_counts.head(10).index

# Step 3: Calculate the average rating for these countries

average\_ratings = df[df['Country'].isin(top\_ten\_countries)].groupby('Country')['Avg\_rating'].mean()

# Step 4: Plot the bar chart

plt.figure(figsize=(10, 6))

average\_ratings.plot(kind='bar', color='skyblue', edgecolor='black')

# Step 5: Add titles and labels

plt.title('Average User Rating from Top Ten Countries')

plt.xlabel('Country')

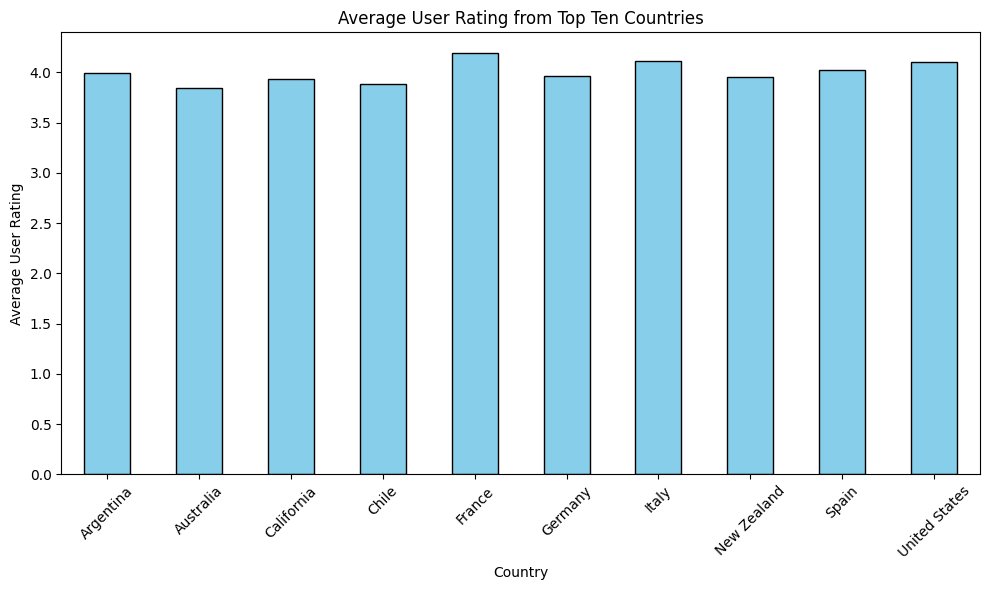
plt.ylabel('Average User Rating')

plt.xticks(rotation=45)

# Show the plot

plt.tight\_layout()

plt.show()



**You can compare these three graphs with the graphs created by the raw data. We can now see the differences about the average user ratings and the most wines.**

**Code**