Coffee_Shop_Analysis_and_Predictions_Report

May 12, 2025

Coffee Shop Analysis and Predictions

This project explores and analyzes a dataset of over 5,000 specialty coffee reviews to uncover trends in flavor, roast types, regions, and ratings. The data was cleaned, merged, and transformed to ensure consistency and usability. Feature engineering techniques were applied to convert raw text and categorical values into meaningful inputs for analysis. Machine learning models were used to predict coffee ratings based on sensory attributes. The goal was to extract insights and build a predictive model.

[1]:
```
# Import all the labriaries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from category_encoders import TargetEncoder
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score


import warnings
warnings.filterwarnings("ignore")
```

[2]:
```
# Load the data

coffee_raw = pd.read_csv('coffee.csv')
coffee_clean = pd.read_csv('coffee_clean.csv')
coffee_id = pd.read_csv('coffee_id.csv')
```

1 Data Exploration and Quality Check

1.1 Review the Dataset Structure

[3]:
```python
# Data Overview

print('\n', '='*20, 'Data Overview of coffee_raw', '='*20, '\n')
print(coffee_raw.head())

print('\n', '='*20, 'Data Overview of coffee_clean', '='*20, '\n')
print(coffee_clean.head())

print('\n', '='*20, 'Data Overview of coffee_id', '='*20, '\n')
print(coffee_id.head())
```

```
 ==================== Data Overview of coffee_raw ====================

                                            all_text  \
0  \n\n\n\n \n93\nFlight Coffee Co.\nEthiopia Der…
1  \n\n\n\n\n91\nDoi Chaang Coffee\nEspresso\nLoc…
2  \n\n\n\n \n95\nTemple Coffee and Tea\nKenya Ru…
3  \n\n\n\n \n93\nTemple Coffee and Tea\nEthiopia…
4  \n\n\n\n\n93\nChoosy Gourmet\nSpecialty Coffee…

                              name rating                roaster  \
0            Ethiopia Deri Kochoha     93       Flight Coffee Co.
1                         Espresso     91       Doi Chaang Coffee
2           Kenya Ruthaka Peaberry     95   Temple Coffee and Tea
3       Ethiopia Gora Kone Sidamo      93   Temple Coffee and Tea
4  Specialty Coffee Blend Espresso     93          Choosy Gourmet

                                      slug  region_africa_arabia  \
0         /review/ethiopia-deri-kochoha-2                      1
1                     /review/espresso-14                      0
2          /review/kenya-ruthaka-peaberry                      1
3       /review/ethiopia-gora-kone-sidamo                      1
4  /review/specialty-coffee-blend-espresso                     0

   region_caribbean  region_central_america  region_hawaii  \
0                 0                       0              0
1                 0                       0              0
2                 0                       0              0
3                 0                       0              0
4                 0                       0              0

   region_asia_pacific  …  aroma  acid  body  flavor  aftertaste  with_milk  \
0                    0  …    9.0   8.0   9.0     9.0         8.0        NaN
```

```
1                     1  …    8.0   NaN   8.0    8.0         8.0        9.0
2                     0  …    9.0   8.0   9.0   10.0         8.0        NaN
3                     0  …    9.0   8.0   9.0    9.0         8.0        NaN
4                     0  …    9.0   NaN   8.0    9.0         8.0        9.0

                                              desc_1  \
0  Bright, crisp, sweetly tart. Citrus medley, ca…
1  Evaluated as espresso. Deeply rich, sweetly ro…
2  Deeply sweet, richly savory. Dark chocolate, p…
3  Fruit-forward, richly chocolaty. Raspberry cou…
4  Evaluated as espresso. Rich, chocolaty, sweetl…

                                              desc_2  \
0  From the Deri Kochoha mill in the Hagere Marya…
1  Doi Chaang is a single-estate coffee produced …
2  Despite challenges ranging from contested gove…
3  Southern Ethiopia coffees like this one are la…
4  A blend of coffees from Ethiopia (natural-proc…

                                              desc_3 desc_4
0  A poised and melodic wet-processed Ethiopia co…    NaN
1  A rich, resonant espresso from Thailand, espec…    NaN
2  A high-toned, nuanced Kenya cup, classic in it…    NaN
3  A playful, unrestrained fruit bomb of a coffee…    NaN
4  An espresso blend in which spice notes - in pa…    NaN

[5 rows x 34 columns]

 ==================== Data Overview of coffee_clean ====================

                               slug  aroma  acid_or_milk  body  flavor  \
0          ethiopia-deri-kochoha-2    9.0           8.0   9.0     9.0
1                      espresso-14    8.0           9.0   8.0     8.0
2           kenya-ruthaka-peaberry    9.0           8.0   9.0    10.0
3         ethiopia-gora-kone-sidamo    9.0           8.0   9.0     9.0
4  specialty-coffee-blend-espresso    9.0           9.0   8.0     9.0

   type_with_milk                                         clean_text  \
0               0  bright crisp sweetli tart citru medley cacao n…
1               1  evalu espresso deepli rich sweetli roast round…
2               0  deepli sweet richli savori dark chocol pistach…
3               0  fruit forward richli chocolati raspberri couli…
4               1  evalu espresso rich chocolati sweetli tart dar…

   roast_dark  roast_light  roast_medium  …  region_asia_pacific  \
0           0            0             0  …                    0
1           0            0             1  …                    1
2           0            0             1  …                    0
```

```
3            0             0             0  ...                  0
4            0             0             0  ...                  0

   region_south_america  type_espresso  type_organic  type_fair_trade  \
0                     0              0             0                0
1                     0              1             0                0
2                     0              0             0                0
3                     0              0             0                0
4                     0              1             0                0

   type_decaffeinated  type_pod_capsule  type_blend  type_estate  \
0                   0                 0           0            0
1                   0                 0           0            1
2                   0                 0           0            0
3                   0                 0           0            0
4                   0                 0           0            0

   type_with_milk.1
0                 0
1                 1
2                 0
3                 0
4                 1

[5 rows x 28 columns]

 ==================== Data Overview of coffee_id ====================

                              slug                            name  \
0          ethiopia-deri-kochoha-2           Ethiopia Deri Kochoha
1                      espresso-14                        Espresso
2           kenya-ruthaka-peaberry          Kenya Ruthaka Peaberry
3        ethiopia-gora-kone-sidamo       Ethiopia Gora Kone Sidamo
4  specialty-coffee-blend-espresso  Specialty Coffee Blend Espresso

               roaster rating review_date
0      Flight Coffee Co.     93  2019-01-01
1      Doi Chaang Coffee     91  2019-01-01
2  Temple Coffee and Tea     95  2019-01-01
3  Temple Coffee and Tea     93  2019-01-01
4         Choosy Gourmet     93  2019-01-01
```

[4]:

```python
# Data Information

print('\n', '='*10, 'Data Information of coffee_raw', '='*10, '\n')
coffee_raw.info()
print('\n', '='*10, 'Data Information of coffee_clean', '='*10, '\n')
```

```python
coffee_clean.info()
print('\n', '='*10, 'Data Information of coffee_id', '='*10, '\n')
coffee_id.info()
```

```
 ========== Data Information of coffee_raw ==========

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5124 entries, 0 to 5123
Data columns (total 34 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   all_text               5124 non-null   object
 1   name                   5124 non-null   object
 2   rating                 5124 non-null   object
 3   roaster                5124 non-null   object
 4   slug                   5124 non-null   object
 5   region_africa_arabia   5124 non-null   int64
 6   region_caribbean       5124 non-null   int64
 7   region_central_america 5124 non-null   int64
 8   region_hawaii          5124 non-null   int64
 9   region_asia_pacific    5124 non-null   int64
 10  region_south_america   5124 non-null   int64
 11  type_espresso          5124 non-null   int64
 12  type_organic           5124 non-null   int64
 13  type_fair_trade        5124 non-null   int64
 14  type_decaffeinated     5124 non-null   int64
 15  type_pod_capsule       5124 non-null   int64
 16  type_blend             5124 non-null   int64
 17  type_estate            5124 non-null   int64
 18  location               5122 non-null   object
 19  origin                 4529 non-null   object
 20  roast                  4696 non-null   object
 21  est_price              3014 non-null   object
 22  review_date            5124 non-null   object
 23  agtron                 5124 non-null   object
 24  aroma                  5085 non-null   float64
 25  acid                   4256 non-null   float64
 26  body                   5111 non-null   float64
 27  flavor                 5106 non-null   float64
 28  aftertaste             4111 non-null   float64
 29  with_milk              700 non-null    float64
 30  desc_1                 5124 non-null   object
 31  desc_2                 5124 non-null   object
 32  desc_3                 971 non-null    object
 33  desc_4                 4153 non-null   object
dtypes: float64(6), int64(13), object(15)
memory usage: 1.3+ MB
```

```
 ========== Data Information of coffee_clean ==========

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4887 entries, 0 to 4886
Data columns (total 28 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   slug                   4887 non-null   object
 1   aroma                  4887 non-null   float64
 2   acid_or_milk           4887 non-null   float64
 3   body                   4887 non-null   float64
 4   flavor                 4887 non-null   float64
 5   type_with_milk         4887 non-null   int64
 6   clean_text             4887 non-null   object
 7   roast_dark             4887 non-null   int64
 8   roast_light            4887 non-null   int64
 9   roast_medium           4887 non-null   int64
 10  roast_medium_dark      4887 non-null   int64
 11  roast_medium_light     4887 non-null   int64
 12  roast_very_dark        4887 non-null   int64
 13  roast_nan              4887 non-null   int64
 14  region_africa_arabia   4887 non-null   int64
 15  region_caribbean       4887 non-null   int64
 16  region_central_america  4887 non-null   int64
 17  region_hawaii          4887 non-null   int64
 18  region_asia_pacific    4887 non-null   int64
 19  region_south_america   4887 non-null   int64
 20  type_espresso          4887 non-null   int64
 21  type_organic           4887 non-null   int64
 22  type_fair_trade        4887 non-null   int64
 23  type_decaffeinated     4887 non-null   int64
 24  type_pod_capsule       4887 non-null   int64
 25  type_blend             4887 non-null   int64
 26  type_estate            4887 non-null   int64
 27  type_with_milk.1       4887 non-null   int64
dtypes: float64(4), int64(22), object(2)
memory usage: 1.0+ MB

 ========== Data Information of coffee_id ==========

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4887 entries, 0 to 4886
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   slug         4887 non-null   object
 1   name         4887 non-null   object
```

```
 2   roaster      4887 non-null   object
 3   rating       4887 non-null   object
 4   review_date  4887 non-null   object
dtypes: object(5)
memory usage: 191.0+ KB
```

[5]:

```python
# Statistical summary

print('\n', '='*20, 'Statistical Summary', '='*20, '\n')

print('\n', '='*10, 'Statistical Summary of coffee_raw', '='*10, '\n')
print(coffee_raw.describe(include='all'))

print('\n', '='*10, 'Statistical Summary of coffee_clean', '='*10, '\n')
print(coffee_clean.describe(include='all'))

print('\n', '='*10, 'Statistical Summary of coffee_id', '='*10, '\n')
print(coffee_id.describe(include='all'))
```

```
 ==================== Statistical Summary ====================


 ========== Statistical Summary of coffee_raw ==========

                                                  all_text           name  \
count                                                 5124           5124
unique                                                5124           4232
top     \n\n\n\n \n93\nFlight Coffee Co.\nEthiopia Der…  Holiday Blend
freq                                                     1             25
mean                                                   NaN            NaN
std                                                    NaN            NaN
min                                                    NaN            NaN
25%                                                    NaN            NaN
50%                                                    NaN            NaN
75%                                                    NaN            NaN
max                                                    NaN            NaN

       rating             roaster                             slug  \
count    5124                5124                             5124
unique     37                1175                             5124
top        93  JBC Coffee Roasters  /review/ethiopia-deri-kochoha-2
freq      789                 179                                1
mean      NaN                 NaN                              NaN
std       NaN                 NaN                              NaN
min       NaN                 NaN                              NaN
25%       NaN                 NaN                              NaN
50%       NaN                 NaN                              NaN
```

```
75%       NaN                 NaN                                NaN
max       NaN                 NaN                                NaN

        region_africa_arabia  region_caribbean  region_central_america  \
count            5124.000000       5124.000000             5124.000000
unique                   NaN               NaN                     NaN
top                      NaN               NaN                     NaN
freq                     NaN               NaN                     NaN
mean                0.217213          0.008197                0.159251
std                 0.412389          0.090173                0.365945
min                 0.000000          0.000000                0.000000
25%                 0.000000          0.000000                0.000000
50%                 0.000000          0.000000                0.000000
75%                 0.000000          0.000000                0.000000
max                 1.000000          1.000000                1.000000

        region_hawaii  region_asia_pacific  …        aroma         acid  \
count     5124.000000          5124.000000  …  5085.000000  4256.000000
unique            NaN                  NaN  …          NaN          NaN
top               NaN                  NaN  …          NaN          NaN
freq              NaN                  NaN  …          NaN          NaN
mean         0.020492             0.076308  …     8.141357     7.726974
std          0.141689             0.265516  …     1.016351     0.974802
min          0.000000             0.000000  …     2.000000     2.000000
25%          0.000000             0.000000  …     8.000000     7.000000
50%          0.000000             0.000000  …     8.000000     8.000000
75%          0.000000             0.000000  …     9.000000     8.000000
max          1.000000             1.000000  …    10.000000    10.000000

               body       flavor   aftertaste   with_milk  \
count   5111.000000  5106.000000  4111.000000  700.000000
unique          NaN          NaN          NaN         NaN
top             NaN          NaN          NaN         NaN
freq            NaN          NaN          NaN         NaN
mean       7.862004     8.213670     7.910727    8.301429
std        0.899891     1.115329     0.787406    0.759676
min        4.000000     1.000000     2.000000    5.000000
25%        7.000000     8.000000     8.000000    8.000000
50%        8.000000     8.000000     8.000000    8.000000
75%        8.000000     9.000000     8.000000    9.000000
max       10.000000    10.000000    10.000000   10.000000

                                                  desc_1  \
count                                               5124
unique                                              5118
top     This Kenya attracted the highest rating achiev…
freq                                                   2
mean                                                 NaN
```

```
std                                                         NaN
min                                                         NaN
25%                                                         NaN
50%                                                         NaN
75%                                                         NaN
max                                                         NaN

                                                        desc_2 desc_3 desc_4
count                                                     5124    971   4153
unique                                                    4904    970   4023
top     Paradise Roasters prides itself on roasting an…
freq                                                         7      2    126
mean                                                       NaN    NaN    NaN
std                                                        NaN    NaN    NaN
min                                                        NaN    NaN    NaN
25%                                                        NaN    NaN    NaN
50%                                                        NaN    NaN    NaN
75%                                                        NaN    NaN    NaN
max                                                        NaN    NaN    NaN

[11 rows x 34 columns]

 ========== Statistical Summary of coffee_clean ==========

                           slug        aroma  acid_or_milk         body  \
count                      4887  4887.000000   4887.000000  4887.000000
unique                     4887          NaN           NaN          NaN
top     ethiopia-deri-kochoha-2          NaN           NaN          NaN
freq                          1          NaN           NaN          NaN
mean                        NaN     8.179108      7.804277     7.893452
std                         NaN     0.997619      0.964405     0.887634
min                         NaN     2.000000      2.000000     4.000000
25%                         NaN     8.000000      7.000000     7.500000
50%                         NaN     8.000000      8.000000     8.000000
75%                         NaN     9.000000      8.000000     8.000000
max                         NaN    10.000000     10.000000    10.000000

             flavor  type_with_milk  \
count   4887.000000     4887.000000
unique          NaN             NaN
top             NaN             NaN
freq            NaN             NaN
mean       8.259975        0.138122
std        1.090489        0.345063
min        1.000000        0.000000
25%        8.000000        0.000000
50%        9.000000        0.000000
75%        9.000000        0.000000
```

```
max       10.000000        1.000000

                                                  clean_text   roast_dark  \
count                                                   4887  4887.000000
unique                                                  4886          NaN
top     light bright fragrantli smooth hot aliv shimme…               NaN
freq                                                       2          NaN
mean                                                     NaN     0.044199
std                                                      NaN     0.205558
min                                                      NaN     0.000000
25%                                                      NaN     0.000000
50%                                                      NaN     0.000000
75%                                                      NaN     0.000000
max                                                      NaN     1.000000

        roast_light  roast_medium  …  region_asia_pacific  \
count   4887.000000   4887.000000  …          4887.000000
unique          NaN           NaN  …                  NaN
top             NaN           NaN  …                  NaN
freq            NaN           NaN  …                  NaN
mean       0.085533      0.277062  …             0.077348
std        0.279702      0.447593  …             0.267170
min        0.000000      0.000000  …             0.000000
25%        0.000000      0.000000  …             0.000000
50%        0.000000      0.000000  …             0.000000
75%        0.000000      1.000000  …             0.000000
max        1.000000      1.000000  …             1.000000

        region_south_america  type_espresso  type_organic  type_fair_trade  \
count            4887.000000    4887.000000   4887.000000      4887.000000
unique                   NaN            NaN           NaN              NaN
top                      NaN            NaN           NaN              NaN
freq                     NaN            NaN           NaN              NaN
mean                0.085533       0.136280      0.087375         0.056067
std                 0.279702       0.343121      0.282412         0.230075
min                 0.000000       0.000000      0.000000         0.000000
25%                 0.000000       0.000000      0.000000         0.000000
50%                 0.000000       0.000000      0.000000         0.000000
75%                 0.000000       0.000000      0.000000         0.000000
max                 1.000000       1.000000      1.000000         1.000000

        type_decaffeinated  type_pod_capsule   type_blend  type_estate  \
count          4887.000000       4887.000000  4887.000000  4887.000000
unique                 NaN               NaN          NaN          NaN
top                    NaN               NaN          NaN          NaN
freq                   NaN               NaN          NaN          NaN
mean              0.012073          0.032535     0.080213     0.134234
std               0.109222          0.177435     0.271650     0.340938
```

```
min                0.000000           0.000000     0.000000     0.000000
25%                0.000000           0.000000     0.000000     0.000000
50%                0.000000           0.000000     0.000000     0.000000
75%                0.000000           0.000000     0.000000     0.000000
max                1.000000           1.000000     1.000000     1.000000

        type_with_milk.1
count        4887.000000
unique               NaN
top                  NaN
freq                 NaN
mean            0.138122
std             0.345063
min             0.000000
25%             0.000000
50%             0.000000
75%             0.000000
max             1.000000

[11 rows x 28 columns]

 ========== Statistical Summary of coffee_id ==========

                           slug                 name             roaster  \
count                      4887                 4887                4887
unique                     4887                 4071                1131
top     ethiopia-deri-kochoha-2  Ethiopia Yirgacheffe  JBC Coffee Roasters
freq                          1                   25                 178

       rating review_date
count    4887        4887
unique     36         250
top        93  2016-11-01
freq      777          51
```

[6]:
```
# Checking common columns acress datasets

print('\n', '='*20, 'Common columns across datasets', '='*20, '\n')

common_column_coffee_raw_coffee_id = set(coffee_raw.columns).
 ↪intersection(coffee_id.columns)
print(f'\nCommon Columns in coffee_raw and coffee_id:
 ↪\n{common_column_coffee_raw_coffee_id}')

common_column_coffee_clean_coffee_id = set(coffee_clean.columns).
 ↪intersection(coffee_id.columns)
```

```
print(f'\nCommon Columns in coffee_clean and coffee_id:
↪\n{common_column_coffee_clean_coffee_id}')

common_column_coffee_raw_coffee_clean = set(coffee_raw.columns).
↪intersection(coffee_clean.columns)
print(f'\nCommon Columns in coffee_raw and coffee_clean:
↪\n{common_column_coffee_raw_coffee_clean}')
```

```
==================== Common columns across datasets ====================


Common Columns in coffee_raw and coffee_id:
{'roaster', 'name', 'slug', 'rating', 'review_date'}

Common Columns in coffee_clean and coffee_id:
{'slug'}

Common Columns in coffee_raw and coffee_clean:
{'type_organic', 'region_asia_pacific', 'type_decaffeinated', 'aroma',
'region_central_america', 'slug', 'type_pod_capsule', 'flavor',
'region_south_america', 'region_hawaii', 'type_fair_trade', 'type_estate',
'type_blend', 'region_caribbean', 'region_africa_arabia', 'body',
'type_espresso'}
```

[7]:

```
# Veryfing unique indentifires

print('\n', '='*20, 'Verify unique identifiers', '='*20, '\n')

print('\n', '='*10, 'Slugs', '='*10, '\n')
print(f"\nSlug in Coffee_raw \n{coffee_raw['slug']}")
print(f"\nSlug in Coffee_clean \n{coffee_clean['slug']}")
print(f"\nSlug in Coffee_id \n{coffee_id['slug']}")

print('\n', '='*10, 'Name', '='*10, '\n')
print(f"\nname in Coffee_raw \n{coffee_raw['name']}")
print(f"\nname in Coffee_id \n{coffee_id['name']}")

print('\n', '='*10, 'Review Date', '='*10, '\n')
print(f"\nreview_date in Coffee_raw \n{coffee_raw['review_date']}")
print(f"\nreview_date in Coffee_id \n{coffee_id['review_date']}")
```

```
==================== Verify unique identifiers ====================


========== Slugs ==========
```

```
Slug in Coffee_raw
0             /review/ethiopia-deri-kochoha-2
1                         /review/espresso-14
2              /review/kenya-ruthaka-peaberry
3            /review/ethiopia-gora-kone-sidamo
4        /review/specialty-coffee-blend-espresso
                          …
5119                     /review/beanery-blend
5120                       /review/house-blend
5121            /review/presidents-private-blend
5122                /review/traditional-roast
5123                     /review/special-roast
Name: slug, Length: 5124, dtype: object

Slug in Coffee_clean
0             ethiopia-deri-kochoha-2
1                         espresso-14
2              kenya-ruthaka-peaberry
3            ethiopia-gora-kone-sidamo
4        specialty-coffee-blend-espresso
                      …
4882                     beanery-blend
4883                       house-blend
4884            presidents-private-blend
4885                traditional-roast
4886                     special-roast
Name: slug, Length: 4887, dtype: object

Slug in Coffee_id
0             ethiopia-deri-kochoha-2
1                         espresso-14
2              kenya-ruthaka-peaberry
3            ethiopia-gora-kone-sidamo
4        specialty-coffee-blend-espresso
                      …
4882                     beanery-blend
4883                       house-blend
4884            presidents-private-blend
4885                traditional-roast
4886                     special-roast
Name: slug, Length: 4887, dtype: object

 ========== Name ==========


name in Coffee_raw
0                 Ethiopia Deri Kochoha
```

```
1                                Espresso
2                 Kenya Ruthaka Peaberry
3              Ethiopia Gora Kone Sidamo
4        Specialty Coffee Blend Espresso
                     …
5119                       Beanery Blend
5120                         House Blend
5121           President's Private Blend
5122                   Traditional Roast
5123                       Special Roast
Name: name, Length: 5124, dtype: object

name in Coffee_id
0                 Ethiopia Deri Kochoha
1                                Espresso
2                 Kenya Ruthaka Peaberry
3              Ethiopia Gora Kone Sidamo
4        Specialty Coffee Blend Espresso
                     …
4882                       Beanery Blend
4883                         House Blend
4884           President's Private Blend
4885                   Traditional Roast
4886                       Special Roast
Name: name, Length: 4887, dtype: object

 ========== Review Date ==========


review_date in Coffee_raw
0         January 2019
1         January 2019
2         January 2019
3         January 2019
4         January 2019
              …
5119    February 1997
5120    February 1997
5121    February 1997
5122    February 1997
5123    February 1997
Name: review_date, Length: 5124, dtype: object

review_date in Coffee_id
0       2019-01-01
1       2019-01-01
2       2019-01-01
3       2019-01-01
```

```
4       2019-01-01
           …
4882    1997-02-01
4883    1997-02-01
4884    1997-02-01
4885    1997-02-01
4886    1997-02-01
Name: review_date, Length: 4887, dtype: object
```

1.2 Identify Data Issues

[8]:
```python
# Checking for missing values

print('\n', '='*10, 'Missing values across datasets', '='*10, '\n')

print(f"\ncoffee_raw Missing Values: \n{coffee_raw.isnull().sum()}")
print(f"\ncoffee_clean Missing Values: \n{coffee_clean.isnull().sum()}")
print(f"\ncoffee_id Missing Values: \n{coffee_id.isnull().sum()}")
```

```
 ========== Missing values across datasets ==========


coffee_raw Missing Values:
all_text                    0
name                        0
rating                      0
roaster                     0
slug                        0
region_africa_arabia        0
region_caribbean            0
region_central_america      0
region_hawaii               0
region_asia_pacific         0
region_south_america        0
type_espresso               0
type_organic                0
type_fair_trade             0
type_decaffeinated          0
type_pod_capsule            0
type_blend                  0
type_estate                 0
location                    2
origin                    595
roast                     428
est_price                2110
review_date                 0
agtron                      0
```

```
aroma                      39
acid                      868
body                       13
flavor                     18
aftertaste               1013
with_milk                4424
desc_1                      0
desc_2                      0
desc_3                   4153
desc_4                    971
dtype: int64

coffee_clean Missing Values:
slug                     0
aroma                    0
acid_or_milk             0
body                     0
flavor                   0
type_with_milk           0
clean_text               0
roast_dark               0
roast_light              0
roast_medium             0
roast_medium_dark        0
roast_medium_light       0
roast_very_dark          0
roast_nan                0
region_africa_arabia     0
region_caribbean         0
region_central_america   0
region_hawaii            0
region_asia_pacific      0
region_south_america     0
type_espresso            0
type_organic             0
type_fair_trade          0
type_decaffeinated       0
type_pod_capsule         0
type_blend               0
type_estate              0
type_with_milk.1         0
dtype: int64

coffee_id Missing Values:
slug           0
name           0
roaster        0
rating         0
```

```
review_date    0
dtype: int64
```

[9]:
```
# Checking ratings and review_date

print('\n', '='*10, 'Checking ratings and review_date', '='*10, '\n')

print(f"\nRatings: \n{coffee_id['rating'].unique()}")
print(f"\nReview Date: \n{coffee_id['review_date'].unique()}")
```

```
 ========== Checking ratings and review_date ==========


Ratings:
['93' '91' '95' '94' '97' '92' '96' '90' '88' '83' '72' '89' '68' '63'
 '86' '84' '67' '87' '75' '85' '80' '79' '77' '82' '71' '73' '74' '78'
 '76' '66' '69' '81' '65' '70' 'NR' '60']

Review Date:
['2019-01-01' '2018-12-01' '2018-11-01' '2018-10-01' '2018-09-01'
 '2018-08-01' '2018-07-01' '2018-06-01' '2018-05-01' '2018-04-01'
 '2018-03-01' '2018-02-01' '2018-01-01' '2017-12-01' '2017-11-01'
 '2017-10-01' '2017-09-01' '2017-08-01' '2017-07-01' '2017-06-01'
 '2017-05-01' '2017-04-01' '2017-03-01' '2017-02-01' '2017-01-01'
 '2016-12-01' '2016-11-01' '2016-10-01' '2016-09-01' '2016-08-01'
 '2016-07-01' '2016-06-01' '2016-05-01' '2016-04-01' '2016-03-01'
 '2016-02-01' '2016-01-01' '2015-12-01' '2015-11-01' '2015-10-01'
 '2015-09-01' '2015-08-01' '2015-07-01' '2015-06-01' '2015-05-01'
 '2015-04-01' '2015-03-01' '2015-02-01' '2015-01-01' '2014-12-01'
 '2014-11-01' '2014-10-01' '2014-09-01' '2014-08-01' '2014-07-01'
 '2014-06-01' '2014-05-01' '2014-04-01' '2014-03-01' '2014-02-01'
 '2014-01-01' '2013-12-01' '2013-11-01' '2013-10-01' '2013-09-01'
 '2013-08-01' '2013-07-01' '2013-06-01' '2013-05-01' '2013-04-01'
 '2013-03-01' '2013-02-01' '2013-01-01' '2012-12-01' '2012-11-01'
 '2012-10-01' '2012-09-01' '2012-08-01' '2012-07-01' '2012-06-01'
 '2012-05-01' '2012-04-01' '2012-03-01' '2012-02-01' '2012-01-01'
 '2011-12-01' '2011-11-01' '2011-10-01' '2011-09-01' '2011-08-01'
 '2011-07-01' '2011-06-01' '2011-05-01' '2011-04-01' '2011-03-01'
 '2011-02-01' '2011-01-01' '2010-12-01' '2010-11-01' '2010-10-01'
 '2010-09-01' '2010-08-01' '2010-07-01' '2010-06-01' '2010-05-01'
 '2010-04-01' '2010-03-01' '2010-02-01' '2010-01-01' '2009-12-01'
 '2009-11-01' '2009-10-01' '2009-09-01' '2009-08-01' '2009-07-01'
 '2009-06-01' '2009-05-01' '2009-04-01' '2009-03-01' '2009-02-01'
 '2009-01-01' '2008-12-01' '2008-11-01' '2008-10-01' '2008-09-01'
 '2008-08-01' '2008-07-01' '2008-06-01' '2008-05-01' '2008-04-01'
 '2008-03-01' '2008-02-01' '2008-01-01' '2007-12-01' '2007-11-01'
 '2007-10-01' '2007-09-01' '2007-08-01' '2007-07-01' '2007-06-01'
```

```
 '2007-05-01' '2007-04-01' '2007-03-01' '2007-02-01' '2007-01-01'
 '2006-12-01' '2006-11-01' '2006-10-01' '2006-09-01' '2006-08-01'
 '2006-07-01' '2006-06-01' '2006-05-01' '2006-04-01' '2006-03-01'
 '2006-02-01' '2006-01-01' '2005-12-01' '2005-11-01' '2005-10-01'
 '2005-09-01' '2005-08-01' '2005-07-01' '2005-06-01' '2005-05-01'
 '2005-04-01' '2005-03-01' '2005-02-01' '2005-01-01' '2004-12-01'
 '2004-11-01' '2004-10-01' '2004-09-01' '2004-08-01' '2004-07-01'
 '2004-06-01' '2004-05-01' '2004-04-01' '2004-03-01' '2004-02-01'
 '2004-01-01' '2003-12-01' '2003-11-01' '2003-10-01' '2003-09-01'
 '2003-08-01' '2003-07-01' '2003-06-01' '2003-05-01' '2003-03-01'
 '2003-02-01' '2003-01-01' '2002-12-01' '2002-11-01' '2002-10-01'
 '2002-09-01' '2002-08-01' '2002-07-01' '2002-06-01' '2002-05-01'
 '2002-04-01' '2002-03-01' '2002-02-01' '2001-12-01' '2001-08-01'
 '2001-06-01' '2001-04-01' '2001-03-01' '2001-01-01' '2000-12-01'
 '2000-09-01' '2000-07-01' '2000-06-01' '2000-05-01' '2000-04-01'
 '2000-03-01' '2000-02-01' '2000-01-01' '1999-12-01' '1999-11-01'
 '1999-10-01' '1999-09-01' '1999-08-01' '1999-07-01' '1999-06-01'
 '1999-05-01' '1999-04-01' '1999-03-01' '1999-02-01' '1999-01-01'
 '1998-12-01' '1998-11-01' '1998-10-01' '1998-09-01' '1998-07-01'
 '1998-06-01' '1998-05-01' '1998-04-01' '1998-03-01' '1998-02-01'
 '1998-01-01' '1997-12-01' '1997-11-01' '1997-09-01' '1997-08-01'
 '1997-07-01' '1997-05-01' '1997-04-01' '1997-03-01' '1997-02-01']
```

[10]:
```python
# Veryfing duplicated rows across datasets

print('\n', '='*10, 'Duplicated rows across datasets', '='*10, '\n')

print(f"coffee_raw Duplicated Values: {coffee_raw.duplicated().sum()}")
print(f"\ncoffee_clean Duplicated Values: {coffee_clean.duplicated().sum()}")
print(f"\ncoffee_id Duplicated Values: {coffee_id.duplicated().sum()}")
```

```
 ========== Duplicated rows across datasets ==========

coffee_raw Duplicated Values: 0

coffee_clean Duplicated Values: 0

coffee_id Duplicated Values: 0
```

[11]:
```python
# Veryfing duplicated entries in slug or name

print('\n', '='*10, 'Duplicated slugs/names across datasets', '='*10, '\n')


print(f"\ncoffe_raw Duplicated Slug Values: {coffee_raw['slug'].duplicated().
 ↪sum()}")
```

```
print(f"\ncoffee_clean Duplicated Slug Values: {coffee_clean['slug'].
    duplicated().sum()}")
print(f"\ncoffee_id Duplicated Slug Values: {coffee_id['slug'].duplicated().
    sum()}")

print(f"\n\ncoffee_raw Duplicated name Values: {coffee_raw['name'].duplicated().
    sum()}")
print(f"\ncoffee_id Duplicated name Values: {coffee_id['name'].duplicated().
    sum()}")
```

```
 ========== Duplicated slugs/names across datasets ==========


coffe_raw Duplicated Slug Values: 0

coffee_clean Duplicated Slug Values: 0

coffee_id Duplicated Slug Values: 0


coffee_raw Duplicated name Values: 892

coffee_id Duplicated name Values: 816
```

[12]:
```
# Veryfing Inconsistencies in categorical value

print('\n', '='*10, 'Inconsistencies in categorical value', '='*10, '\n')

print(f"\nRoast Types: {coffee_raw['roast'].unique()}")

print(f"\nRegions in coffe_raw:\n{coffee_raw[['region_africa_arabia',
    'region_caribbean', 'region_central_america', 'region_hawaii',
    'region_asia_pacific', 'region_south_america']].nunique()}")
print(f"\nRegions in coffe_clean:\n{coffee_clean[['region_africa_arabia',
    'region_caribbean', 'region_central_america', 'region_hawaii',
    'region_asia_pacific', 'region_south_america']].nunique()}")
print(f"\nTypes in coffe_clean\n{coffee_clean[['type_espresso', 'type_organic',
    'type_fair_trade', 'type_decaffeinated', 'type_pod_capsule', 'type_blend',
    'type_estate', 'type_with_milk.1']].nunique()}")

print(f"\nRegions in coffe_raw:\n{coffee_raw[['region_africa_arabia',
    'region_caribbean', 'region_central_america', 'region_hawaii',
    'region_asia_pacific', 'region_south_america']].head(5)}")
print(f"\nRegions in coffe_clean:\n{coffee_clean[['region_africa_arabia',
    'region_caribbean', 'region_central_america', 'region_hawaii',
    'region_asia_pacific', 'region_south_america']].head(5).head(5)}")
```

```
print(f"\nTypes in coffe_clean\n{coffee_clean[['type_espresso', 'type_organic',␣
↪'type_fair_trade', 'type_decaffeinated', 'type_pod_capsule', 'type_blend',␣
↪'type_estate', 'type_with_milk.1']].head(5)}")

print(f"\nSlug in Coffee_raw \n{coffee_raw['slug'].head(10)}")
```

```
 ========== Inconsistencies in categorical value ==========


Roast Types: ['Medium-Light' 'Medium' 'Light' nan 'Medium-Dark' 'Very Dark'
'Dark']

Regions in coffe_raw:
region_africa_arabia      2
region_caribbean          2
region_central_america    2
region_hawaii             2
region_asia_pacific       2
region_south_america      2
dtype: int64

Regions in coffe_clean:
region_africa_arabia      2
region_caribbean          2
region_central_america    2
region_hawaii             2
region_asia_pacific       2
region_south_america      2
dtype: int64

Types in coffe_clean
type_espresso         2
type_organic          2
type_fair_trade       2
type_decaffeinated    2
type_pod_capsule      2
type_blend            2
type_estate           2
type_with_milk.1      2
dtype: int64

Regions in coffe_raw:
   region_africa_arabia  region_caribbean  region_central_america  \
0                     1                 0                       0
1                     0                 0                       0
2                     1                 0                       0
3                     1                 0                       0
```

```
4                    0                0                      0

   region_hawaii  region_asia_pacific  region_south_america
0              0                    0                     0
1              0                    1                     0
2              0                    0                     0
3              0                    0                     0
4              0                    0                     0

Regions in coffe_clean:
   region_africa_arabia  region_caribbean  region_central_america  \
0                     1                 0                       0
1                     0                 0                       0
2                     1                 0                       0
3                     1                 0                       0
4                     0                 0                       0

   region_hawaii  region_asia_pacific  region_south_america
0              0                    0                     0
1              0                    1                     0
2              0                    0                     0
3              0                    0                     0
4              0                    0                     0

Types in coffe_clean
   type_espresso  type_organic  type_fair_trade  type_decaffeinated  \
0              0             0                0                   0
1              1             0                0                   0
2              0             0                0                   0
3              0             0                0                   0
4              1             0                0                   0

   type_pod_capsule  type_blend  type_estate  type_with_milk.1
0                 0           0            0                 0
1                 0           0            1                 1
2                 0           0            0                 0
3                 0           0            0                 0
4                 0           0            0                 1

Slug in Coffee_raw
0            /review/ethiopia-deri-kochoha-2
1                        /review/espresso-14
2             /review/kenya-ruthaka-peaberry
3          /review/ethiopia-gora-kone-sidamo
4    /review/specialty-coffee-blend-espresso
5      /review/honduras-las-flores-parainema
6                      /review/kivu-dr-congo
7                 /review/santa-luzia-brazil
```

```
8      /review/porfirio-castellanos-honduras
9        /review/ethiopia-yirgacheffe-awassa
Name: slug, dtype: object
```

[13]:
```
# Checking origin in coffee_raw

print(coffee_raw['origin'].head(20))
```

```
0     West Guji Zone, Oromia Region, southeastern Et…
1                                   Northern Thailand
2             Nyeri growing region, south-central Kenya
3     Sidamo (also Sidama) growing region, south-cen…
4                           Ethiopia, Colombia, Kenya
5                             Santa Bárbara, Honduras
6     Kalehe, South Kivu Province, Democratic Republ…
7     Cerrado Mineiro growing region, Santa Luzia, B…
8                             Santa Bárbara, Honduras
9     Yirgacheffe growing region, south-central Ethi…
10                Chiriqui Province, far western Panama
11                          Tolima Department, Colombia
12                                      Huila, Colombia
13                         Jutiapa Department, Guatemala
14    Alaka District, Guji Zone, Oromia region, sout…
15                          Guji Zone, southern Ethiopia
16            Nyeri growing region, south-central Kenya
17                        São Sebastião de Grama, Brazil
18        Yirgacheffe growing region, southern Ethiopia
19                   Antigua growing region, Guatemala
Name: origin, dtype: object
```

[14]:
```
# Checking roast in coffee_raw

print(coffee_raw['roast'].head(20))
```

```
0     Medium-Light
1           Medium
2           Medium
3     Medium-Light
4     Medium-Light
5     Medium-Light
6     Medium-Light
7     Medium-Light
8     Medium-Light
9     Medium-Light
10    Medium-Light
11    Medium-Light
12    Medium-Light
13    Medium-Light
14    Medium-Light
```

```
15    Medium-Light
16    Medium-Light
17    Medium-Light
18    Medium-Light
19    Medium-Light
Name: roast, dtype: object
```

1.3 Summary of Data Exploration

We are working with three datasets: coffee_raw, coffee_clean, and coffee_id.

coffee_raw:

- *Record Count:* 5,124 rows.
- *Data Quality:* Many columns contain a significant amount of missing values, particularly columns like withmilk, est_price, aftertaste, desc3, and desc4.
- *Slug Column:* The slug field includes unwanted prefixes like /review/. It requires cleaning.
- *One-Hot Encoded Columns:* Coffee regions and coffee types are already one-hot encoded.
- *Review Date:* Available in the format January 2019
- *Duplications*: No duplicated rows overall; however, there are duplicated coffee names

coffee_clean:

- *Record Count:* 4,887 rows.
- *Data Quality:* No missing values across any column.
- *Slug Column:* The slug field is clean and formatted correctly.
- *One-Hot Encoded Columns:* Coffee regions, coffee types and coffee roast are already one-hot encoded.
- *Roast Column:* Some inconsistencies are present, for example: medium-light, very dark.
- *Duplications*: No duplicated rows overall

coffee_id:

- *Record Count:* 4,887 rows.
- *Data Quality:* No missing values across any column.
- *Slug Column:* The slug field is clean and formatted correctly.
- *Name Column:* The name field is consistent and matches coffee_raw.
- *Review Date:* Provided in the dd.mm.yyyy format.
- *Ratings:* One record has an invalid rating marked as NR.
- *Duplications*: No duplicated rows overall; however, there are duplicated coffee names

Proposed solution

1. Clean the slug field in coffee_raw to match the formatting of coffee_clean and coffee_id, removing prefixes such as /review/
2. Merge coffee_raw, coffee_clean, and coffee_id
3. Drop unnecessary columns like : all_text, , est_price, desc_3, desc_4, with_milk
4. Verify whether type_with_milk and type_with_milk_1 represent the same data and if they are redundant, retain only one of them.

5. Reverse one-hot encoding for region, type
6. Standardize the values in the roast column and drop one hot encoded roasts
7. Use roast levels to infer and populate an aftertaste and acid feature
8. Replace invalid rating entries with an estimated rating based on the associated roaster's average rating and standardize all rating.
9. Split the review_date into month and year

2 Data Cleaning and Transformation

[15]:
```python
# Fixing the slug in coffee_raw

coffee_raw['slug'] = coffee_raw['slug'].str.replace('/review/', '', regex=False)
```

[16]:
```python
# Merging dataframes of coffee_clean and coffee_id

merged_data = coffee_clean.merge(coffee_id, on="slug", how="inner")
```

[17]:
```python
# List of column names that are in coffee_raw but not in merged_data

extra_columns = coffee_raw.columns.difference(merged_data.columns).tolist()
print(extra_columns)
```

```
['acid', 'aftertaste', 'agtron', 'all_text', 'desc_1', 'desc_2', 'desc_3',
'desc_4', 'est_price', 'location', 'origin', 'roast', 'with_milk']
```

[18]:
```python
# Merging merged_data with selected columns from coffee_raw on the slug column

extra_columns = ['slug', 'acid', 'aftertaste', 'agtron', 'all_text', 'desc_1',␣
↪'desc_2', 'desc_3', 'desc_4', 'est_price', 'location', 'origin', 'roast',␣
↪'with_milk']

cleaned_data = merged_data.merge(coffee_raw[extra_columns], on="slug",␣
↪how="left")
cleaned_data.head()
```

[18]:
```
                              slug  aroma  acid_or_milk  body  flavor  \
0          ethiopia-deri-kochoha-2    9.0           8.0   9.0     9.0
1                      espresso-14    8.0           9.0   8.0     8.0
2            kenya-ruthaka-peaberry    9.0           8.0   9.0    10.0
3         ethiopia-gora-kone-sidamo    9.0           8.0   9.0     9.0
4  specialty-coffee-blend-espresso    9.0           9.0   8.0     9.0

   type_with_milk                                         clean_text  \
0               0  bright crisp sweetli tart citru medley cacao n…
1               1  evalu espresso deepli rich sweetli roast round…
2               0  deepli sweet richli savori dark chocol pistach…
3               0  fruit forward richli chocolati raspberri couli…
```

```
4              1  evalu espresso rich chocolati sweetli tart dar…

   roast_dark  roast_light  roast_medium  …  \
0           0            0             0  …
1           0            0             1  …
2           0            0             1  …
3           0            0             0  …
4           0            0             0  …

                                             all_text  \
0  \n\n\n\n \n93\nFlight Coffee Co.\nEthiopia Der…
1  \n\n\n\n\n91\nDoi Chaang Coffee\nEspresso\nLoc…
2  \n\n\n\n \n95\nTemple Coffee and Tea\nKenya Ru…
3  \n\n\n\n \n93\nTemple Coffee and Tea\nEthiopia…
4  \n\n\n\n\n93\nChoosy Gourmet\nSpecialty Coffee…

                                               desc_1  \
0  Bright, crisp, sweetly tart. Citrus medley, ca…
1  Evaluated as espresso. Deeply rich, sweetly ro…
2  Deeply sweet, richly savory. Dark chocolate, p…
3  Fruit-forward, richly chocolaty. Raspberry cou…
4  Evaluated as espresso. Rich, chocolaty, sweetl…

                                               desc_2  \
0  From the Deri Kochoha mill in the Hagere Marya…
1  Doi Chaang is a single-estate coffee produced …
2  Despite challenges ranging from contested gove…
3  Southern Ethiopia coffees like this one are la…
4  A blend of coffees from Ethiopia (natural-proc…

                                               desc_3  desc_4  \
0  A poised and melodic wet-processed Ethiopia co…     NaN
1  A rich, resonant espresso from Thailand, espec…     NaN
2  A high-toned, nuanced Kenya cup, classic in it…     NaN
3  A playful, unrestrained fruit bomb of a coffee…     NaN
4  An espresso blend in which spice notes - in pa…     NaN

              est_price                             location  \
0      $17.00/12 ounces             Bedford, New Hampshire
1  CAD $29.99/32 ounces  Richmond, British Columbia, Canada
2      $19.00/12 ounces             Sacramento, California
3      $20.00/12 ounces             Sacramento, California
4     NT $250/16 ounces                  Kaohsiung, Taiwan

                                               origin         roast  with_milk
0  West Guji Zone, Oromia Region, southeastern Et…  Medium-Light        NaN
1                                Northern Thailand        Medium        9.0
```

```
2          Nyeri growing region, south-central Kenya        Medium        NaN
3  Sidamo (also Sidama) growing region, south-cen…  Medium-Light       NaN
4                            Ethiopia, Colombia, Kenya  Medium-Light        9.0

[5 rows x 45 columns]
```

[19]:

```
# Data information

cleaned_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4887 entries, 0 to 4886
Data columns (total 45 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   slug                    4887 non-null   object 
 1   aroma                   4887 non-null   float64
 2   acid_or_milk            4887 non-null   float64
 3   body                    4887 non-null   float64
 4   flavor                  4887 non-null   float64
 5   type_with_milk          4887 non-null   int64  
 6   clean_text              4887 non-null   object 
 7   roast_dark              4887 non-null   int64  
 8   roast_light             4887 non-null   int64  
 9   roast_medium            4887 non-null   int64  
 10  roast_medium_dark       4887 non-null   int64  
 11  roast_medium_light      4887 non-null   int64  
 12  roast_very_dark         4887 non-null   int64  
 13  roast_nan               4887 non-null   int64  
 14  region_africa_arabia    4887 non-null   int64  
 15  region_caribbean        4887 non-null   int64  
 16  region_central_america  4887 non-null   int64  
 17  region_hawaii           4887 non-null   int64  
 18  region_asia_pacific     4887 non-null   int64  
 19  region_south_america    4887 non-null   int64  
 20  type_espresso           4887 non-null   int64  
 21  type_organic            4887 non-null   int64  
 22  type_fair_trade         4887 non-null   int64  
 23  type_decaffeinated      4887 non-null   int64  
 24  type_pod_capsule        4887 non-null   int64  
 25  type_blend              4887 non-null   int64  
 26  type_estate             4887 non-null   int64  
 27  type_with_milk.1        4887 non-null   int64  
 28  name                    4887 non-null   object 
 29  roaster                 4887 non-null   object 
 30  rating                  4887 non-null   object 
 31  review_date             4887 non-null   object 
 32  acid                    4220 non-null   float64
```

```
 33  aftertaste              3959 non-null   float64
 34  agtron                  4887 non-null   object
 35  all_text                4887 non-null   object
 36  desc_1                  4887 non-null   object
 37  desc_2                  4887 non-null   object
 38  desc_3                  953 non-null    object
 39  desc_4                  3934 non-null   object
 40  est_price               2949 non-null   object
 41  location                4885 non-null   object
 42  origin                  4334 non-null   object
 43  roast                   4516 non-null   object
 44  with_milk               675 non-null    float64
dtypes: float64(7), int64(22), object(16)
memory usage: 1.7+ MB
```

2.1 Handle Missing Data

[20]:
```
# Dropping non-essential columns

cleaned_data = cleaned_data.drop(columns=['all_text', 'desc_3', 'desc_4',␣
↪'est_price', 'with_milk'])
```

[21]:
```
# Checking if the columns type_with_milk and type_with_milk.1 in cleaned_data␣
↪are identical

are_identical = cleaned_data['type_with_milk'].
↪equals(cleaned_data['type_with_milk.1'])
print(are_identical)
```

```
True
```

[22]:
```
# Dropping one of the columns

cleaned_data = cleaned_data.drop(columns=['type_with_milk.1'])
```

[23]:
```
# Mapping region indicator columns to region names

region_mapping = {
    'region_africa_arabia': 'Africa Arabia',
    'region_caribbean': 'Caribbean',
    'region_central_america': 'Central America',
    'region_hawaii': 'Hawaii',
    'region_asia_pacific': 'Asia Pacific',
    'region_south_america': 'South America'
}

def get_region(row):
    for col, region_name in region_mapping.items():
```

```python
        if row[col] == 1:
            return region_name
    return 'Unknown'

cleaned_data['region'] = cleaned_data.apply(get_region, axis=1)

print(cleaned_data[['region']].value_counts())
print(cleaned_data[['region']].count())
```

```
region
Unknown            2064
Africa Arabia      1107
Central America     806
South America       407
Asia Pacific        361
Hawaii              100
Caribbean            42
Name: count, dtype: int64
region    4887
dtype: int64
```

[24]:

```python
# Updating the region column by mapping unknown regions based on keywords found␣
↪in the 'origin' column

def map_region_from_text(row):
    africa_arabia_keywords = [
    'Algeria', 'Angola', 'Benin', 'Botswana', 'Burkina Faso', 'Burundi',
    'Cabo Verde', 'Cameroon', 'Central African Republic', 'Chad', 'Comoros',
    'Democratic Republic of the Congo', 'Djibouti', 'Egypt', 'Equatorial␣
↪Guinea',
    'Eritrea', 'Eswatini', 'Ethiopia', 'Gabon', 'Gambia', 'Ghana', 'Guinea',
    'Guinea-Bissau', 'Ivory Coast', 'Kenya', 'Lesotho', 'Liberia', 'Libya',
    'Madagascar', 'Malawi', 'Mali', 'Mauritania', 'Mauritius', 'Morocco',
    'Mozambique', 'Namibia', 'Niger', 'Nigeria', 'Republic of the Congo',
    'Rwanda', 'Sao Tome and Principe', 'Senegal', 'Seychelles', 'Sierra Leone',
    'Somalia', 'South Africa', 'South Sudan', 'Sudan', 'Tanzania', 'Togo',
    'Tunisia', 'Uganda', 'Zambia', 'Zimbabwe', 'Africa', 'Arabia'
]

    asia_and_pacific_keywords = [
    'Afghanistan', 'Armenia', 'Azerbaijan', 'Bahrain', 'Bangladesh', 'Bhutan',
    'Brunei', 'Cambodia', 'China', 'Cyprus', 'Georgia', 'India', 'Indonesia',
    'Iran', 'Iraq', 'Israel', 'Japan', 'Jordan', 'Kazakhstan', 'Kuwait',
    'Kyrgyzstan', 'Laos', 'Lebanon', 'Malaysia', 'Maldives', 'Mongolia',
    'Myanmar', 'Nepal', 'North Korea', 'Oman', 'Pakistan', 'Palestine',
    'Philippines', 'Qatar', 'Saudi Arabia', 'Singapore', 'South Korea',
    'Sri Lanka', 'Syria', 'Tajikistan', 'Thailand', 'Timor-Leste',
```

```
    'Turkey', 'Turkmenistan', 'United Arab Emirates', 'Uzbekistan',
    'Vietnam', 'Yemen', 'Taiwan', 'Australia', 'Fiji', 'Kiribati', 'Marshall␣
↪Islands', 'Micronesia',
    'Nauru', 'New Zealand', 'Palau', 'Papua New Guinea', 'Samoa', 'Timor',␣
↪'Sumatra', 'Asia', 'Pacific'
]
    south_america_keywords = [
    'Colombia', 'Brazil', 'Peru', 'Ecuador', 'Honduras', 'Bolivia',
    'Argentina', 'Chile', 'Paraguay', 'Uruguay', 'Venezuela', 'Guyana',
    'Suriname', 'South America', 'Latin America'
]

    central_america_keywords = [
    'Costa Rica', 'Panama', 'Guatemala', 'Mexico', 'El Salvador', 'Nicaragua',␣
↪'Belize', 'Central America'
]

    caribbean_keywords = [
    'Puerto Rico', 'Jamaica', 'Dominican Republic', 'Haiti', 'Cuba',
    'Barbados', 'Saint Lucia', 'Saint Vincent and the Grenadines',
    'Trinidad and Tobago', 'Antigua and Barbuda', 'Saint Kitts and Nevis',
    'Grenada', 'Bahamas', 'Aruba', 'Cayman Islands', 'Bermuda', 'French Guiana'
]

    hawaii_keywords = ['Hawaii']

    if row['region'] == 'Unknown':
        if isinstance(row['origin'], str):
            if any(keyword in row['origin'] for keyword in␣
↪africa_arabia_keywords):
                return 'Africa Arabia'
            elif any(keyword in row['origin'] for keyword in␣
↪asia_and_pacific_keywords):
                return 'Asia Pacific'
            elif any(keyword in row['origin'] for keyword in␣
↪south_america_keywords):
                return 'South America'
            elif any(keyword in row['origin'] for keyword in␣
↪caribbean_keywords):
                return 'Caribbean'
            elif any(keyword in row['origin'] for keyword in␣
↪central_america_keywords):
                return 'Central America'
            elif any(keyword in row['origin'] for keyword in hawaii_keywords):
                return 'Hawaii'
    return row['region']
```

```
cleaned_data['region'] = cleaned_data.apply(map_region_from_text, axis=1)
print(cleaned_data['region'].value_counts())
print(cleaned_data[['region']].count())
```

```
region
Africa Arabia      1662
Central America    1106
Unknown             693
South America       688
Asia Pacific        543
Hawaii              136
Caribbean            59
Name: count, dtype: int64
region    4887
dtype: int64
```

[25]:

```
# Cleaning roaster

cleaned_data['roaster'] = cleaned_data['roaster'].str.strip().str.lower()
```

[26]:

```
# Filling 'Unknown' regions based on the most common region per roaster

roaster_to_region = cleaned_data[cleaned_data['region'] != 'Unknown'].
 ↪groupby('roaster')['region'].agg(lambda x: x.mode()[0])
cleaned_data.loc[cleaned_data['region'] == 'Unknown', 'region'] =␣
 ↪cleaned_data['roaster'].map(roaster_to_region).fillna('Unknown')

print(cleaned_data['region'].value_counts())
print(cleaned_data[['region']].count())
```

```
region
Africa Arabia      1871
Central America    1193
South America       721
Asia Pacific        640
Unknown             254
Hawaii              144
Caribbean            64
Name: count, dtype: int64
region    4887
dtype: int64
```

[27]:

```
# Filling 'Unknown' regions based on the most common region per location

location_to_region = cleaned_data[cleaned_data['region'] != 'Unknown'].
 ↪groupby('location')['region'].agg(lambda x: x.mode()[0])
cleaned_data.loc[cleaned_data['region'] == 'Unknown', 'region'] =␣
 ↪cleaned_data['location'].map(location_to_region).fillna('Unknown')
```

```
print(cleaned_data['region'].value_counts())
print(cleaned_data[['region']].count())
```

```
region
Africa Arabia      1914
Central America    1204
South America       733
Asia Pacific        669
Unknown             158
Hawaii              145
Caribbean            64
Name: count, dtype: int64
region    4887
dtype: int64
```

[28]:
```
# Filling missing values as 'Unknown'

cleaned_data['origin'].fillna('Unknown', inplace=True)
cleaned_data['location'].fillna('Unknown', inplace=True)
```

[29]:
```
# Dropping encoded region columns

cleaned_data = cleaned_data.drop(columns=['region_africa_arabia',␣
↪'region_caribbean', 'region_central_america', 'region_hawaii',␣
↪'region_asia_pacific', 'region_south_america'])
```

[30]:
```
# Replacing 'Very Dark' with 'Very-Dark' and fills missing roast values with␣
↪'Unknown'

cleaned_data['roast'] = cleaned_data['roast'].replace('Very Dark', 'Very-Dark').
↪fillna('Unknown')

print(cleaned_data['roast'].value_counts())
print(cleaned_data['roast'].count())
```

```
roast
Medium-Light    1661
Medium          1354
Medium-Dark      764
Light            418
Unknown          371
Dark             216
Very-Dark        103
Name: count, dtype: int64
4887
```

[31]:
```python
# Filling 'Unknown' values in the roast column based on the most popular roast␣
↪for each region

popular_roast_per_region = cleaned_data[cleaned_data['roast'] != 'Unknown'].
↪groupby('region')['roast'].agg(lambda x: x.mode()[0])

cleaned_data.loc[cleaned_data['roast'] == 'Unknown', 'roast'] =␣
↪cleaned_data['region'].map(popular_roast_per_region)

print(cleaned_data['roast'].value_counts())
print(cleaned_data['roast'].count())
```

```
roast
Medium-Light    1878
Medium          1427
Medium-Dark      845
Light            418
Dark             216
Very-Dark        103
Name: count, dtype: int64
4887
```

[32]:
```python
# Dropping encoded roast columns

cleaned_data = cleaned_data.drop(columns=['roast_dark', 'roast_light',␣
↪'roast_medium', 'roast_medium_dark', 'roast_medium_light',␣
↪'roast_very_dark', 'roast_nan'])
```

[33]:
```python
# Mapping type indicator columns to type names

type_mapping = {
    'type_espresso': 'espresso',
    'type_organic': 'organic',
    'type_fair_trade': 'fair trade',
    'type_decaffeinated': 'decaffeinated',
    'type_pod_capsule': 'pod capsule',
    'type_blend': 'blend',
    'type_estate': 'estate',
    'type_with_milk': 'with milk'
}

def get_type(row):
    for col, type_name in type_mapping.items():
        if row[col] == 1:
            return type_name
    return 'Unknown'
```

```python
cleaned_data['type'] = cleaned_data.apply(get_type, axis=1)

print(cleaned_data[['type']].value_counts())
print(cleaned_data[['type']].count())
```

```
type
Unknown          2851
espresso          666
estate            574
organic           387
blend             179
pod capsule       109
fair trade         56
decaffeinated      34
with milk          31
Name: count, dtype: int64
type    4887
dtype: int64
```

[34]:
```python
# Updating the type column by mapping unknown types based on keywords found in␣
↪the 'desc_1' and 'desc_2' columns

def map_type_from_text(row):
    espresso_keywords = [
    'espresso', 'espresso roast', 'strong coffee', 'dark roast', 'ristretto',
    'doppio', 'short black', 'long black', 'intense coffee', 'italian roast',
    'full-bodied', 'bold coffee', 'high-intensity', 'robust coffee', 'strong',␣
↪'intense',
    'deep roast', 'powerful roast', 'extra dark', 'turbo shot', 'midnight roast'
]

    estate_keywords = [
    'estate', 'single origin', 'exclusive estate', 'specialty coffee',
    'microlot', 'limited edition', 'direct trade', 'farm-to-cup',
    'private estate', 'regional selection', 'terroir coffee', 'exclusive',␣
↪'specialty',
    'origin select', 'handcrafted', 'small-batch', 'private reserve', 'artisan␣
↪farm'
]

    organic_keywords = [
    'organic', 'bio', 'eco-friendly', 'certified organic', 'natural coffee',
    'biodynamic', 'chemical-free', 'sustainable farm', 'shade-grown',
    'wild-grown', 'no pesticides', 'handpicked', 'natural', 'earth-friendly',
    'pure coffee', 'non-GMO', 'green certified', 'organic beans',␣
↪'soil-friendly'
]
```

```
    blend_keywords = [
    'blend', 'house blend', 'signature blend', 'special blend', 'barista blend',
    'classic blend', 'artisan blend', 'balanced roast', 'custom blend',
    'all-purpose blend', 'breakfast blend', 'medium-dark blend',
    'fusion roast', 'flavorful mix', 'harmonized roast', 'smooth blend',␣
 ↪'complex blend'
]

    pod_capsule_keywords = [
    'pod capsule', 'pod_capsule', 'coffee pod', 'capsule coffee',␣
 ↪'single-serve',
    'compatible capsule', 'nespresso pod', 'k-cup', 'keurig pod', 'dolce gusto␣
 ↪pod',
    'tassimo pod', 'keurig-compatible', 'easy serving espresso', 'E.S.E pod',␣
 ↪'capsule', 'pod',
    'one-cup', 'quick brew', 'easy brew', 'fast coffee', 'machine-friendly'
]

    fair_trade_keywords = [
    'fair trade', 'fair_trade', 'ethical trade', 'fairly traded', 'sustainable␣
 ↪coffee',
    'responsibly sourced', 'fair wage', 'rainforest alliance', 'UTZ certified',
    'direct trade', 'social impact coffee', 'people-first coffee', 'fair',␣
 ↪'trade',
    'equitable sourcing', 'human-first', 'eco-conscious', 'better future',␣
 ↪'community coffee'
]

    decaffeinated_keywords = [
    'decaffeinated', 'decaf', 'caffeine-free', 'low caffeine', 'half-caf',
    'water process decaf', 'swiss water process', 'chemical-free decaf',
    'naturally decaffeinated', 'mellow decaf', 'smooth decaf',
    'zero caffeine', 'night-friendly', 'late-night coffee', 'relaxing brew',␣
 ↪'gentle coffee'
]

    with_milk_keywords = [
    'with milk', 'latte', 'cappuccino', 'milk-based', 'flat white',
    'macchiato', 'mocha', 'creamy coffee', 'steamed milk', 'frothy milk',
    'white coffee', 'milky espresso', 'coffee with cream', 'milk', 'milky',
    'velvety texture', 'buttery smooth', 'frothy delight', 'rich foam', 'soft␣
 ↪crema'
]

    if row['type'] == 'Unknown':
```

```
        if isinstance(row['desc_2'], str):
            desc_text = (str(row['desc_1']) + " " + str(row['desc_2'])).lower()
            if any(keyword in desc_text for keyword in espresso_keywords):
                return 'espresso'
            elif any(keyword in desc_text for keyword in estate_keywords):
                return 'estate'
            elif any(keyword in desc_text for keyword in organic_keywords):
                return 'organic'
            elif any(keyword in desc_text for keyword in blend_keywords):
                return 'blend'
            elif any(keyword in desc_text for keyword in pod_capsule_keywords):
                return 'pod capsule'
            elif any(keyword in desc_text for keyword in fair_trade_keywords):
                return 'fair trade'
            elif any(keyword in desc_text for keyword in␣
 ↪decaffeinated_keywords):
                return 'decaffeinated'
            elif any(keyword in desc_text for keyword in with_milk_keywords):
                return 'with milk'
            else:
                return 'general coffee'
    return row['type']

cleaned_data['type'] = cleaned_data.apply(map_type_from_text, axis=1)
print(cleaned_data['type'].value_counts())
print(cleaned_data[['type']].count())
```

```
type
estate            1445
espresso          1298
general coffee     737
organic            732
blend              290
fair trade         142
pod capsule        115
with milk           76
decaffeinated       52
Name: count, dtype: int64
type    4887
dtype: int64
```

[35]:
```
# Dropping encoded type columns

cleaned_data = cleaned_data.drop(columns=['type_espresso', 'type_organic',␣
 ↪'type_fair_trade', 'type_decaffeinated', 'type_pod_capsule', 'type_blend',␣
 ↪'type_estate', 'type_with_milk'])
```

[36]:
```python
# Checking 'acid_or_milk'

cleaned_data['acid_or_milk']
```

[36]:
```
0       8.0
1       9.0
2       8.0
3       8.0
4       9.0
       …
4882    7.0
4883    8.0
4884    5.0
4885    6.0
4886    5.0
Name: acid_or_milk, Length: 4887, dtype: float64
```

[37]:
```python
# Dropping 'acid_or_milk' column

cleaned_data = cleaned_data.drop(columns='acid_or_milk')
```

[38]:
```python
# Checking 'aftertaste' and 'acid' column

cleaned_data[['aftertaste', 'acid']]
```

[38]:
```
      aftertaste  acid
0            8.0   8.0
1            8.0   NaN
2            8.0   8.0
3            8.0   8.0
4            8.0   NaN
…            …     …
4882         NaN   7.0
4883         NaN   8.0
4884         NaN   5.0
4885         NaN   6.0
4886         NaN   5.0

[4887 rows x 2 columns]
```

[39]:
```python
# Filling missing values in the 'aftertaste' column with the mean aftertaste␣
↪value for each roast group

cleaned_data['aftertaste'] = cleaned_data.groupby('roast')['aftertaste'].
↪transform(lambda x: x.fillna(x.mean()))
```

[40]:
```
# Filling missing values in the 'acid' column with the mean aftertaste value␣
↪for each roast group

cleaned_data['acid'] = cleaned_data.groupby('roast')['acid'].transform(lambda x:
↪ x.fillna(x.mean()))
```

[41]:
```
# Converting the rating column to numeric values

cleaned_data['rating'] = pd.to_numeric(cleaned_data['rating'], errors='coerce')
```

[42]:
```
# Replacing missing rating values with the average rating for each roaster

roaster_avg_rating = cleaned_data.groupby('roaster')['rating'].mean()
cleaned_data['rating'] = cleaned_data.apply(lambda x:␣
↪roaster_avg_rating[x['roaster']] if pd.isna(x['rating']) else x['rating'],␣
↪axis=1)
cleaned_data['rating'].unique()
```

[42]:
```
array([93.  , 91.  , 95.  , 94.  , 97.  , 92.  , 96.  , 90.  , 88.  ,
       83.  , 72.  , 89.  , 68.  , 63.  , 86.  , 84.  , 67.  , 87.  ,
       75.  , 85.  , 80.  , 79.  , 77.  , 82.  , 71.  , 73.  , 74.  ,
       78.  , 76.  , 66.  , 69.  , 81.  , 65.  , 70.  , 85.25, 81.6 ,
       60.  ])
```

[43]:
```
# Data information

cleaned_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4887 entries, 0 to 4886
Data columns (total 19 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   slug         4887 non-null   object
 1   aroma        4887 non-null   float64
 2   body         4887 non-null   float64
 3   flavor       4887 non-null   float64
 4   clean_text   4887 non-null   object
 5   name         4887 non-null   object
 6   roaster      4887 non-null   object
 7   rating       4887 non-null   float64
 8   review_date  4887 non-null   object
 9   acid         4887 non-null   float64
 10  aftertaste   4887 non-null   float64
 11  agtron       4887 non-null   object
 12  desc_1       4887 non-null   object
 13  desc_2       4887 non-null   object
 14  location     4887 non-null   object
```

```
 15  origin        4887 non-null   object
 16  roast         4887 non-null   object
 17  region        4887 non-null   object
 18  type          4887 non-null   object
dtypes: float64(6), object(13)
memory usage: 725.5+ KB
```

2.2 Resolve Duplicates

[44]:
```python
# Counting the number of duplicate rows based on the slug and review_date␣
↪columns.

duplicates = cleaned_data.duplicated(subset=['slug', 'review_date']).sum()
print(f"Number of duplicates: {duplicates}")
```

```
Number of duplicates: 0
```

[45]:
```python
# Identifying the rows where there are conflicts in the name column for the same␣
↪slug and review_date

name_conflicts = cleaned_data.groupby(['slug', 'review_date'])['name'].nunique()
print(name_conflicts[name_conflicts > 1])
```

```
Series([], Name: name, dtype: int64)
```

2.3 Standardize and Enrich

[46]:
```python
# Scaling the rating column

scaler = MinMaxScaler(feature_range=(0, 10))

cleaned_data['rating'] = scaler.fit_transform(cleaned_data[['rating']])

print(cleaned_data['rating'].head())
```

```
0    8.918919
1    8.378378
2    9.459459
3    8.918919
4    8.918919
Name: rating, dtype: float64
```

[47]:
```python
# Converting review_date to a datetime format and extracts the year and month

cleaned_data['review_date'] = pd.to_datetime(cleaned_data['review_date'],␣
↪errors='coerce')

cleaned_data['review_year'] = cleaned_data['review_date'].dt.year
cleaned_data['review_month'] = cleaned_data['review_date'].dt.month
```

```python
print(cleaned_data[['review_date', 'review_year', 'review_month']].head())
```

```
  review_date  review_year  review_month
0  2019-01-01         2019             1
1  2019-01-01         2019             1
2  2019-01-01         2019             1
3  2019-01-01         2019             1
4  2019-01-01         2019             1
```

[48]:
```python
# Data information

cleaned_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4887 entries, 0 to 4886
Data columns (total 21 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   slug          4887 non-null   object
 1   aroma         4887 non-null   float64
 2   body          4887 non-null   float64
 3   flavor        4887 non-null   float64
 4   clean_text    4887 non-null   object
 5   name          4887 non-null   object
 6   roaster       4887 non-null   object
 7   rating        4887 non-null   float64
 8   review_date   4887 non-null   datetime64[ns]
 9   acid          4887 non-null   float64
 10  aftertaste    4887 non-null   float64
 11  agtron        4887 non-null   object
 12  desc_1        4887 non-null   object
 13  desc_2        4887 non-null   object
 14  location      4887 non-null   object
 15  origin        4887 non-null   object
 16  roast         4887 non-null   object
 17  region        4887 non-null   object
 18  type          4887 non-null   object
 19  review_year   4887 non-null   int32
 20  review_month  4887 non-null   int32
dtypes: datetime64[ns](1), float64(6), int32(2), object(12)
memory usage: 763.7+ KB
```

[49]:
```python
# Data overview

print('\n', '='*20, 'Data Overview', '='*20, '\n')
print(cleaned_data.head())
```

```
=================== Data Overview ===================

                             slug  aroma  body  flavor  \
0         ethiopia-deri-kochoha-2    9.0   9.0     9.0   
1                     espresso-14    8.0   8.0     8.0   
2          kenya-ruthaka-peaberry    9.0   9.0    10.0   
3       ethiopia-gora-kone-sidamo    9.0   9.0     9.0   
4  specialty-coffee-blend-espresso    9.0   8.0     9.0   

                                          clean_text  \
0  bright crisp sweetli tart citru medley cacao n…   
1  evalu espresso deepli rich sweetli roast round…   
2  deepli sweet richli savori dark chocol pistach…   
3  fruit forward richli chocolati raspberri couli…   
4  evalu espresso rich chocolati sweetli tart dar…   

                             name                roaster    rating  \
0           Ethiopia Deri Kochoha      flight coffee co.  8.918919   
1                        Espresso      doi chaang coffee  8.378378   
2          Kenya Ruthaka Peaberry  temple coffee and tea  9.459459   
3       Ethiopia Gora Kone Sidamo  temple coffee and tea  8.918919   
4  Specialty Coffee Blend Espresso        choosy gourmet  8.918919   

  review_date      acid  …  agtron  \
0  2019-01-01  8.000000  …   56/80   
1  2019-01-01  7.721710  …   46/68   
2  2019-01-01  8.000000  …   48/72   
3  2019-01-01  8.000000  …   55/77   
4  2019-01-01  7.911995  …   51/75   

                                              desc_1  \
0  Bright, crisp, sweetly tart. Citrus medley, ca…   
1  Evaluated as espresso. Deeply rich, sweetly ro…   
2  Deeply sweet, richly savory. Dark chocolate, p…   
3  Fruit-forward, richly chocolaty. Raspberry cou…   
4  Evaluated as espresso. Rich, chocolaty, sweetl…   

                                              desc_2  \
0  From the Deri Kochoha mill in the Hagere Marya…   
1  Doi Chaang is a single-estate coffee produced …   
2  Despite challenges ranging from contested gove…   
3  Southern Ethiopia coffees like this one are la…   
4  A blend of coffees from Ethiopia (natural-proc…   

                           location  \
0             Bedford, New Hampshire   
1  Richmond, British Columbia, Canada   
2             Sacramento, California   
```

```
3             Sacramento, California
4                  Kaohsiung, Taiwan

                                                  origin         roast  \
0  West Guji Zone, Oromia Region, southeastern Et…  Medium-Light
1                                Northern Thailand        Medium
2          Nyeri growing region, south-central Kenya        Medium
3  Sidamo (also Sidama) growing region, south-cen…  Medium-Light
4                          Ethiopia, Colombia, Kenya  Medium-Light

         region      type review_year  review_month
0  Africa Arabia  espresso        2019             1
1   Asia Pacific  espresso        2019             1
2  Africa Arabia    estate        2019             1
3  Africa Arabia    estate        2019             1
4  Africa Arabia  espresso        2019             1

[5 rows x 21 columns]
```

[50]:

```
# Statistical summary

print('\n', '='*20, 'Statistical Summary', '='*20, '\n')
print(cleaned_data.describe(include='all'))
```

```
 ==================== Statistical Summary ====================

                           slug        aroma         body       flavor  \
count                      4887  4887.000000  4887.000000  4887.000000
unique                     4887          NaN          NaN          NaN
top     ethiopia-deri-kochoha-2          NaN          NaN          NaN
freq                          1          NaN          NaN          NaN
mean                        NaN     8.179108     7.893452     8.259975
min                         NaN     2.000000     4.000000     1.000000
25%                         NaN     8.000000     7.500000     8.000000
50%                         NaN     8.000000     8.000000     9.000000
75%                         NaN     9.000000     8.000000     9.000000
max                         NaN    10.000000    10.000000    10.000000
std                         NaN     0.997619     0.887634     1.090489

                                               clean_text  \
count                                                4887
unique                                               4886
top     light bright fragrantli smooth hot aliv shimme…
freq                                                    2
mean                                                  NaN
min                                                   NaN
25%                                                   NaN
```

```
50%                                                     NaN
75%                                                     NaN
max                                                     NaN
std                                                     NaN

                       name             roaster       rating  \
count                  4887                4887  4887.000000
unique                 4071                1120          NaN
top     Ethiopia Yirgacheffe  jbc coffee roasters          NaN
freq                     25                 178          NaN
mean                    NaN                 NaN     8.124802
min                     NaN                 NaN     0.000000
25%                     NaN                 NaN     7.702703
50%                     NaN                 NaN     8.378378
75%                     NaN                 NaN     8.918919
max                     NaN                 NaN    10.000000
std                     NaN                 NaN     1.209412

                          review_date         acid  …  agtron  \
count                            4887  4887.000000  …    4887
unique                            NaN          NaN  …    1045
top                               NaN          NaN  …       /
freq                              NaN          NaN  …     297
mean    2010-09-15 23:49:23.535911680     7.700424  …     NaN
min               1997-02-01 00:00:00     2.000000  …     NaN
25%               2007-02-01 00:00:00     7.000000  …     NaN
50%               2011-10-01 00:00:00     8.000000  …     NaN
75%               2015-06-01 00:00:00     8.000000  …     NaN
max               2019-01-01 00:00:00    10.000000  …     NaN
std                               NaN     0.913070  …     NaN

                                                 desc_1  \
count                                              4887
unique                                             4887
top     Bright, crisp, sweetly tart. Citrus medley, ca…
freq                                                  1
mean                                                NaN
min                                                 NaN
25%                                                 NaN
50%                                                 NaN
75%                                                 NaN
max                                                 NaN
std                                                 NaN

                                                 desc_2            location  \
count                                              4887                4887
unique                                             4683                 711
top     Paradise Roasters prides itself on roasting an…  Madison, Wisconsin
```

```
freq                                                      7                  198
mean                                                    NaN                  NaN
min                                                     NaN                  NaN
25%                                                     NaN                  NaN
50%                                                     NaN                  NaN
75%                                                     NaN                  NaN
max                                                     NaN                  NaN
std                                                     NaN                  NaN

         origin         roast         region    type  review_year  \
count      4887          4887           4887    4887  4887.000000
unique     1434             6              7       9          NaN
top     Unknown  Medium-Light  Africa Arabia  estate          NaN
freq        553          1878           1914    1445          NaN
mean        NaN           NaN            NaN     NaN  2010.232658
min         NaN           NaN            NaN     NaN  1997.000000
25%         NaN           NaN            NaN     NaN  2007.000000
50%         NaN           NaN            NaN     NaN  2011.000000
75%         NaN           NaN            NaN     NaN  2015.000000
max         NaN           NaN            NaN     NaN  2019.000000
std         NaN           NaN            NaN     NaN     5.768924

        review_month
count    4887.000000
unique           NaN
top              NaN
freq             NaN
mean        6.710661
min         1.000000
25%         4.000000
50%         7.000000
75%        10.000000
max        12.000000
std         3.502846

[11 rows x 21 columns]
```

[51]:
```python
# Saving cleaned_data as a CSV file

cleaned_data.to_csv('cleaned_data.csv')
```

2.4 Summary of Data Cleaning

The `slug` field in `coffee_raw` was standardized by removing redundant prefixes to align with the formatting used in `coffee_clean` and `coffee_id`. An inner join was then performed between `coffee_clean` and `coffee_id` on the slug field to retain only matched records. To enrich the dataset with additional context, selected extra columns from `coffee_raw` were merged using a left join, ensuring the completeness of existing entries.

Several columns were dropped due to irrelevance or high missing values, including `all_text`, `desc_3`, `desc_4`, `est_price`, and `with_milk`. Duplicate columns `type_with_milk` and `type_with_milk.1` were reviewed, confirmed to be identical, and one was removed.

The dataset initially used one-hot encoding for regional indicators. These were consolidated into a single `region` column. Unknown values were partially resolved through keyword detection in the `origin` field (e.g., matching "Morocco" to "Africa Arabia"). As a further refinement step, remaining unknown values were imputed based on the most frequent region associated with each `roaster` or `location`. The original one-hot encoded region columns were then dropped.

Inconsistent roast names were standardized (e.g., converting `Very Dark` to `Very-Dark`), and missing values were filled with `Unknown`. Later, unknown roast values were inferred using the most frequent roast type within each region. The original one-hot encoded roast columns were subsequently removed.

A similar process was applied to coffee type classification. One-hot encoded type indicators were mapped into a single `type` column. Unknown values were addressed through keyword matching from the combined `desc_1` and `desc_2` text fields. For example, espresso-related keywords included terms like espresso, dark roast, and ristretto. Remaining unmatched entries were labeled as `general coffee`. All one-hot encoded type columns were then dropped, along with the redundant `acid_or_milk column`.

Missing values in `origin` and `location` were filled with `Unknown`. For the numeric fields `aftertaste` and `acid`, missing values were imputed using the mean within each roast group.

The `rating` field was converted to numeric format, and missing or invalid entries such as `"NR"` were replaced with the mean rating for each roaster. Ratings were then scaled to a 0–10 range using MinMaxScaler to enable uniform comparison.

The `review_date` field was parsed into proper datetime format. From this, `review_year` and `review_month` columns were extracted to support time-based analysis.

Finally, a duplicate check confirmed there were no repeated rows based on the composite key of `slug` and `review_date`.

2.5 Data Overview

`slug`: Unique identifier for each coffee entry.

`aroma`: Aroma score of the coffee (1–10).

`body`: Body score (1–10).

`flavor`: Flavor score (1–10).

`clean_text`: Preprocessed version of the review text for analysis.

`name`: Coffee product name.

`roaster`: Name of the roasting company.

`rating`: Overall expert rating (scaled 0–10).

`review_date`: Full date of the review (1997–2019).

`acid`: Acidity score (1–10).

`aftertaste`: Aftertaste score (1–10).

`agtron`: Roast classification based on color (Agtron scale).

`desc_1`: First part of the original text description

`desc_2`: Second part of the original text description

`location`: Roaster's geographic location.

`origin`: Country or region of the coffee bean.

`region`: Standardized broader region (e.g., "Africa Arabia").

`roast`: Coffee roast level (e.g., Medium-Light, Dark).

`type`: Coffee type category (e.g., Espresso, Organic, Blend).

`review_year`: Year extracted from review_date.

`review_month`: Month extracted from review_date.

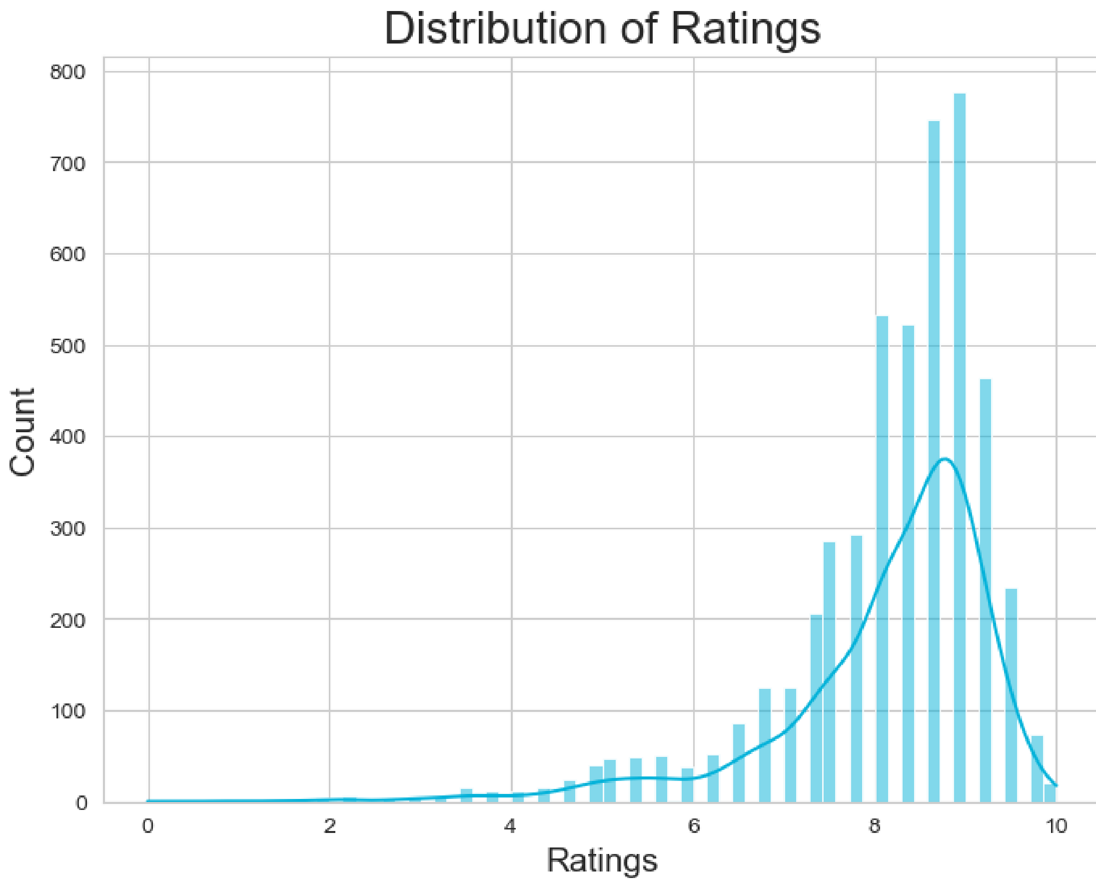
3 Exploratory Data Analysis

[52]:
```
# Load the data

df = pd.read_csv('cleaned_data.csv')
```

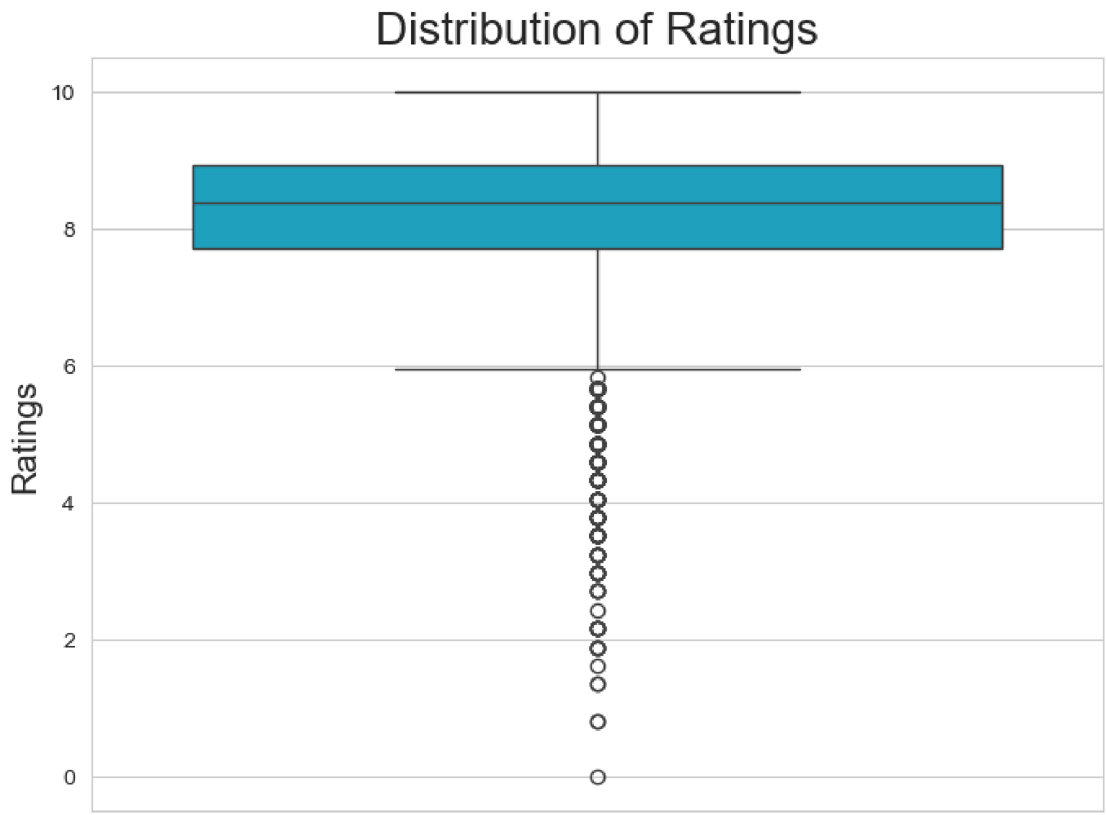
[53]:
```
# Set palette

palette = ["#04B2D9", "#4AA2D9", "#2B96D9", "#0367A6", "#034C8C", "#02386F"]
sns.set_palette(palette)

# Set style

sns.set_style('whitegrid')
```

3.1 Ratings Distribution

[54]:
```
# Distribution of ratings with histogram

plt.figure(figsize=(8,6))
sns.histplot(df['rating'], kde=True)
plt.xlabel('Ratings', fontsize=15)
plt.ylabel('Count', fontsize=15)
plt.title('Distribution of Ratings', fontsize=20)
plt.show()
```

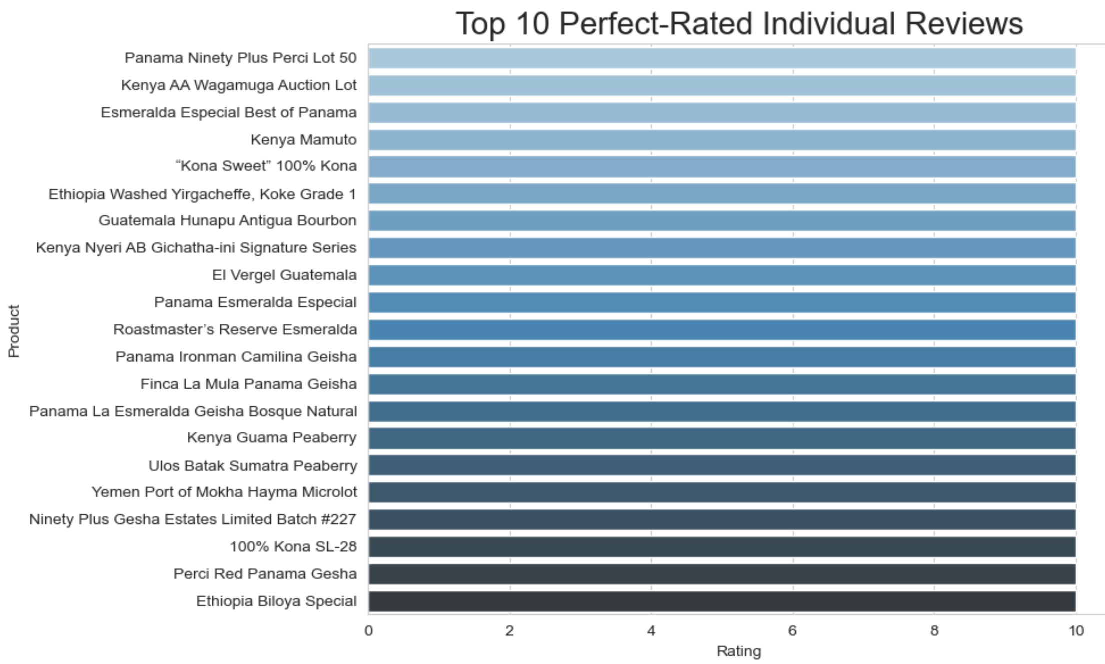


[55]:
```
# Distribution of ratings with boxplot

plt.figure(figsize=(8,6))
sns.boxplot(df['rating'])
plt.ylabel('Ratings', fontsize=15)
plt.title('Distribution of Ratings', fontsize=20)
plt.show()
```

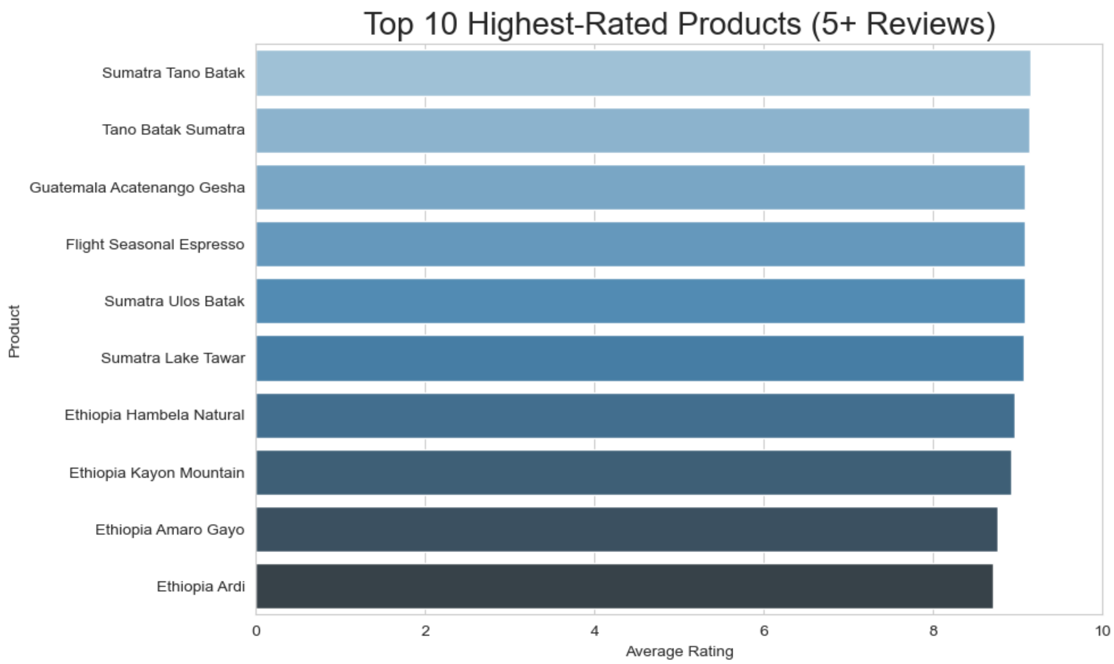


3.2 Top-Rated Products

[56]:
```
# Display individual products with a perfect rating of 10

top_reviews = df[df['rating'] == 10].sort_values(by='rating')

plt.figure(figsize=(10, 6))
sns.barplot(data=top_reviews, x='rating', y='name', palette="Blues_d")
plt.xlabel('Rating')
plt.ylabel('Product')
plt.title('Top 10 Perfect-Rated Individual Reviews', fontsize=20)

plt.tight_layout()
plt.show()
```



[57]:
```
# Display the top 10 products with the highest average rating among those with␣
↪at least 5 reviews

top_products = df.groupby('name').agg(
    avg_rating=('rating', 'mean'),
    review_count=('rating', 'count')
)

top_products = top_products[top_products['review_count'] >= 5]

top_products = top_products.sort_values(by='avg_rating', ascending=False).
↪head(10).reset_index()

plt.figure(figsize=(10, 6))
sns.barplot(data=top_products, x='avg_rating', y='name', palette="Blues_d")
plt.xlabel('Average Rating')
plt.ylabel('Product')
plt.title('Top 10 Highest-Rated Products (5+ Reviews)', fontsize=20)
plt.xlim(0, 10)
plt.tight_layout()
plt.show()
```

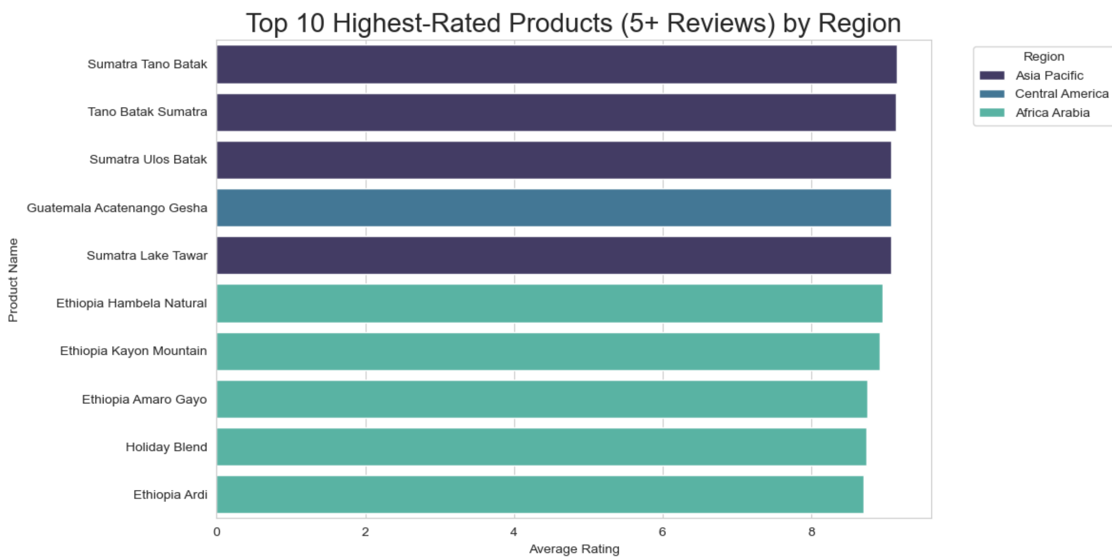


[58]:
```
top_products
```

[58]:
```
                         name  avg_rating  review_count
0           Sumatra Tano Batak    9.150579             7
1           Tano Batak Sumatra    9.135135             5
2   Guatemala Acatenango Gesha    9.081081             5
3     Flight Seasonal Espresso    9.081081             5
4           Sumatra Ulos Batak    9.081081             5
5           Sumatra Lake Tawar    9.073359             7
6     Ethiopia Hambela Natural    8.957529             7
7      Ethiopia Kayon Mountain    8.918919             5
8          Ethiopia Amaro Gayo    8.756757             5
9                Ethiopia Ardi    8.702703             5
```

[59]:
```
# Display the top 10 highest-rated products with at least 5 reviews, broken␣
↪down by region

product_stats = (
    df.groupby(['name', 'region'])
    .agg(avg_rating=('rating', 'mean'), count=('rating', 'count'))
    .query('count >= 5')
    .sort_values(by='avg_rating', ascending=False)
    .reset_index()
)
```

```python
top10_products = product_stats.head(10).reset_index()

plt.figure(figsize=(12,6))
sns.barplot( data=top10_products, x='avg_rating', y='name', hue='region', dodge=False, palette='mako')
plt.title('Top 10 Highest-Rated Products (5+ Reviews) by Region', fontsize=20)
plt.xlabel('Average Rating')
plt.ylabel('Product Name')
plt.legend(title='Region', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



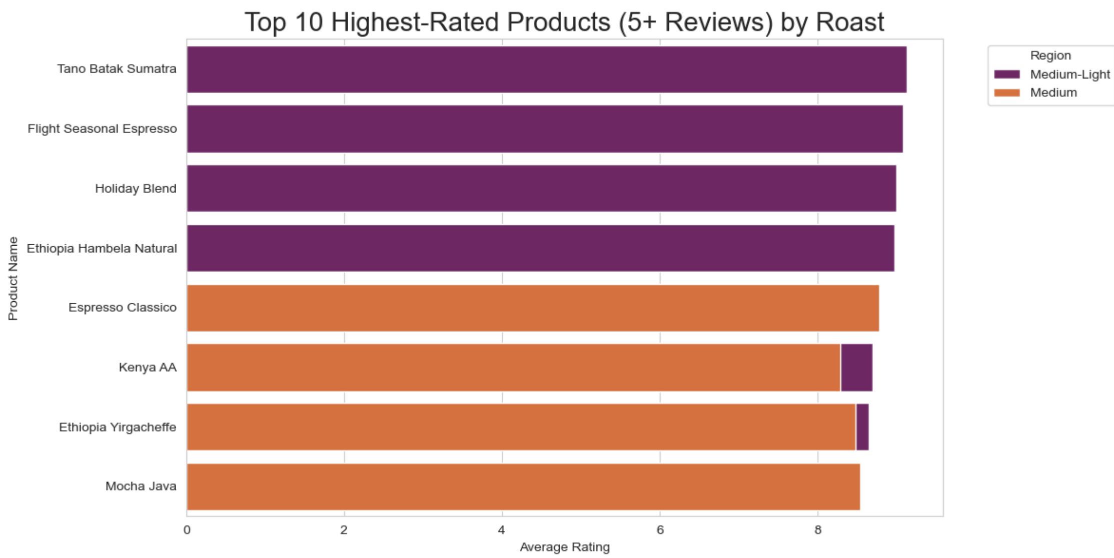
[60]:
```python
product_stats.head(10)
```

[60]:

	name	region	avg_rating	count
0	Sumatra Tano Batak	Asia Pacific	9.150579	7
1	Tano Batak Sumatra	Asia Pacific	9.135135	5
2	Sumatra Ulos Batak	Asia Pacific	9.081081	5
3	Guatemala Acatenango Gesha	Central America	9.081081	5
4	Sumatra Lake Tawar	Asia Pacific	9.073359	7
5	Ethiopia Hambela Natural	Africa Arabia	8.957529	7
6	Ethiopia Kayon Mountain	Africa Arabia	8.918919	5
7	Ethiopia Amaro Gayo	Africa Arabia	8.756757	5
8	Holiday Blend	Africa Arabia	8.744038	17
9	Ethiopia Ardi	Africa Arabia	8.702703	5

[61]:
```python
# Display the top 10 highest-rated products with at least 5 reviews, broken down by roast
```

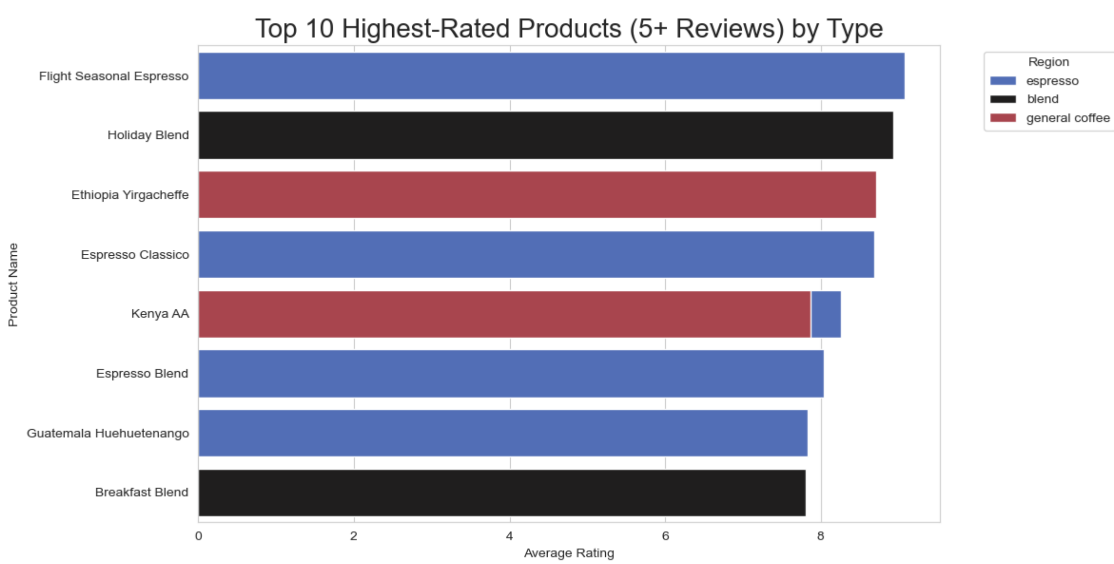
```python
product_stats = (
    df.groupby(['name', 'roast'])
    .agg(avg_rating=('rating', 'mean'), count=('rating', 'count'))
    .query('count >= 5')
    .sort_values(by='avg_rating', ascending=False)
    .reset_index()
)

top10_products = product_stats.head(10)

plt.figure(figsize=(12,6))
sns.barplot(data=top10_products, x='avg_rating', y='name', hue='roast',␣
 ↪dodge=False, palette='inferno')
plt.title('Top 10 Highest-Rated Products (5+ Reviews) by Roast', fontsize=20)
plt.xlabel('Average Rating')
plt.ylabel('Product Name')
plt.legend(title='Region', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



[62]:
```python
product_stats = (
    df.groupby(['name', 'roast'])
    .agg(avg_rating=('rating', 'mean'), count=('rating', 'count'))
    .query('count >= 5')
    .sort_values(by='avg_rating', ascending=False)
    .reset_index()
)
```

```
top10_products = product_stats.head(10)
```

[63]:
```
# Display the top 10 highest-rated products with at least 5 reviews, broken␣
↪down by type


product_stats = (
    df.groupby(['name', 'type'])
    .agg(avg_rating=('rating', 'mean'), count=('rating', 'count'))
    .query('count >= 5')
    .sort_values(by='avg_rating', ascending=False)
    .reset_index()
)

top10_products = product_stats.head(10)

plt.figure(figsize=(12,6))
sns.barplot(data=top10_products, x='avg_rating', y='name', hue='type',␣
↪dodge=False, palette='icefire')
plt.title('Top 10 Highest-Rated Products (5+ Reviews) by Type', fontsize=20)
plt.xlabel('Average Rating')
plt.ylabel('Product Name')
plt.legend(title='Region', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



[64]:
```
top10_products
```

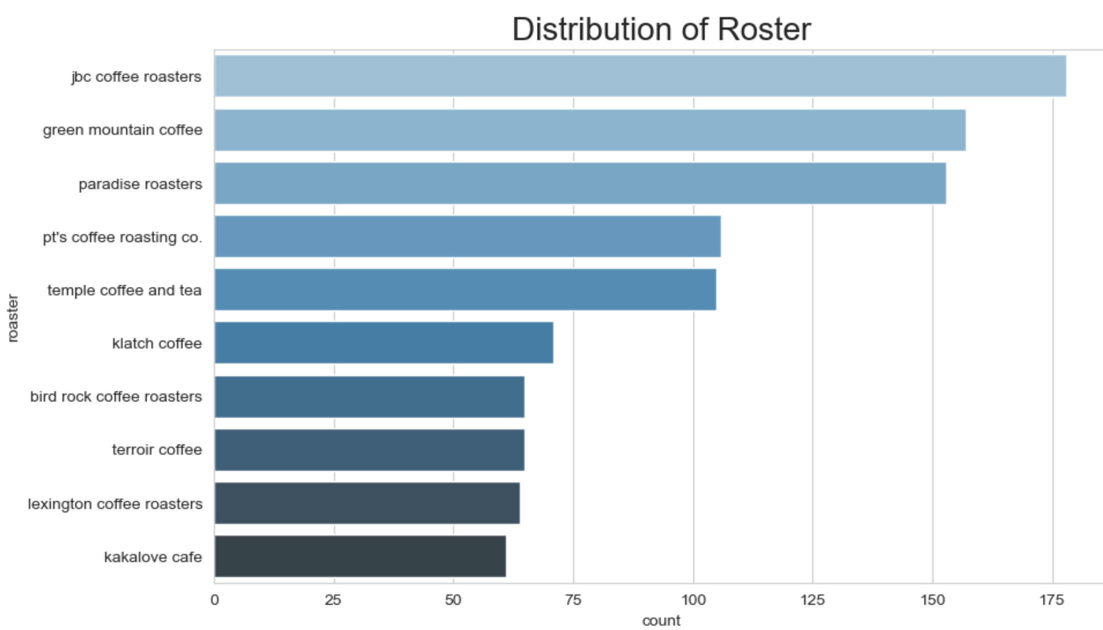
[64]:

	name	type	avg_rating	count
0	Flight Seasonal Espresso	espresso	9.081081	5
1	Holiday Blend	blend	8.934817	17
2	Ethiopia Yirgacheffe	general coffee	8.708709	9
3	Espresso Classico	espresso	8.687259	7
4	Kenya AA	espresso	8.262548	7
5	Ethiopia Yirgacheffe	espresso	8.243243	8
6	Espresso Blend	espresso	8.036036	15
7	Kenya AA	general coffee	7.867868	9
8	Guatemala Huehuetenango	espresso	7.837838	5
9	Breakfast Blend	blend	7.807808	9

3.3 Roaster Analysis

[65]:

```
# Display the 10 most frequently reviewed roasters

plt.figure(figsize=(10,6))
sns.countplot(data=df, y='roaster', order=df['roaster'].value_counts().
↪nlargest(10).index, palette="Blues_d")
plt.title('Distribution of Roster', fontsize=20)
plt.show()
```



[66]:

```
df['roaster'].value_counts().nlargest(10)
```

[66]:

```
roaster
jbc coffee roasters          178
```

```
green mountain coffee        157
paradise roasters            153
pt's coffee roasting co.     106
temple coffee and tea        105
klatch coffee                 71
bird rock coffee roasters     65
terroir coffee                65
lexington coffee roasters     64
kakalove cafe                 61
Name: count, dtype: int64
```

[67]:

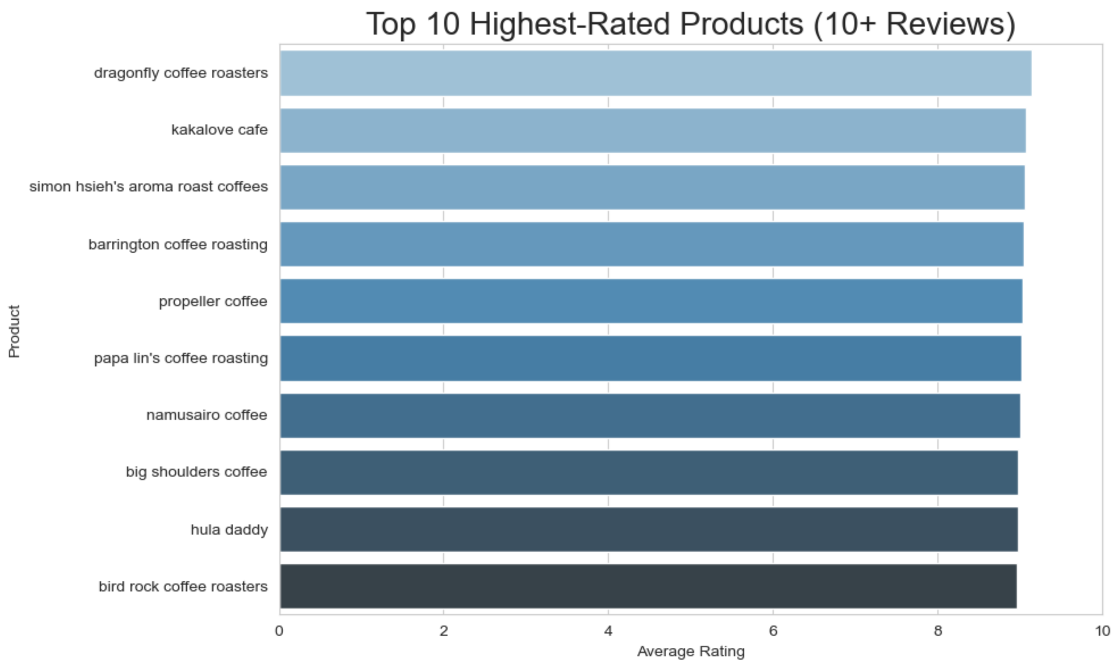
```
top_products
```

[67]:

```
                         name  avg_rating  review_count
0           Sumatra Tano Batak    9.150579             7
1           Tano Batak Sumatra    9.135135             5
2   Guatemala Acatenango Gesha    9.081081             5
3     Flight Seasonal Espresso    9.081081             5
4           Sumatra Ulos Batak    9.081081             5
5           Sumatra Lake Tawar    9.073359             7
6     Ethiopia Hambela Natural    8.957529             7
7      Ethiopia Kayon Mountain    8.918919             5
8          Ethiopia Amaro Gayo    8.756757             5
9                Ethiopia Ardi    8.702703             5
```

[68]:

```
# Display the top 10 roasters with the highest average rating among those with␣
↪at least 10 reviews

top_products = df.groupby('roaster').agg(
    avg_rating=('rating', 'mean'),
    review_count=('rating', 'count')
)

top_products = top_products[top_products['review_count'] >= 10]

top_products = top_products.sort_values(by='avg_rating', ascending=False).
↪head(10).reset_index()

plt.figure(figsize=(10, 6))
sns.barplot(data=top_products, x='avg_rating', y='roaster', palette='Blues_d')
plt.xlabel('Average Rating')
plt.ylabel('Product')
plt.title('Top 10 Highest-Rated Products (10+ Reviews)', fontsize=20)
plt.xlim(0, 10)
plt.tight_layout()
plt.show()
```



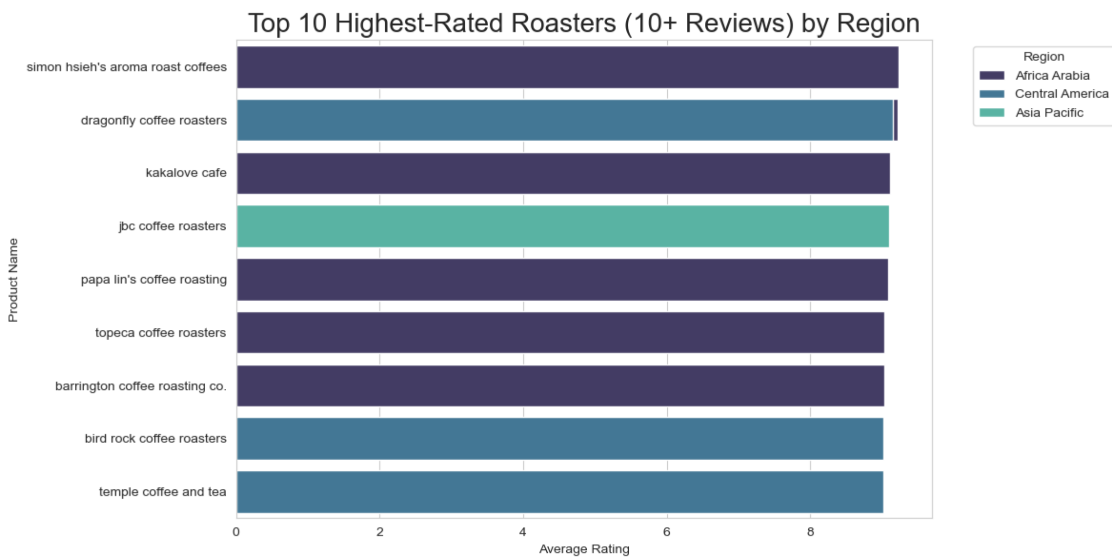
[69]:
```python
top_products
```

[69]:

	roaster	avg_rating	review_count
0	dragonfly coffee roasters	9.141494	51
1	kakalove cafe	9.078423	61
2	simon hsieh's aroma roast coffees	9.062328	49
3	barrington coffee roasting	9.041769	11
4	propeller coffee	9.027027	10
5	papa lin's coffee roasting	9.013514	20
6	namusairo coffee	9.000000	10
7	big shoulders coffee	8.978979	18
8	hula daddy	8.975818	19
9	bird rock coffee roasters	8.964657	65

[70]:
```python
# Display the top 10 highest-rated roasters with at least 10 reviews, broken␣
↪down by region

product_stats = (
    df.groupby(['roaster', 'region'])
    .agg(avg_rating=('rating', 'mean'), count=('rating', 'count'))
    .query('count >= 10')
    .sort_values(by='avg_rating', ascending=False)
    .reset_index()
)
```

```
top10_products = product_stats.head(10)

plt.figure(figsize=(12,6))
sns.barplot(data=top10_products, x='avg_rating', y='roaster', hue='region',
    dodge=False, palette='mako')
plt.title('Top 10 Highest-Rated Roasters (10+ Reviews) by Region', fontsize=20)
plt.xlabel('Average Rating')
plt.ylabel('Product Name')
plt.legend(title='Region', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



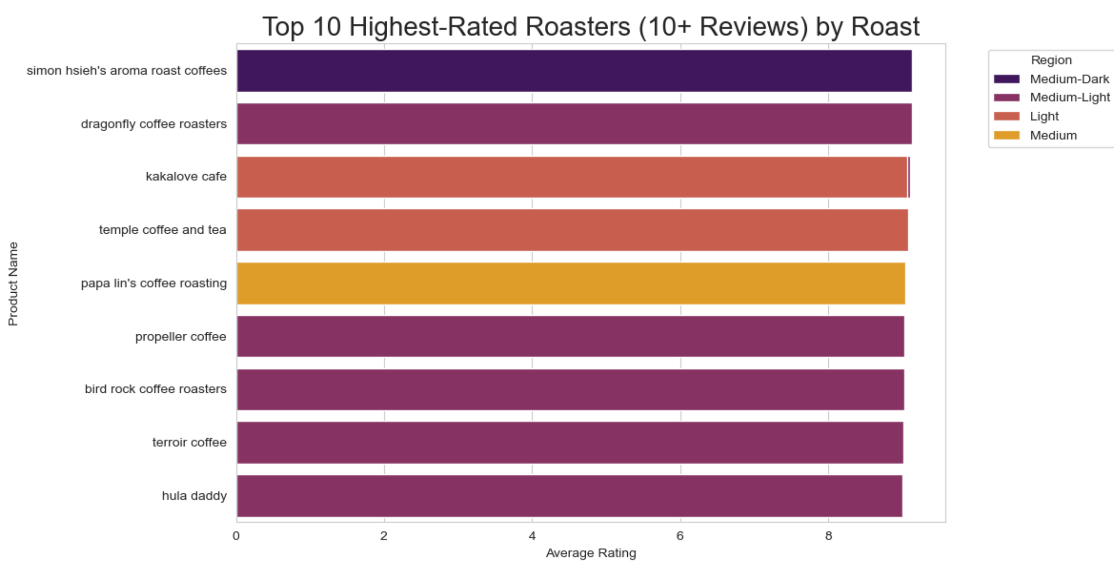
[71]:
```
product_stats.head(10)
```

[71]:

	roaster	region	avg_rating	count
0	simon hsieh's aroma roast coffees	Africa Arabia	9.229229	27
1	dragonfly coffee roasters	Africa Arabia	9.216216	20
2	dragonfly coffee roasters	Central America	9.156757	25
3	kakalove cafe	Africa Arabia	9.112808	46
4	jbc coffee roasters	Asia Pacific	9.099099	18
5	papa lin's coffee roasting	Africa Arabia	9.085239	13
6	topeca coffee roasters	Africa Arabia	9.027027	10
7	barrington coffee roasting co.	Africa Arabia	9.027027	25
8	bird rock coffee roasters	Central America	9.016216	25
9	temple coffee and tea	Central America	9.015015	45

[72]:
```
# Display the top 10 highest-rated roasters with at least 10 reviews, broken
    down by roast
```

```python
product_stats = (
    df.groupby(['roaster', 'roast'])
    .agg(avg_rating=('rating', 'mean'), count=('rating', 'count'))
    .query('count >= 10')
    .sort_values(by='avg_rating', ascending=False)
    .reset_index()
)

top10_products = product_stats.head(10)

plt.figure(figsize=(12,6))
sns.barplot(data=top10_products, x='avg_rating', y='roaster', hue='roast', dodge=False, palette='inferno')
plt.title('Top 10 Highest-Rated Roasters (10+ Reviews) by Roast', fontsize=20)
plt.xlabel('Average Rating')
plt.ylabel('Product Name')
plt.legend(title='Region', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



[73]:
```python
product_stats.head(10)
```

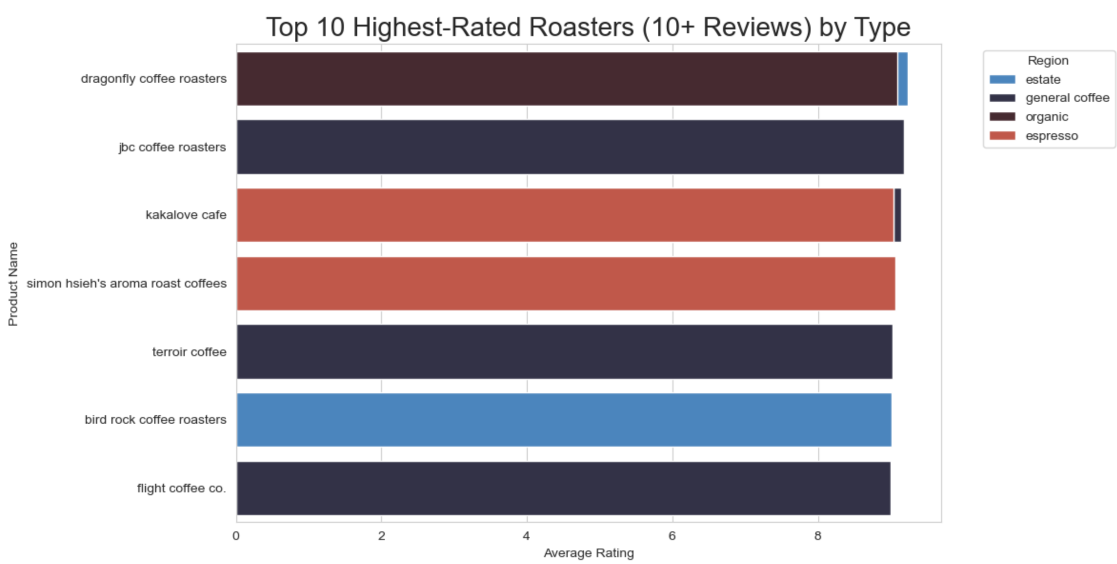
[73]:

	roaster	roast	avg_rating	count
0	simon hsieh's aroma roast coffees	Medium-Dark	9.131274	14
1	dragonfly coffee roasters	Medium-Light	9.130752	37
2	kakalove cafe	Medium-Light	9.101351	40
3	temple coffee and tea	Light	9.081081	15

```
4                  kakalove cafe         Light   9.063063     15
5       papa lin's coffee roasting        Medium   9.043659     13
6                 propeller coffee  Medium-Light   9.027027     10
7          bird rock coffee roasters  Medium-Light   9.025770     43
8                   terroir coffee  Medium-Light   9.011583     35
9                       hula daddy  Medium-Light   9.009009     12
```

[74]:

```python
# Display the top 10 highest-rated roasters with at least 10 reviews, broken␣
↪down by type

product_stats = (
    df.groupby(['roaster', 'type'])
    .agg(avg_rating=('rating', 'mean'), count=('rating', 'count'))
    .query('count >= 10')
    .sort_values(by='avg_rating', ascending=False)
    .reset_index()
)

top10_products = product_stats.head(10)

plt.figure(figsize=(12,6))
sns.barplot(data=top10_products, x='avg_rating', y='roaster', hue='type',␣
↪dodge=False, palette='icefire')
plt.title('Top 10 Highest-Rated Roasters (10+ Reviews) by Type', fontsize=20)
plt.xlabel('Average Rating')
plt.ylabel('Product Name')
plt.legend(title='Region', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



[75]:
```
product_stats.head(10)
```

[75]:

	roaster	type	avg_rating	count
0	dragonfly coffee roasters	estate	9.236193	23
1	jbc coffee roasters	general coffee	9.189189	22
2	kakalove cafe	general coffee	9.145946	25
3	dragonfly coffee roasters	organic	9.090909	11
4	simon hsieh's aroma roast coffees	espresso	9.062328	49
5	kakalove cafe	espresso	9.041769	11
6	kakalove cafe	organic	9.039039	18
7	terroir coffee	general coffee	9.027027	10
8	bird rock coffee roasters	estate	9.009009	24
9	flight coffee co.	general coffee	8.996139	14

3.4 Seasonal Trends

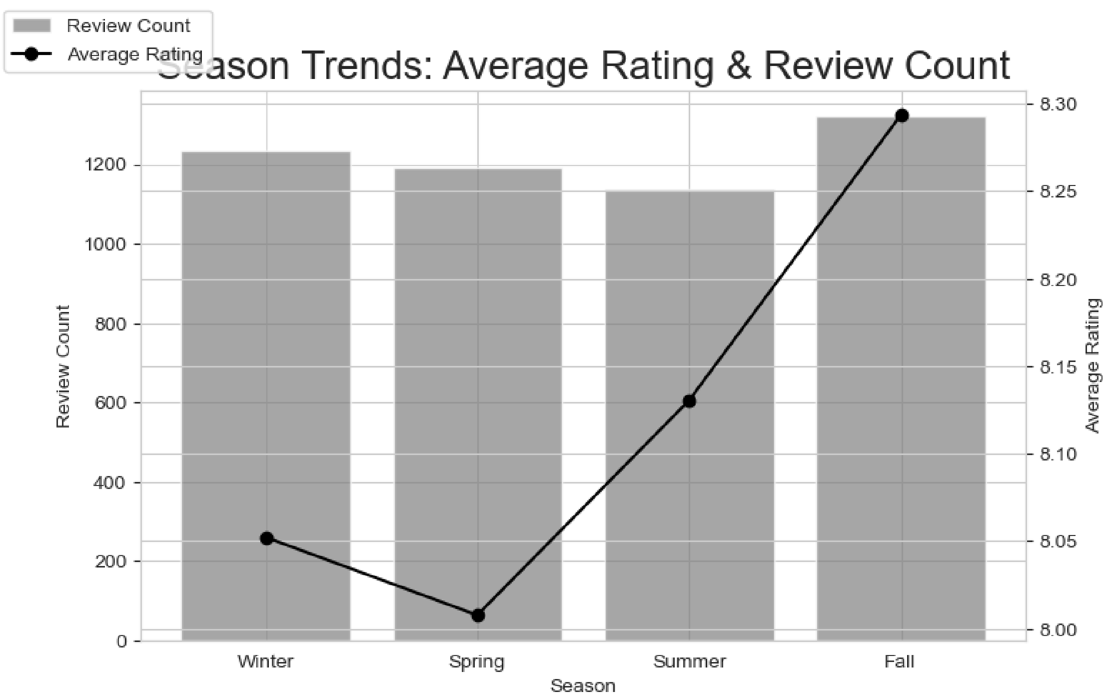
[76]:
```
# Analyze seasonal trends by calculating average rating and review count for␣
↪each season

def get_season(month):
    if month in [12, 1, 2]:
        return 'Winter'
    elif month in [3, 4, 5]:
        return 'Spring'
    elif month in [6, 7, 8]:
        return 'Summer'
    else:
        return 'Fall'

df['season'] = df['review_month'].apply(get_season)

seasonal_trends = df.groupby('season').agg(
    avg_rating=('rating', 'mean'),
    review_count=('rating', 'count')
).reindex(['Winter', 'Spring', 'Summer', 'Fall'])

seasonal_trends
```

[76]:

season	avg_rating	review_count
Winter	8.052258	1234
Spring	8.007748	1193
Summer	8.130413	1139
Fall	8.293441	1321

[77]:

```
# Visualize seasonal trends by plotting review count as bars and average rating
↪as a line

fig, ax1 = plt.subplots(figsize=(8, 5))
ax1.bar(seasonal_trends.index, seasonal_trends['review_count'], color='grey',
↪alpha=0.7, label='Review Count')
ax1.set_ylabel('Review Count')
ax1.set_xlabel('Season')
ax2 = ax1.twinx()
ax2.plot(seasonal_trends.index, seasonal_trends['avg_rating'], marker='o',
↪color='black', label='Average Rating')
ax2.set_ylabel('Average Rating')
ax2.set_xlabel('Season')

plt.title('Season Trends: Average Rating & Review Count', fontsize=20)

fig.legend(loc='upper left')
plt.show()
```



3.5 Long-Term Trends

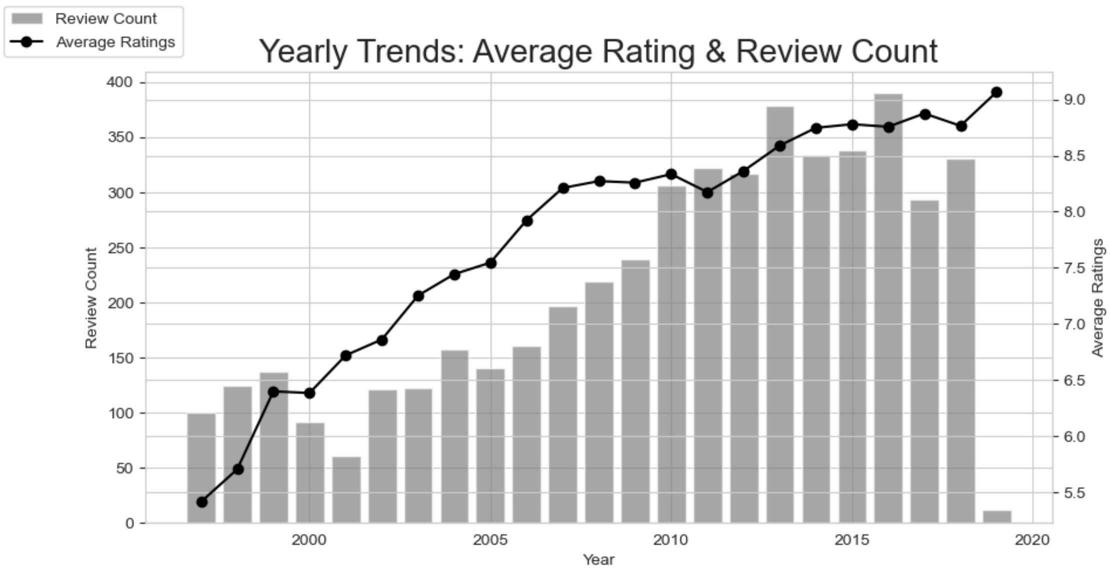
[78]:
```
# Analyze yearly trends by calculating average rating and review count for each␣
↪year

yearly_trends = df.groupby('review_year').agg(
    avg_rating=('rating', 'mean'),
    review_count=('rating', 'count')
)

yearly_trends
```

[78]:

review_year	avg_rating	review_count
1997	5.416216	100
1998	5.705863	124
1999	6.399684	137
2000	6.385506	91
2001	6.720721	60
2002	6.861738	121
2003	7.252991	122
2004	7.443622	157
2005	7.546332	140
2006	7.923986	160
2007	8.211003	197
2008	8.272245	219
2009	8.258510	239
2010	8.334217	306
2011	8.174417	322
2012	8.363032	317
2013	8.588589	378
2014	8.746561	334
2015	8.778986	338
2016	8.756064	390
2017	8.874643	293
2018	8.764595	331
2019	9.066339	11

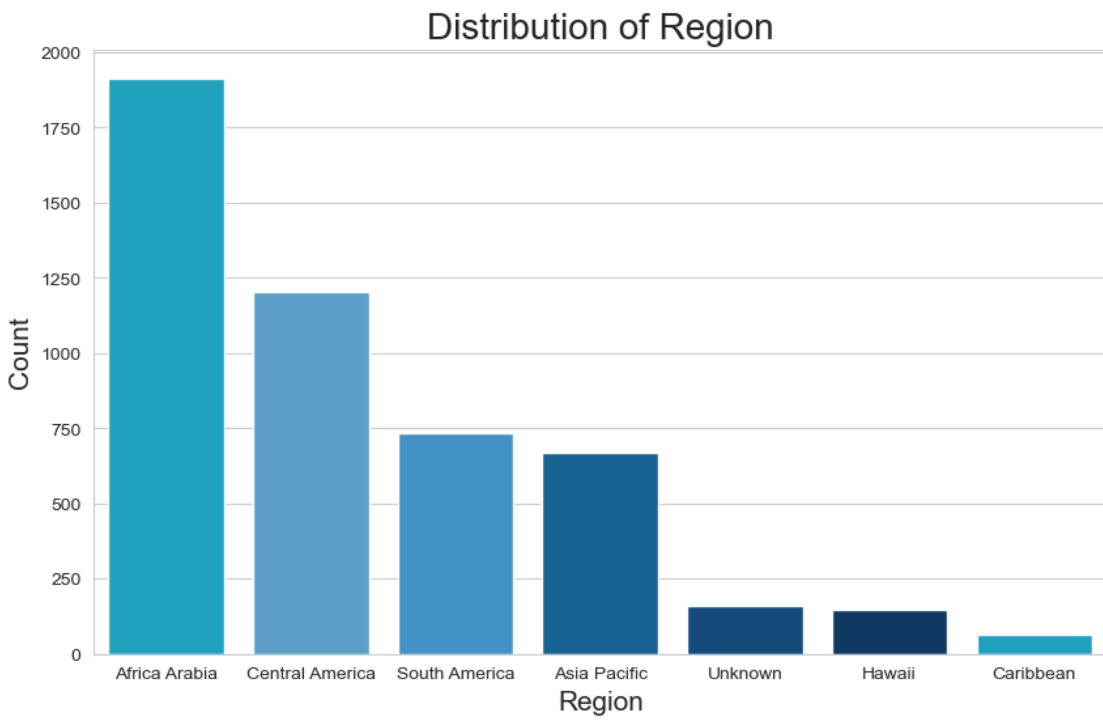
[79]:
```
# Visualize yearly trends by plotting review count as bars and average rating␣
↪as a line

fig, ax1 = plt.subplots(figsize=(10, 5))

ax1.bar(yearly_trends.index, yearly_trends['review_count'], color='grey',␣
↪alpha=0.7, label='Review Count')
ax1.set_ylabel('Review Count')
ax1.set_xlabel('Year')
```

```
ax2 = ax1.twinx()
ax2.plot(yearly_trends.index, yearly_trends['avg_rating'], color='black',␣
    ↪label='Average Ratings', marker='o')
ax2.set_ylabel('Average Ratings')
plt.title('Yearly Trends: Average Rating & Review Count', fontsize=20)

fig.legend(loc='upper left')
plt.show()
```



3.6 Regional Performance

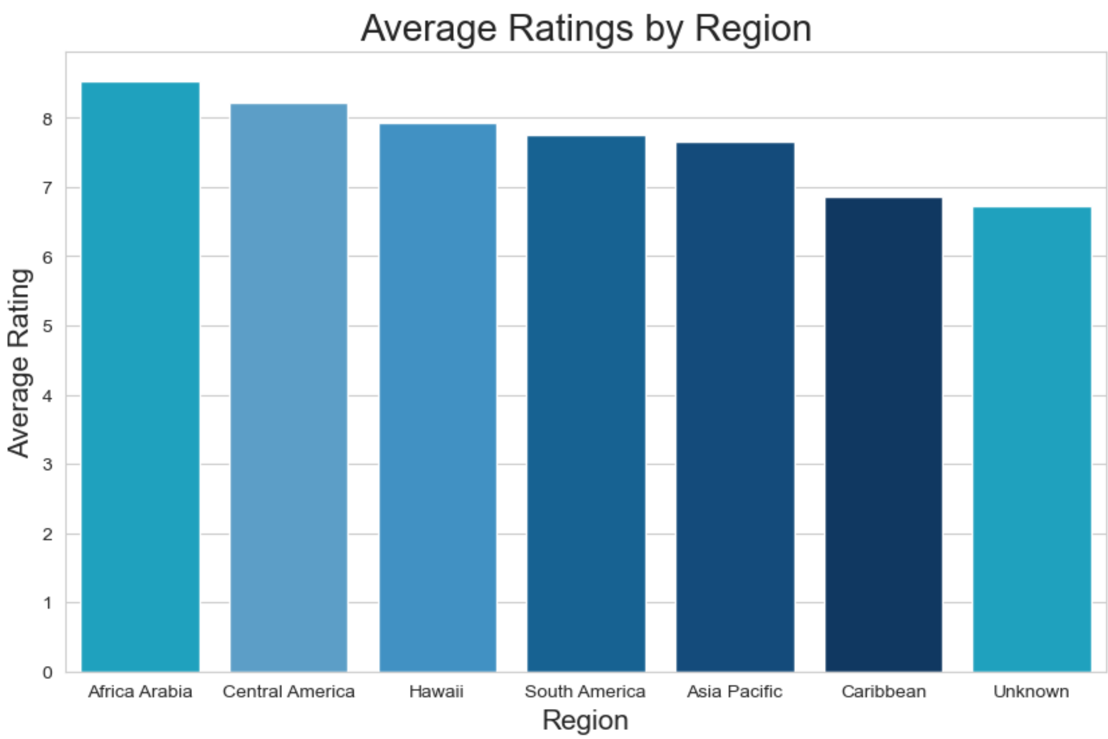
[80]:
```
# Distribution of region

plt.figure(figsize=(10,6))
sns.countplot(data=df, x='region', order=df['region'].value_counts().index,␣
    ↪palette=palette)
plt.xlabel('Region', fontsize=15)
plt.ylabel('Count', fontsize=15)
plt.title('Distribution of Region', fontsize=20)
plt.show()
```



[81]:
```
df['region'].value_counts()
```

[81]:
```
region
Africa Arabia      1914
Central America    1204
South America       733
Asia Pacific        669
Unknown             158
Hawaii              145
Caribbean            64
Name: count, dtype: int64
```

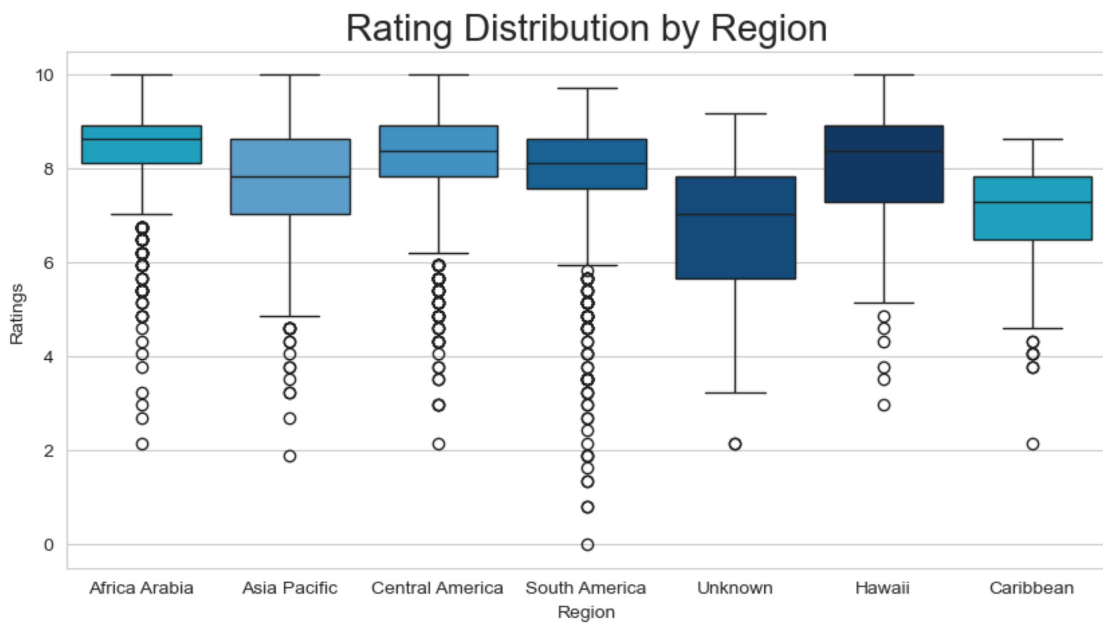
[82]:
```
# Display the average ratings by region

region_ratings = df.groupby('region')['rating'].mean().
 ↪sort_values(ascending=False)

plt.figure(figsize=(10,6))
sns.barplot(x=region_ratings.index, y=region_ratings.values, palette=palette)
plt.xlabel('Region', fontsize=15)
plt.ylabel('Average Rating', fontsize=15)
plt.title('Average Ratings by Region', fontsize=20)
plt.show()
```



[83]:
```
region_ratings
```

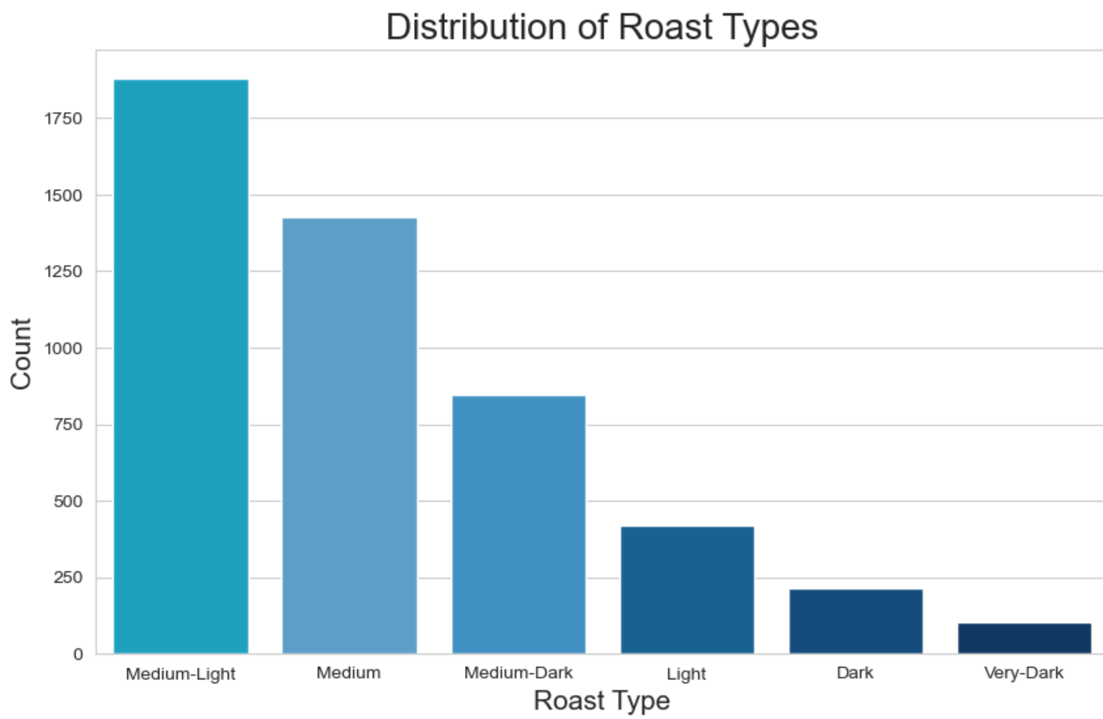
[83]:
```
region
Africa Arabia      8.536107
Central America    8.223489
Hawaii             7.919851
South America      7.763578
Asia Pacific       7.660991
Caribbean          6.853885
Unknown            6.732809
Name: rating, dtype: float64
```

[84]:
```
# Display the average ratings by region with boxplot

plt.figure(figsize=(10,5))
sns.boxplot(data=df, x='region', y='rating', palette=palette)
plt.xlabel('Region')
plt.ylabel('Ratings')
plt.title('Rating Distribution by Region', fontsize=20)
plt.show()
```



3.7 Roast Types

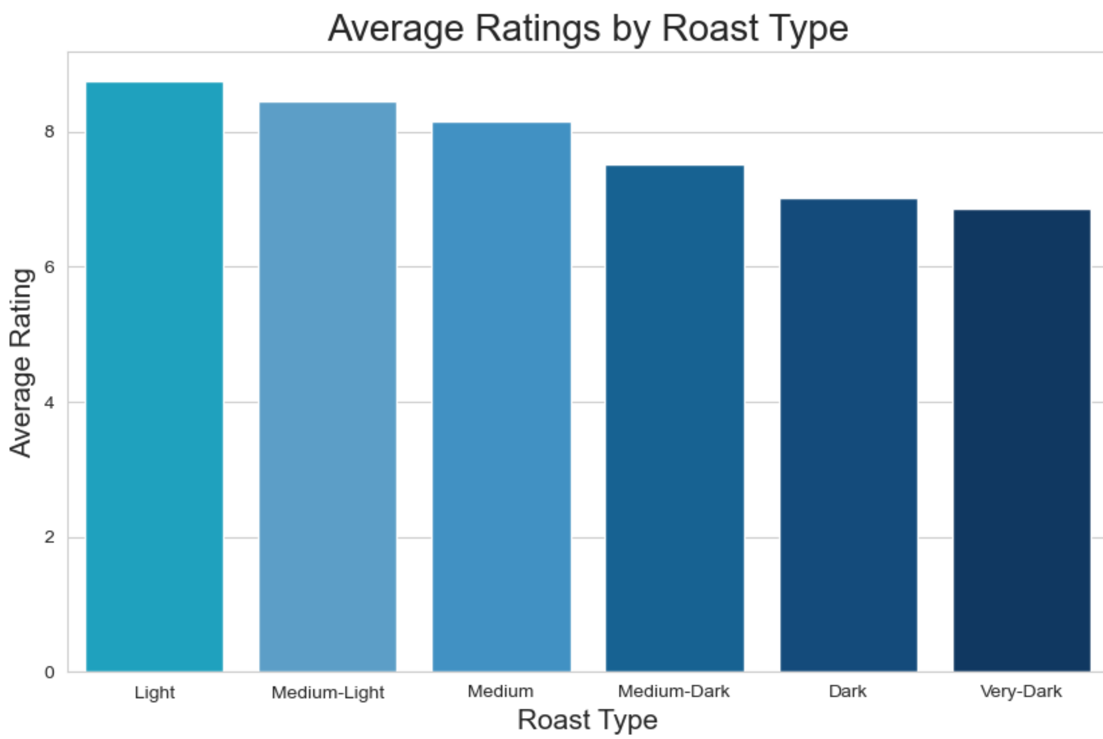
[85]:
```
# Distribution of roast types

plt.figure(figsize=(10,6))
sns.countplot(data=df, x='roast', order=df['roast'].value_counts().index,␣
↪palette=palette)
plt.xlabel('Roast Type', fontsize=15)
plt.ylabel('Count', fontsize=15)
plt.title('Distribution of Roast Types', fontsize=20)
plt.show()
```



[86]:
```
df['roast'].value_counts()
```

[86]:
```
roast
Medium-Light    1878
Medium          1427
Medium-Dark      845
Light            418
Dark             216
Very-Dark        103
Name: count, dtype: int64
```

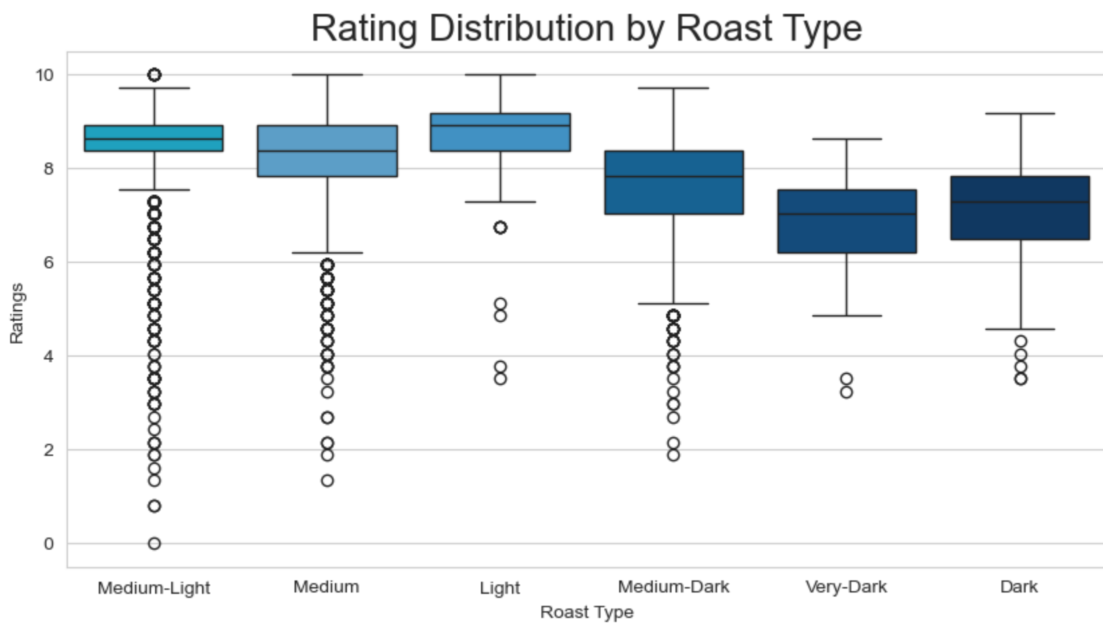
[87]:
```
# Display the average ratings by roast type

roast_ratings = df.groupby('roast')['rating'].mean().
 ↪sort_values(ascending=False)

plt.figure(figsize=(10,6))
sns.barplot(x=roast_ratings.index, y=roast_ratings.values, palette=palette)
plt.xlabel('Roast Type', fontsize=15)
plt.ylabel('Average Rating', fontsize=15)
plt.title('Average Ratings by Roast Type', fontsize=20)
plt.show()
```



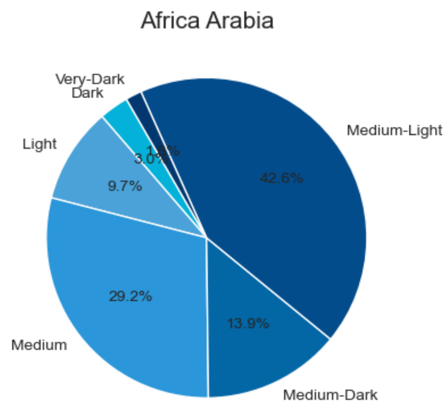
[88]:
```
roast_ratings
```

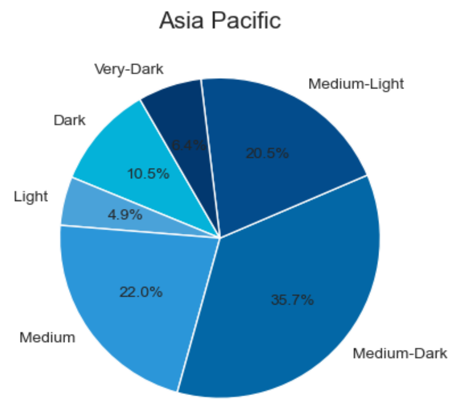
[88]:
```
roast
Light           8.741110
Medium-Light    8.441700
Medium          8.150154
Medium-Dark     7.510587
Dark            7.014515
Very-Dark       6.861716
Name: rating, dtype: float64
```

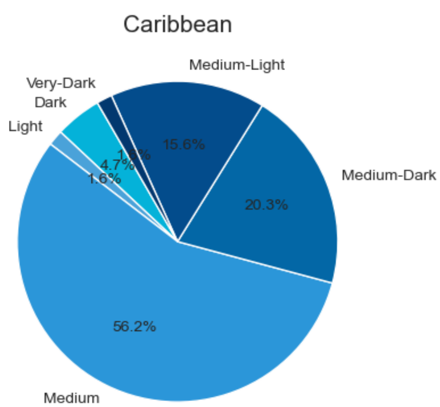
[89]:
```
# Display the average ratings by roast type with boxplot

plt.figure(figsize=(10,5))
sns.boxplot(data=df, x='roast', y='rating', palette=palette)
plt.xlabel('Roast Type')
plt.ylabel('Ratings')
plt.title('Rating Distribution by Roast Type', fontsize=20)
plt.show()
```

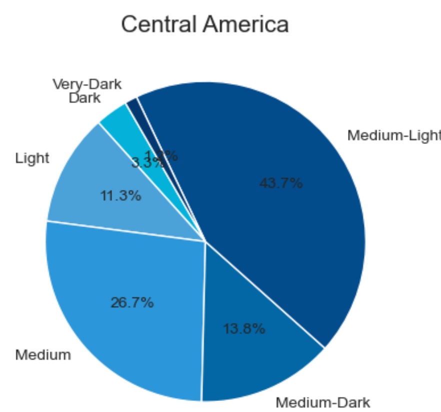


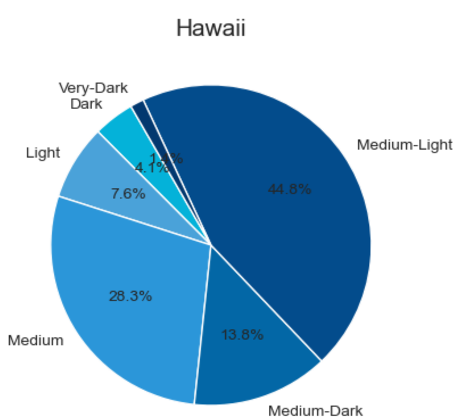
[90]:
```python
# Pie charts for each region showing the distribution of roasts

roast_region_counts = df.groupby(['region', 'roast']).size().
    ↪unstack(fill_value=0)

num_regions = len(roast_region_counts)
rows = (num_regions // 2) + 1
cols = min(num_regions, 2)

fig, axes = plt.subplots(rows, cols, figsize=(12, rows * 4))
axes = axes.flatten()
for i, (region, counts) in enumerate(roast_region_counts.iterrows()):
    roast_labels = counts.index.tolist()


    axes[i].pie(counts, labels=roast_labels, autopct='%1.1f%%', startangle=120)
    axes[i].set_title(region, fontsize=15)

plt.tight_layout()
fig.delaxes(axes[-1])
plt.show()
```

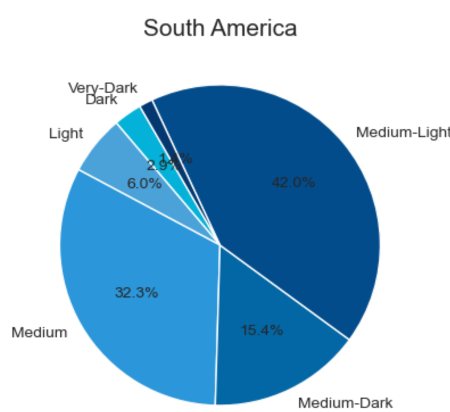

Africa Arabia
Very-Dark
Dark
Light
Medium
Medium-Dark
Medium-Light
42.6%
13.9%
29.2%
9.7%
3.0%

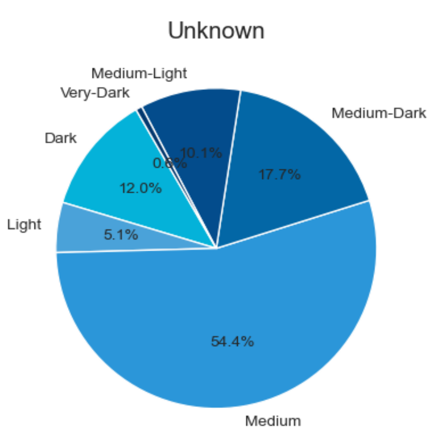

Asia Pacific
Very-Dark
Dark
Light
Medium
Medium-Dark
Medium-Light
6.4%
20.5%
35.7%
22.0%
4.9%
10.5%


Caribbean
Medium-Light
Very-Dark
Dark
Light
Medium-Dark
Medium
15.6%
20.3%
56.2%
4.7%
1.6%

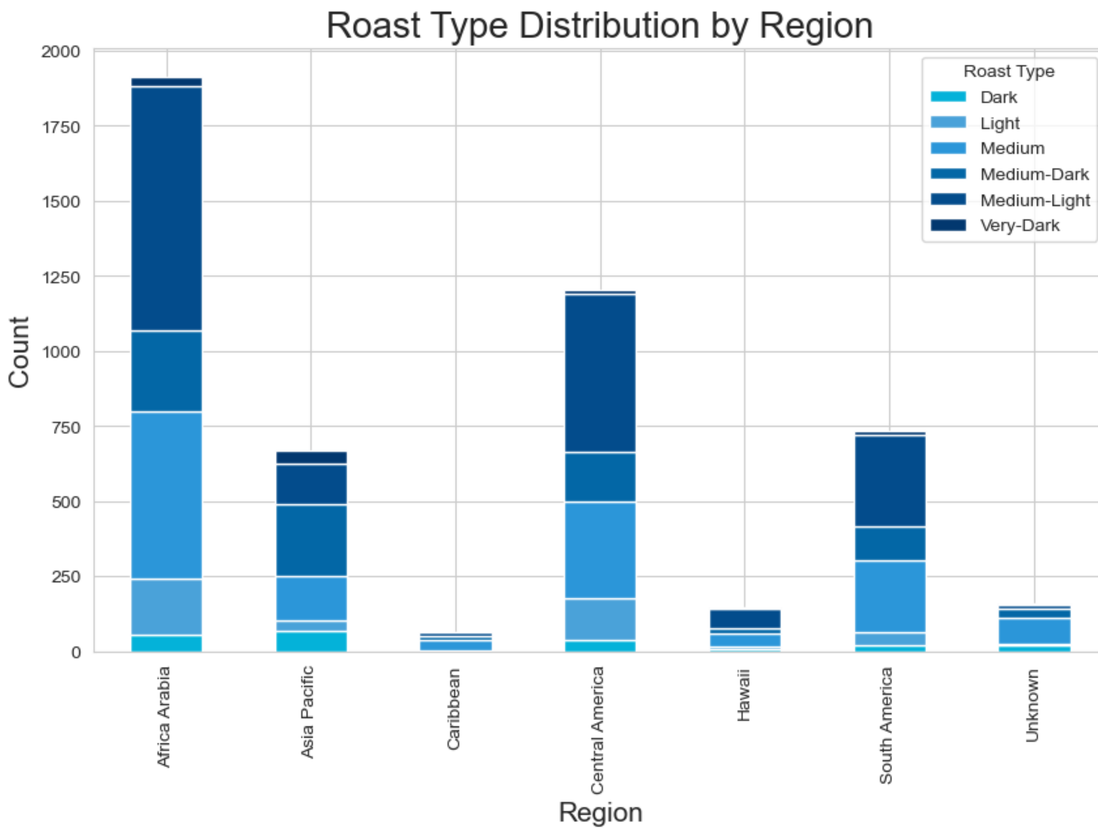

Central America
Very-Dark
Dark
Light
Medium
Medium-Dark
Medium-Light
43.7%
13.8%
26.7%
11.3%
3.3%


Hawaii
Very-Dark
Dark
Light
Medium
Medium-Dark
Medium-Light
44.8%
13.8%
28.3%
7.6%
4.1%

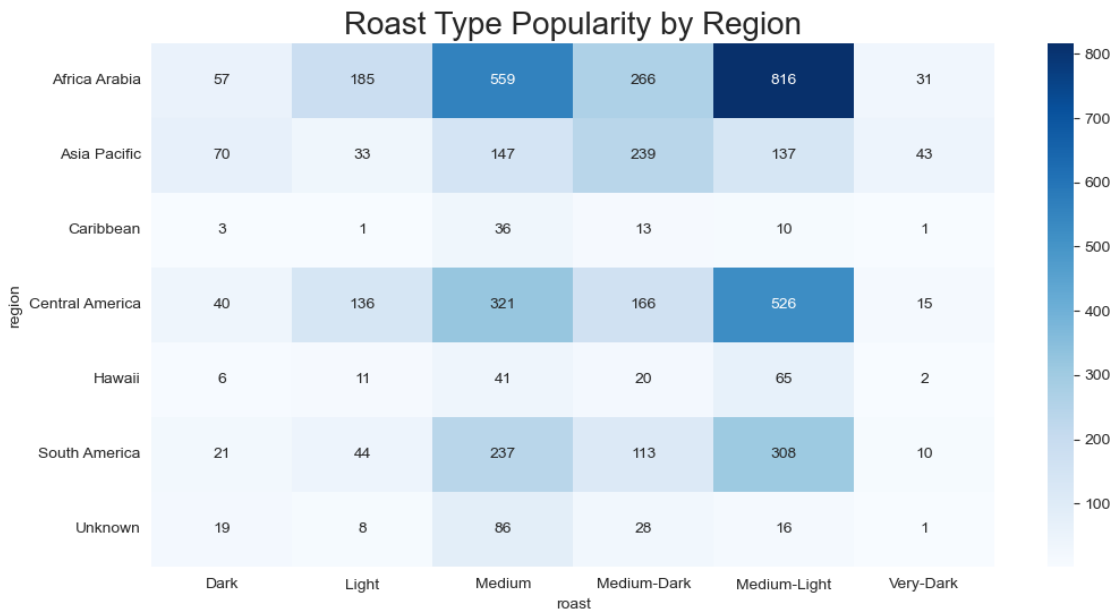

South America
Very-Dark
Dark
Light
Medium
Medium-Dark
Medium-Light
42.0%
15.4%
32.3%
6.0%
2.9%


Unknown
Medium-Light
Very-Dark
Dark
Light
Medium-Dark
Medium
10.1%
17.7%
54.4%
5.1%
12.0%
0.6%

[91]:
```
# Stacked bar chart showing the roast type distribution by region

roast_region_counts.plot(kind='bar', stacked=True, figsize=(10, 6))
plt.xlabel('Region', fontsize=15)
plt.ylabel('Count', fontsize=15)
plt.title('Roast Type Distribution by Region', fontsize=20)
plt.legend(title='Roast Type')
plt.show()
```

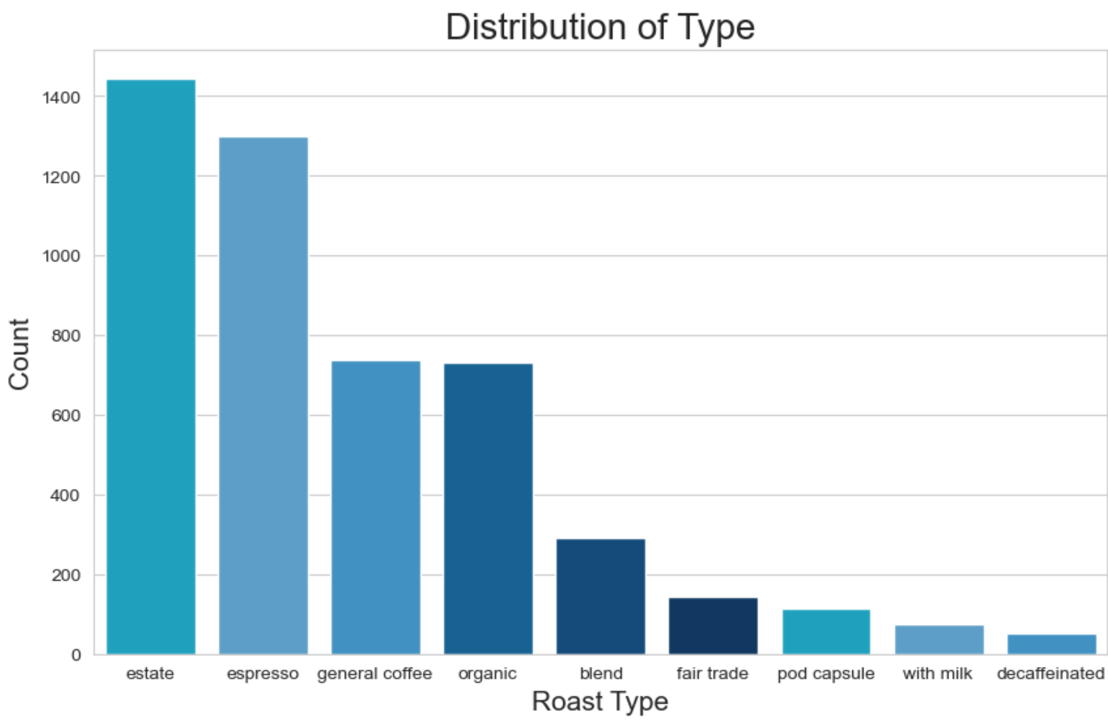


[92]:
```
# Heatmap showing the popularity of each roast type across different regions

roast_region = df.pivot_table(index='region', columns='roast', values='rating',␣
↪aggfunc='count')

plt.figure(figsize=(12,6))
sns.heatmap(roast_region, cmap='Blues', annot=True, fmt='.0f')
plt.title('Roast Type Popularity by Region', fontsize=20)
plt.show()
```



3.8 Coffee Types

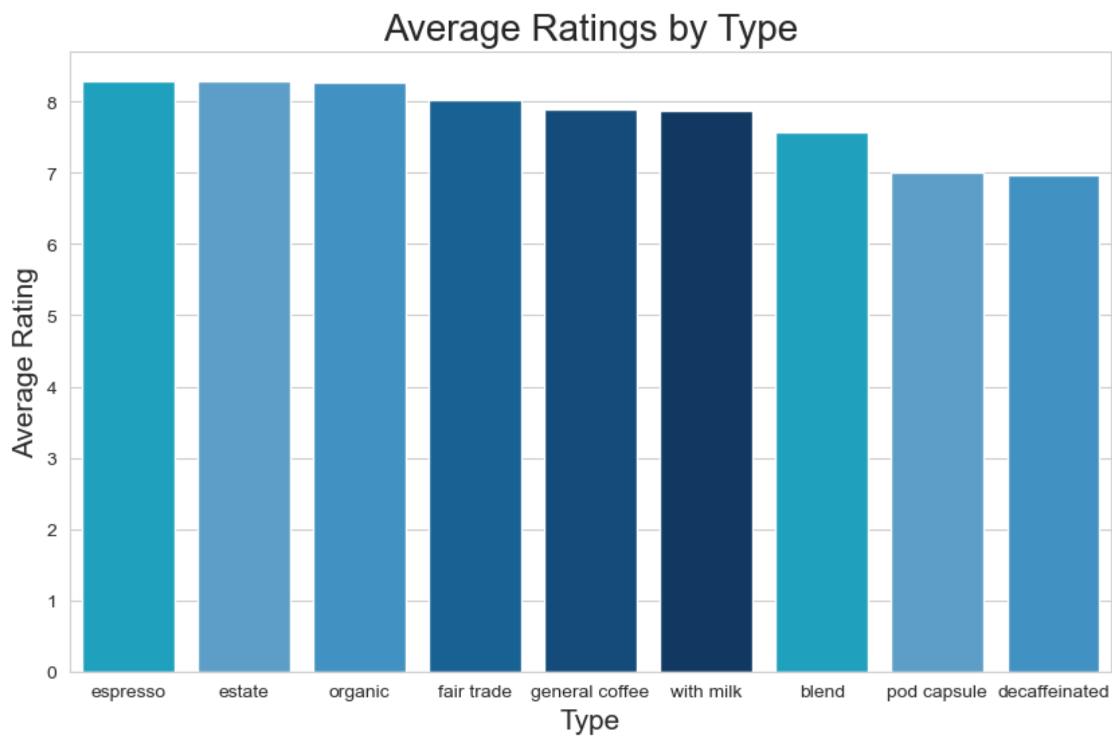
[93]:
```
# Distribution of type

plt.figure(figsize=(10,6))
sns.countplot(data=df, x='type', order=df['type'].value_counts().index,␣
  ↪palette=palette)
plt.xlabel('Roast Type', fontsize=15)
plt.ylabel('Count', fontsize=15)
plt.title('Distribution of Type', fontsize=20)
plt.show()
```



[94]:
```
df['type'].value_counts()
```

[94]:
```
type
estate            1445
espresso          1298
general coffee     737
organic            732
blend              290
fair trade         142
pod capsule        115
with milk           76
decaffeinated       52
Name: count, dtype: int64
```

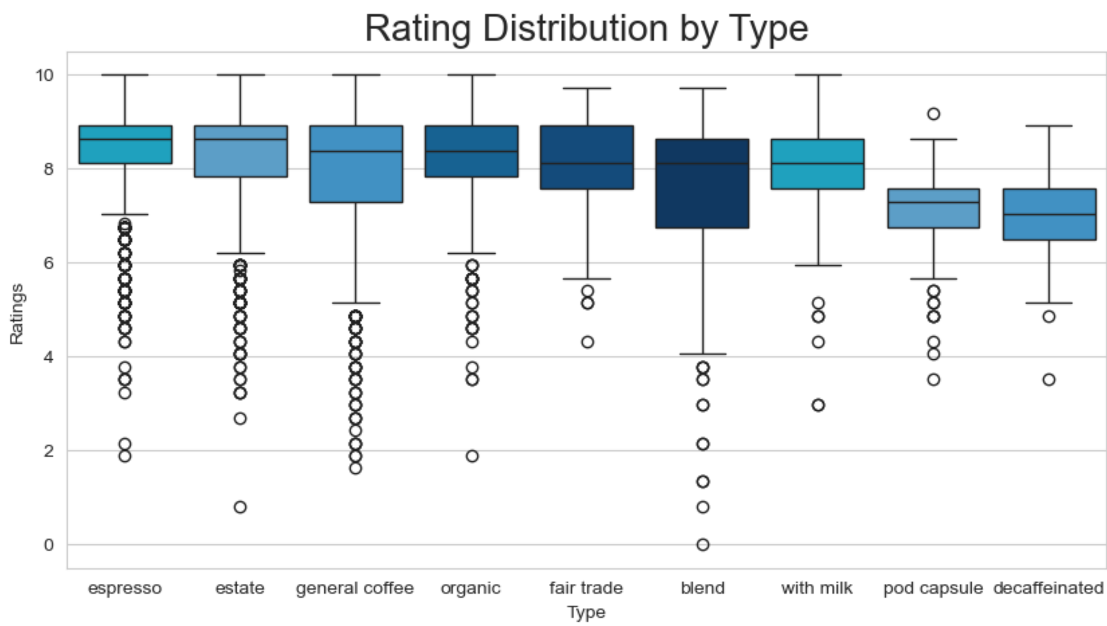
[95]:
```
# Display the average ratings by type

type_ratings = df.groupby('type')['rating'].mean().sort_values(ascending=False)

plt.figure(figsize=(10,6))
sns.barplot(x=type_ratings.index, y=type_ratings.values, palette=palette)
plt.xlabel('Type', fontsize=15)
plt.ylabel('Average Rating', fontsize=15)
plt.title('Average Ratings by Type', fontsize=20)
plt.show()
```



[96]:
```
type_ratings
```

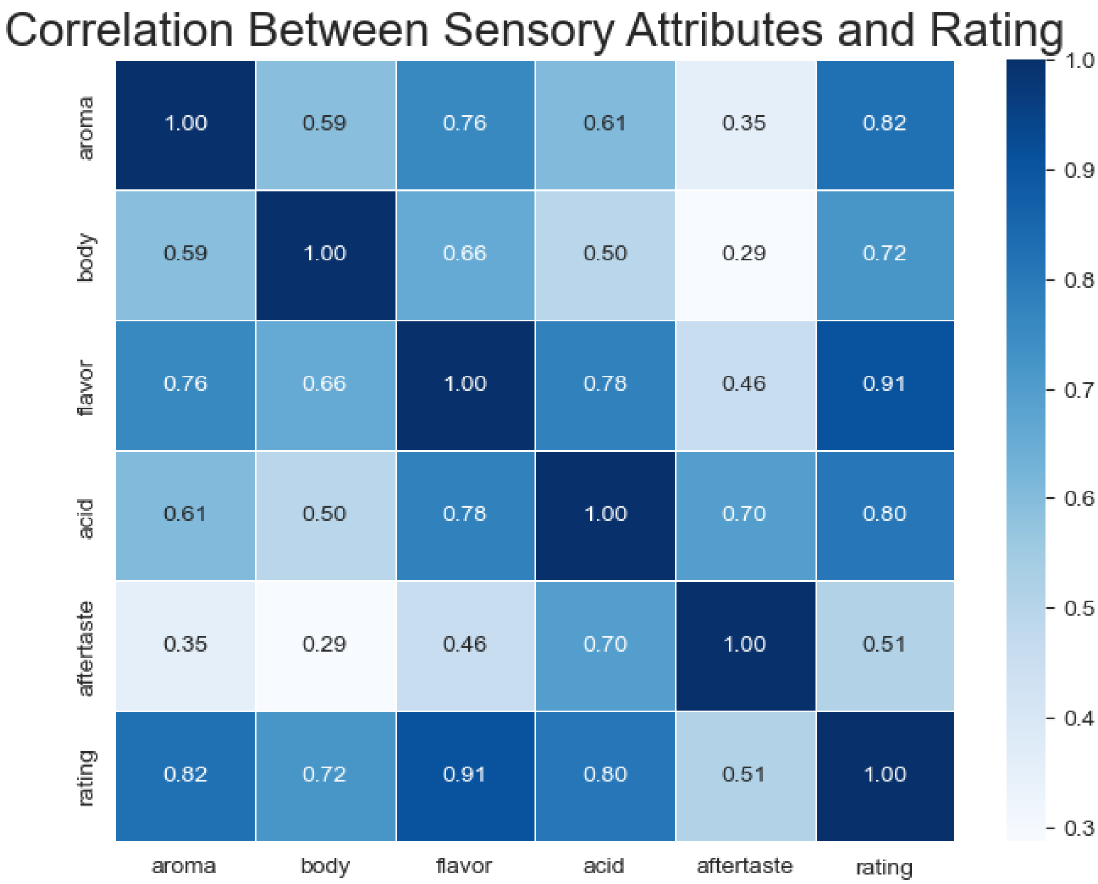
[96]:
```
type
espresso          8.294101
estate            8.283662
organic           8.262074
fair trade        8.020556
general coffee    7.893579
with milk         7.880512
blend             7.580615
pod capsule       7.010576
decaffeinated     6.969854
Name: rating, dtype: float64
```

[97]:
```
# Display the average ratings by type with boxplot

plt.figure(figsize=(10,5))
sns.boxplot(data=df, x='type', y='rating', palette=palette)
plt.xlabel('Type')
plt.ylabel('Ratings')
plt.title('Rating Distribution by Type', fontsize=20)
plt.show()
```



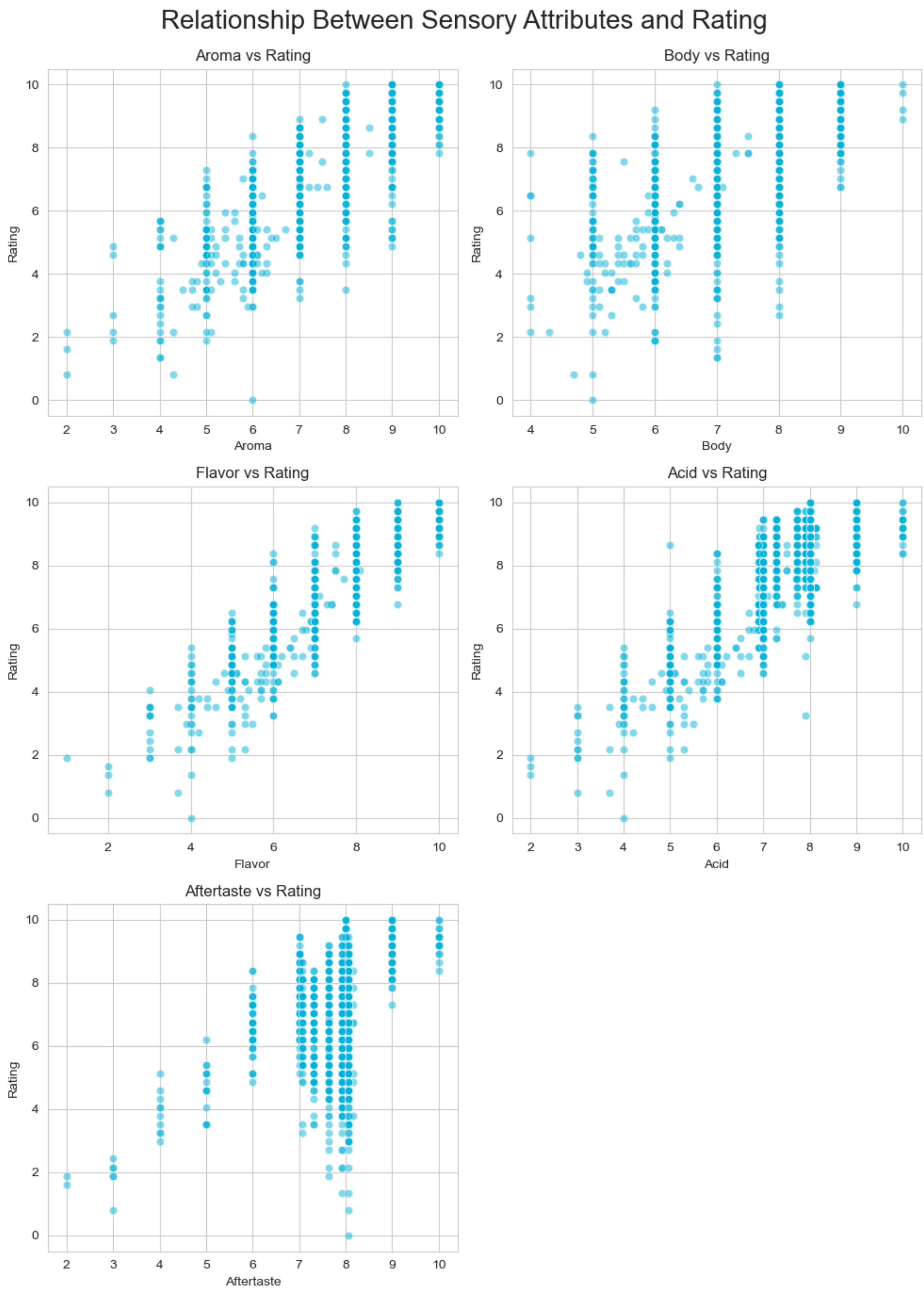
3.9 Sensory Attributes Analysis

[98]:
```
# Display how sensory attributes correlate with each other and with overall␣
↪rating

sensory_cols = ['aroma', 'body', 'flavor', 'acid', 'aftertaste', 'rating']
sensory_data = df[sensory_cols]

correlation_matrix = sensory_data.corr()

plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='Blues', fmt='.2f',␣
↪linewidths=0.5)
plt.title('Correlation Between Sensory Attributes and Rating', fontsize=20)
plt.show()
```



[99]:
```python
sensory_cols = ['aroma', 'body', 'flavor', 'acid', 'aftertaste']
```

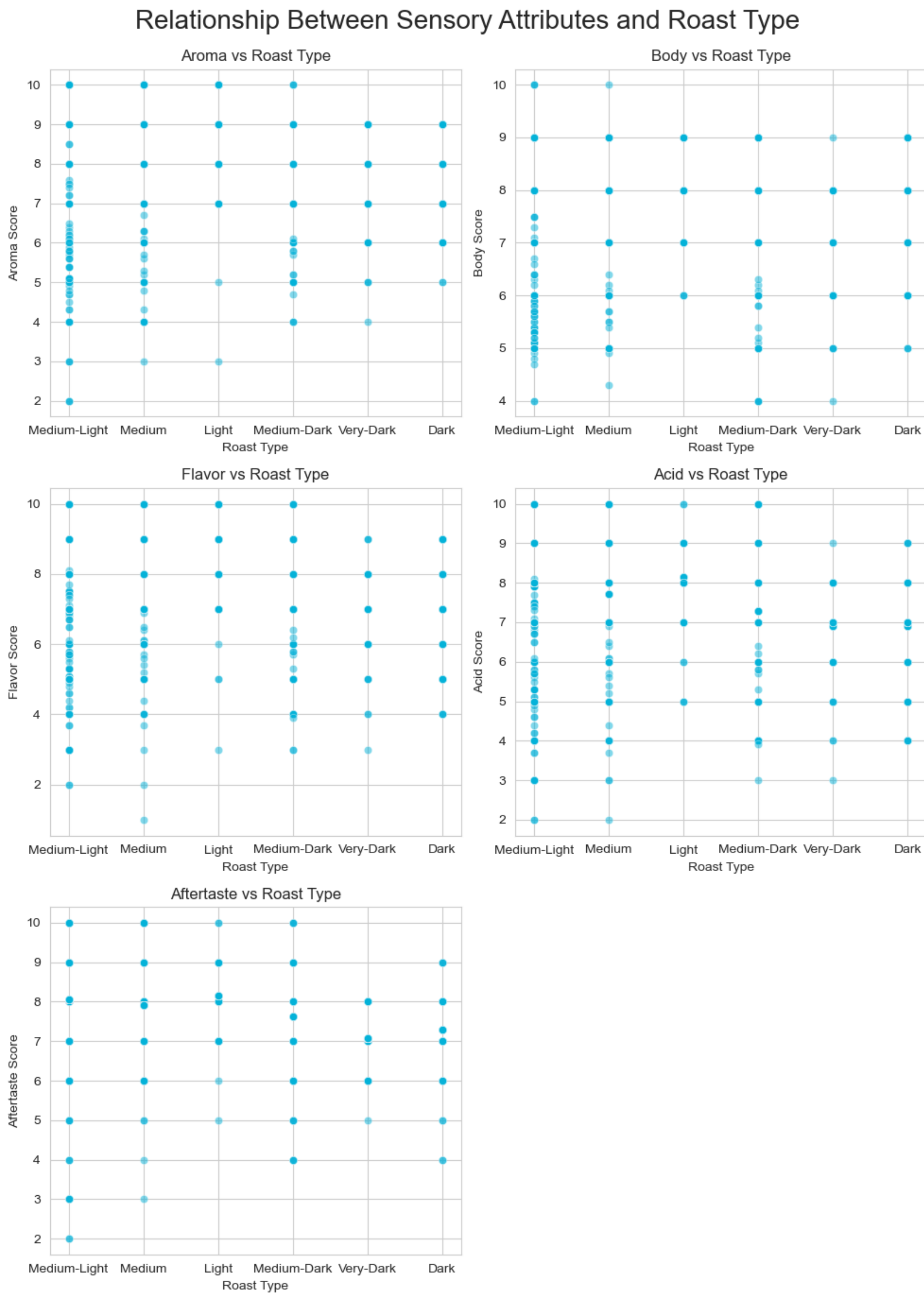
[100]:
```python
# Visualize the relationship between sensory attributes and rating

fig, axes = plt.subplots(3, 2, figsize=(10, 14))
axes = axes.flatten()

for i, col in enumerate(sensory_cols):
    sns.scatterplot(data=df, x=col, y='rating', alpha=0.5, ax=axes[i])
    axes[i].set_title(f'{col.capitalize()} vs Rating')
    axes[i].set_xlabel(col.capitalize())
    axes[i].set_ylabel('Rating')

fig.delaxes(axes[-1])

plt.suptitle('Relationship Between Sensory Attributes and Rating', fontsize=20)
plt.tight_layout(rect=[0, 0, 1, 0.99])
plt.show()
```



Relationship Between Sensory Attributes and Rating
Aroma vs Rating
Body vs Rating
Flavor vs Rating
Acid vs Rating
Aftertaste vs Rating
Rating
Aroma
Body
Flavor
Acid
Aftertaste

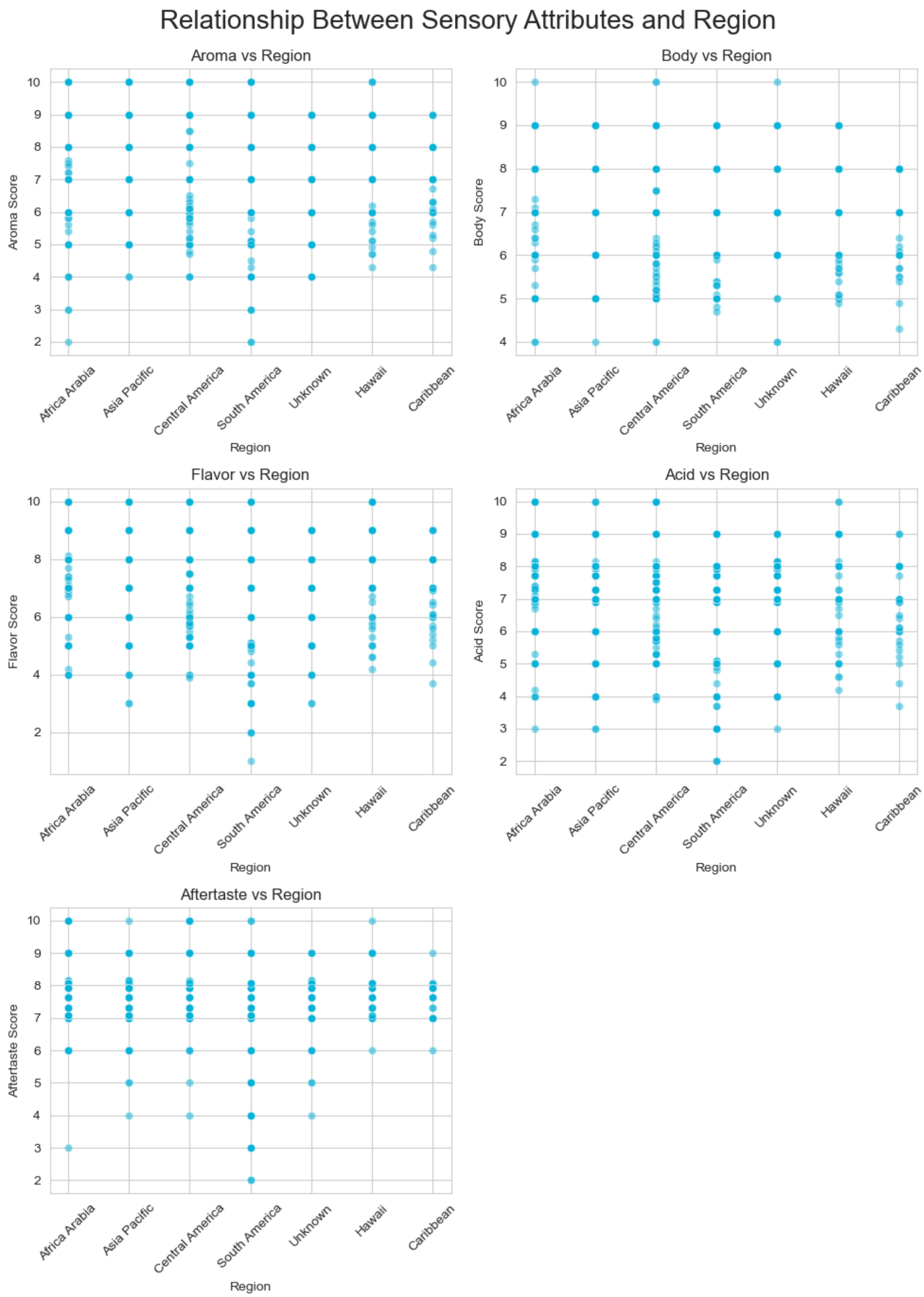
[101]:

```
# Visualize the relationship between sensory attributes and roast type

fig, axes = plt.subplots(3, 2, figsize=(10, 14))
axes = axes.flatten()

for i, col in enumerate(sensory_cols):
    sns.scatterplot(data=df, y=col, x='roast', alpha=0.5, ax=axes[i])
    axes[i].set_title(f'{col.capitalize()} vs Roast Type')
    axes[i].set_xlabel('Roast Type')
    axes[i].set_ylabel(f'{col.capitalize()} Score')

fig.delaxes(axes[-1])

plt.suptitle('Relationship Between Sensory Attributes and Roast Type',␣
 ↪fontsize=20)
plt.tight_layout(rect=[0, 0, 1, 0.99])
plt.show()
```

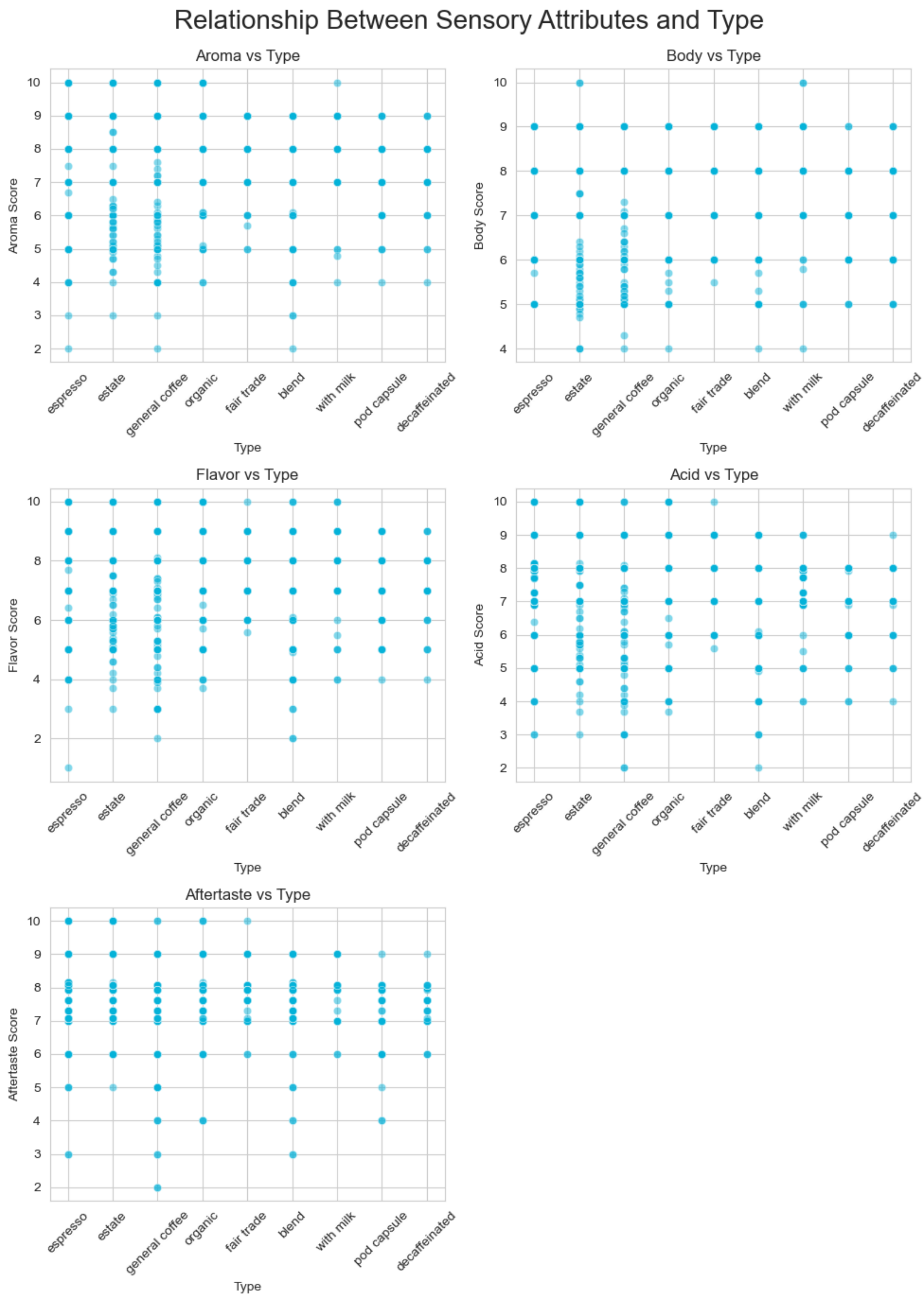

Relationship Between Sensory Attributes and Roast Type
Aroma vs Roast Type
Aroma Score
Body vs Roast Type
Body Score
Flavor vs Roast Type
Flavor Score
Acid vs Roast Type
Acid Score
Aftertaste vs Roast Type
Aftertaste Score
Roast Type
Medium-Light
Medium
Light
Medium-Dark
Very-Dark
Dark

[102]:

```python
# Visualize the relationship between sensory attributes and region

fig, axes = plt.subplots(3, 2, figsize=(10, 14))
axes = axes.flatten()

for i, col in enumerate(sensory_cols):
    sns.scatterplot(data=df, y=col, x='region', alpha=0.5, ax=axes[i])
    axes[i].set_title(f'{col.capitalize()} vs Region')
    axes[i].set_xlabel('Region')
    axes[i].set_ylabel(f'{col.capitalize()} Score')
    axes[i].tick_params(axis='x', rotation=45)

fig.delaxes(axes[-1])

plt.suptitle('Relationship Between Sensory Attributes and Region', fontsize=20)
plt.tight_layout(rect=[0, 0, 1, 0.99])
plt.show()
```


Relationship Between Sensory Attributes and Region
Aroma vs Region
Aroma Score
Body vs Region
Body Score
Flavor vs Region
Flavor Score
Acid vs Region
Acid Score
Aftertaste vs Region
Aftertaste Score
Region
Africa Arabia
Asia Pacific
Central America
South America
Unknown
Hawaii
Caribbean

[103]:
```
# Visualize the relationship between sensory attributes and type

fig, axes = plt.subplots(3, 2, figsize=(10, 14))
axes = axes.flatten()

for i, col in enumerate(sensory_cols):
    sns.scatterplot(data=df, y=col, x='type', alpha=0.5, ax=axes[i])
    axes[i].set_title(f'{col.capitalize()} vs Type')
    axes[i].set_xlabel('Type')
    axes[i].set_ylabel(f'{col.capitalize()} Score')
    axes[i].tick_params(axis='x', rotation=45)

fig.delaxes(axes[-1])

plt.suptitle('Relationship Between Sensory Attributes and Type', fontsize=20)
plt.tight_layout(rect=[0, 0, 1, 0.99])
plt.show()
```


Relationship Between Sensory Attributes and Type
Aroma vs Type
Aroma Score
Body vs Type
Body Score
Flavor vs Type
Flavor Score
Acid vs Type
Acid Score
Aftertaste vs Type
Aftertaste Score
Type
espresso
estate
general coffee
organic
fair trade
blend
with milk
pod capsule
decaffeinated

3.10 Summary of EDA

3.10.1 Ratings Distribution

Mean is 8.12. The distribution of ratings is right-peaked, with most values between 7.5 and 9.5, showing a clear tendency toward high scores. However, a long left tail and several low outliers (some near 0) create a left-skewed distribution. The box plot confirms this pattern, with ratings mostly between 7.6 and 9.0 and a median of 8.3, but the presence of low outliers pulls the overall distribution downward.

3.10.2 Top-Rated Products

Top rated products with 5+ reviews stand out with average ratings above 8.17. Notable entries include:

- Sumatra Tano Batak 9.15 (7 reviews)
- Tano Batak Sumatra 9.13 (5 reviews)
- Guatemala Acatenango Gesha 9.08 (5 reviews)

The top-rated coffees primarily come from Asia Pacific and Africa Arabia. Notable entries: - Sumatra Tano Batak Asia Pacific 9.15 (7 reviews) - Tano Batak Sumatra Asia Pacific 9.13 (5 reviews) - Sumatra Ulos Batak Asia Pacific 9.08 (5 reviews) - Ethiopia Hambela Natural Africa Arabia 8.95 (7 reviews)

Coffees roasted as Medium-Light dominate the top rankings. Notable entries: - Tano Batak Sumatra Medium-Light 9.13 (5 reviews) - Flight Seasonal Espresso Medium-Light 9.08 (5 reviews) - Holiday Blend Medium-Light 9.0 (10 reviews)

Among different coffee types, the best rated were: - Flight Seasonal Espresso Espresso 9.08 - Holiday Blend Blend 8.93 - Ethiopia Yirgacheffe General coffee 8.71

Top-rated coffees tend to come from the Asia Pacific and Africa Arabia regions, with Sumatra varieties performing especially well. Medium-light roasts consistently receive the highest ratings, and among coffee types, espressos lead in overall quality

3.10.3 Roaster Analysis

The most frequently reviewed roaster include: - JBC Coffee Roasters – 178 reviews - Green Mountain Coffee – 157 reviews - Paradise Roasters – 153 reviews

The highest average-rated roasters - Dragonfly Coffee Roasters – 9.14 (51 reviews) - Kakalove Cafe – 9.08 (61 reviews) - Simon Hsieh's Aroma Roast Coffees – 9.06 (49 reviews)

Roaster with offerings from Africa Arabia and Central America received the highest average ratings: - Simon Hsieh's Aroma Roast Coffees – Africa Arabia 9.23 (27 reviews) - Dragonfly Coffee Roasters – Africa Arabia 9.22 (20 reviews) / Central America 9.15 (25 reviews) - Kakalove Cafe - Africa Arabia 9.11 (46 reviews)

Roasters roasted as Medium-Light and Light dominate the top rankings - dragonfly coffee roasters Medium-Light 9.13 (37 reviews) - Kakalove Cafe Medium-Light 9.10 (40 reviews) / Light 9.06 (15 reviews) - propeller coffee Medium-Light 9.02 (10 reviews)

But we also have Simon Hsieh's Aroma Roast Coffees Medium-Dark 9.13 (14 reviews)

Among different coffee types the best rated were: - Dragonfly Coffee Roasters Estate 9.23 23 reviews / organic 9.09 (25 reviews) - JBC Coffee Roasters general coffee 9.18 (22 reviews) - Kakalove Cafe general coffee 9.14 (25 reviews) / espresso 9.04 (11 reviews) / organic 9.03 (18 reviews) - Simon Hsieh's Aroma Roast Coffees espresso 9.06 49 reviews

The highest-rated roasters are not always the most frequently reviewed, with standout quality coming from smaller, premium producers like Dragonfly Coffee Roasters, Kakalove Cafe, and Simon Hsieh's Aroma Roast Coffees. Roasters sourcing from Africa Arabia and Central America consistently receive top ratings. Medium-light and light roasts dominate the high scores, though select medium-dark offerings also perform strongly. Across coffee types, estate, organic, and general coffee categories lead in average ratings, especially from Dragonfly, JBC, and Kakalove.

3.10.4 Seasonal Trends

- Fall received the highest average rating 8.29 with 1,321 reviews.
- Summer received average rating 8.13 with 1139 reviews.
- Winter received average rating 8.05 with 1234 reviews.
- Spring had the lowest average rating 8.01 with 1193 reviews.
- The results suggest that consumers tend to rate products more favourably in the second half of the year.

3.10.5 Long-Term Trends

The long-term trend in average ratings from 1997 to 2019 shows a clear upward trajectory in coffee quality or consumer satisfaction over time. In the late 1990s average ratings was significantly lower, starting at 5.42 in 1997 and gradually increasing. From 2003 onward, there was a consistent rise in rating surpassing the 8.0 threshold by 2007. The peak average rating was observed in 2019 reaching 9.07 although this value is based on a limited number of views (11), so it may be less reliable. The years 2013-2018 show a period of stable high performance with average ratings consistently above 8.5 and strong review counts indicating high-quality products and a more favourable reviewing climate.

Overall, these trends reflect a steady improvement in product quality or perception, with recent years achieving consistently high ratings, particularly in fall seasons.

3.10.6 Regional Performance

Africa Arabia clearly dominating in both volume and quality. - Africa Arabia is the most represented region with 1914 reviews and the highest average rating of 8.54 - Central America has 1204 reviews and average rating of 8.22 - South Africa and Asia Pacific are mid-tier in both volume and rating with reviews 773 and 669 and average ratings 7.76 and 7.66 - Hawaii has relatively few reviews 145 and average of rating 7.92 - Caribbean and Unknown regions has the lowest rating 6.85 and 6.73 with reviews 64 and 158, suggesting either lower quality or less favourable perception.

Coffees from Africa Arabia and Central America are commonly reviewed and consistently better rated by costumers.

3.10.7 Roast Types

- Light roast performs the best with 8.74 average rating despite having only 418 reviews
- Medium-light is the most popular with 1878 reviews with average rating 8.74
- Medium roast are solid but slightly lower with average rating 8,15 and 1427 reviews
- Medium-Dark has 845 reviews and average rating 7.51
- Dark has 216 reviews and average rating 7.01
- Vert-Dark has the worst average rating 6.68 and has the least reviews 103

These results imply that lighter roasts are generally preferred by reviewers and may be associated with higher quality beans or more refined roasting practices.

3.10.8 Coffee Types

The most common coffee types in the dataset are estate, espresso, and general coffee - Estate is the most reviews with 1445 reviews, with average rating 8.28 - Espresso has 1298 reviews and average rating 8.28 - General coffee has 939 reviews but slightly lower average rating 7.89 - Organic has 732 reviews and hight average rating 8.26 - Blend has 290 reviews and average rating 7.58 - Fair Trade has 142 reviews and average rating 8.02 - Pod/Capsule has 115 reviews and average rating 7.01 - With milk has 76 reviews and average rating 7.88 - Decaffeinated has the least reviews 52 and the worst average rating 6.96

These findings indicate a clear preference for more traditional and specialty-oriented coffee types, particularly espresso and estate, both in terms of frequency and reviewer satisfaction.

3.10.9 Sensory Attributes Analysis

Correlations with Overall Rating The strongest correlation with overall rating is flavor 0.91 (the strongest predictor) aroma 0.82, acidity 0.80, body 0.72, the weakest is aftertaste 0.51 – has the least influence on the overall rating compared to other attributes. Flavor and acidity has strong correlation of 0.78, suggesting these qualities often go hand in hand. Flavor and aroma another strong correlation 0.76, acidity and aftertaste surprisingly high correlation of 0.70 possibly indicating that acidity contributes to a longer-lasting sensory impression.

Key attribute relationships **Aroma vs rating**

A clean positive correlation, as aroma scores increase overall ratings tend to be higher.

Body vs rating

Corelation is weaker than for aroma but still noticeable, when body scores exceed 8, ratings are almost alvays higher (>7), however for body scores around 6-7 there is a wide range of final ratings.

Flavor vs rating

The strongest positive correlation, as flavor increases, rating almost linearly increases as well. Flavor appears to be a key determinant of a high overall score.

Acid vs rating

A clear but more moderate positive correlation is visible, acid values above 7 are mostly associated with higher ratings, below 7 the result show greater variability.

Aftertaste vs rating

Similar to body, the relationship is weaker and more diffuse, there is a concentration of points in the 7-9 range for both aftertaste and rating, however the spread is quite broad for similar aftertaste values.

Flavor and Aroma have the strongest influence on the final rating. Acidity also shows a meaningful relationship, though with greater variability. Body and Aftertaste are relevant, but their connection to the overall rating is weaker and less direct.

Sensory Attributes by Category Roast type

There is no strong or consistent relationship between most sensory attributes and roast level. The majority of data comes from Medium-Light and Medium roasts, and the underrepresentation of other roast types may limit the reliability of broader conclusions.

Region

Coffees from Africa/Arabia and Asia Pacific consistently exhibit the highest sensory quality across all evaluated attributes. In contrast, Caribbean and Unknown regions generally show lower performance across most categories. While Hawaiian coffees achieve strong scores, the limited number of samples restricts broader conclusions. Central and South America offer a well-balanced and reliable sensory profile - not exceptional, but consistently solid.

Coffee type

Estate and Espresso are the most represented types, which may explain their consistent and high sensory scores. General coffee and Organic also have substantial representation but show more variability and lower ratings. With milk and Decaffeinated are underrepresented, so their trends should be interpreted with caution. Despite a smaller sample size, Fair trade performs well and may offer valuable insights for further research or marketing.

4 Feature Engineering

[104]:
```
# Encodes categorical columns based on their correlation with the target␣
↪variable using target encoding.

encoder = TargetEncoder(cols=['roast', 'region', 'type'])
df[['roast', 'region', 'type']] = encoder.fit_transform(df[['roast', 'region',␣
↪'type']], df['rating'])
```

[105]:
```
# Creates interaction features by multiplying numerical and encoded categorical␣
↪columns to capture combined effects.

df['flavor_x_region'] = df['flavor'] * df['region']
df['flavor_x_roast'] = df['flavor'] * df['roast']
df['acid_x_roast'] = df['acid'] * df['roast']
df['aroma_x_region'] = df['aroma'] * df['region']
df['aroma_x_roast'] = df['aroma'] * df['roast']
df['flavor_x_type'] = df['flavor'] * df['type']
```

[106]:
```python
# Applies log transformation to the 'rating' column to reduce skewness and␣
↪stabilize variance.

df['rating_log'] = np.log1p(df['rating'])
```

[107]:
```python
# Defines a preprocessing pipeline to standardize selected numerical and␣
↪interaction features using StandardScaler

num_cols = ['roast', 'region','type', 'flavor_x_region', 'flavor_x_roast',␣
↪'aroma_x_roast', 'aroma_x_region', 'acid_x_roast', 'flavor_x_type']

preprocessor = ColumnTransformer(transformers=[
    ('num', StandardScaler(), num_cols)])
```

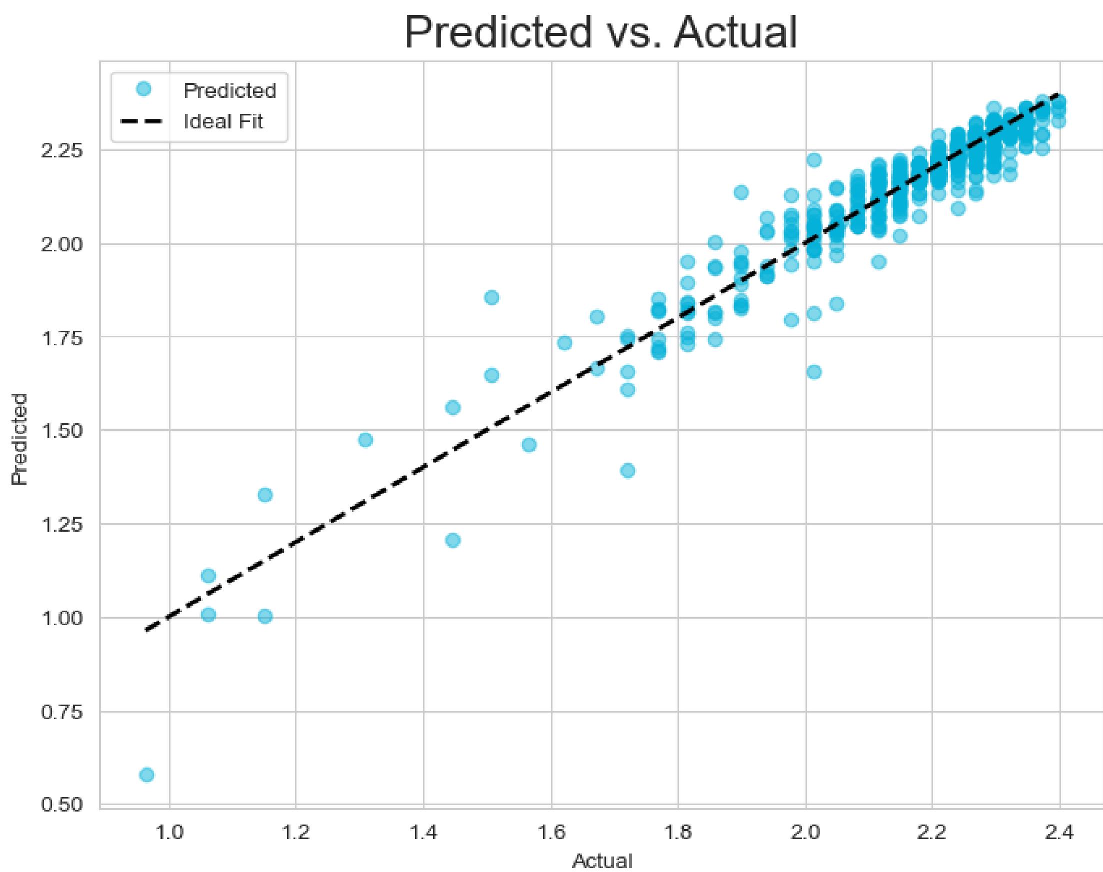
5 Modeling

[108]:
```python
# Splits the dataset into training and testing sets for model evaluation, using␣
↪log-transformed ratings as the target.

X = df[['roast', 'region', 'type','flavor_x_region', 'flavor_x_roast',␣
↪'aroma_x_roast', 'aroma_x_region', 'acid_x_roast', 'flavor_x_type']]
y = df['rating_log']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[109]:
```python
# Builds and fits a pipeline with preprocessing and a Gradient Boosting␣
↪Regressor, then evaluates performance using MSE and R-Squared on the test␣
↪set.

pipeline = Pipeline(steps=[
    ('preprocessing', preprocessor),
    ('model', GradientBoostingRegressor())
])

pipeline.fit(X_train, y_train)
y_pred = pipeline.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-Squared: {r2}")
```

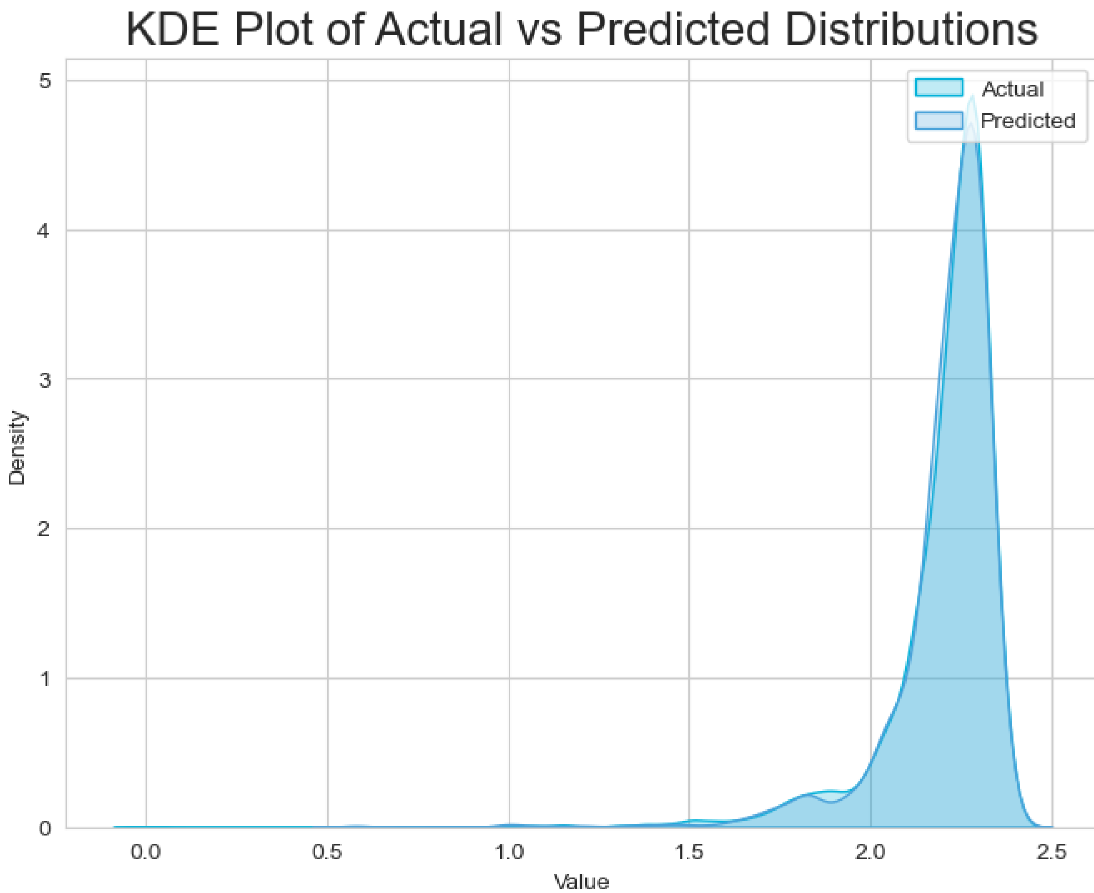
```
Mean Squared Error: 0.0022461694997454314
R-Squared: 0.9113018594526301
```

[110]:
```python
# Visualizes predicted vs. actual log-transformed ratings to assess model␣
↪performance and alignment with the ideal fit.

plt.figure(figsize=(8, 6))
plt.plot(y_test, y_pred, 'o', alpha=0.5, label='Predicted')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--',␣
↪lw=2, label='Ideal Fit')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Predicted vs. Actual', fontsize=20)
plt.legend()
plt.grid(True)
plt.show()
```

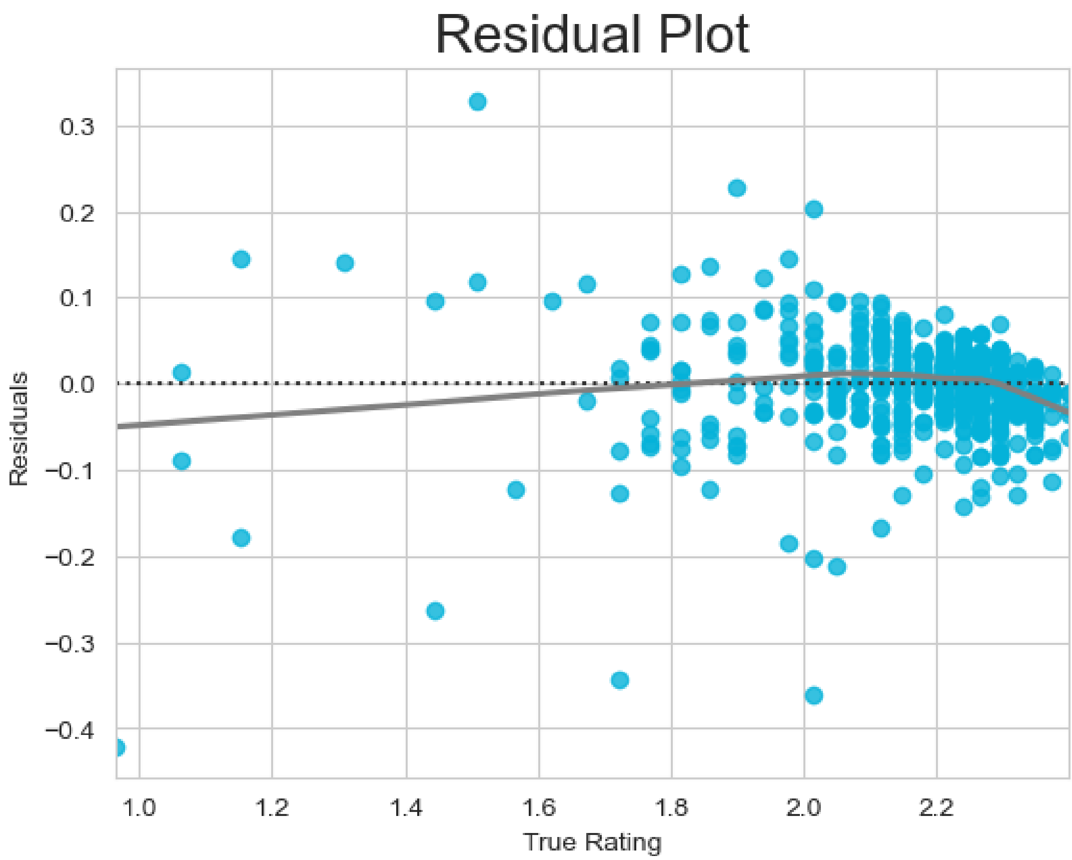


[111]:
```python
# Visualizes the KDE plot to compare the distributions of actual and predicted␣
↪ratings.

plt.figure(figsize=(8, 6))
sns.kdeplot(y, label='Actual', shade=True)
```

```
sns.kdeplot(y_pred, label='Predicted', shade=True)
plt.title('KDE Plot of Actual vs Predicted Distributions', fontsize=20)
plt.xlabel('Value')
plt.ylabel('Density')
plt.legend()
plt.grid(True)
plt.show()
```



[112]:
```
# Visualizes residuals to assess the difference between true and predicted␣
↪ratings.

sns.residplot(x=y_test, y=y_pred, lowess=True, line_kws={"color": "grey"})
plt.xlabel("True Rating")
plt.ylabel("Residuals")
plt.title("Residual Plot", fontsize=20)
plt.show()
```



[113]:
```
# Extracts the list of feature names from the preprocessing step in the
↪pipeline.

preprocessor = pipeline.named_steps['preprocessing']

feature_names = preprocessor.transformers_[0][2]
feature_names
```

[113]:
```
['roast',
 'region',
 'type',
 'flavor_x_region',
 'flavor_x_roast',
 'aroma_x_roast',
 'aroma_x_region',
 'acid_x_roast',
 'flavor_x_type']
```

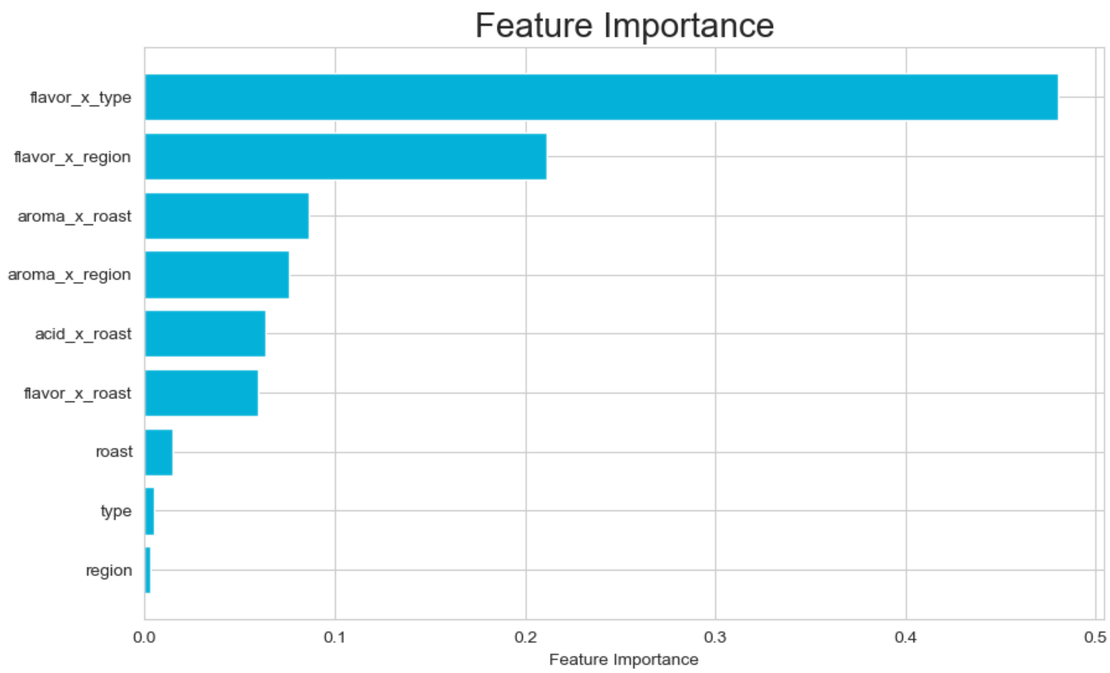
[114]:
```python
# Extracts and ranks feature importances from the trained model.

model = pipeline.named_steps['model']

importances = model.feature_importances_

importance_df = pd.DataFrame({
    'feature': feature_names,
    'importance': importances
}).sort_values(by='importance', ascending=False)
```

[116]:
```python
importance_df
```

[116]:

	feature	importance
8	flavor_x_type	0.480436
3	flavor_x_region	0.211537
5	aroma_x_roast	0.086137
6	aroma_x_region	0.076083
7	acid_x_roast	0.063553
4	flavor_x_roast	0.059611
0	roast	0.014875
2	type	0.004974
1	region	0.002795

[117]:
```python
# Visualize feature importances, with the most important features at the top.

plt.figure(figsize=(10, 6))
plt.barh(importance_df['feature'], importance_df['importance'])
plt.xlabel("Feature Importance")
plt.title("Feature Importance", fontsize=20)
plt.gca().invert_yaxis()
plt.show()
```



5.1 Summary

5.1.1 Feature Engineering

Target Encoding

Categorical variables were encoded using target encoding, leveraging their correlation with the target variable.

Interaction Features

New features were engineered by creating pairwise interactions between key numeric and categorical columns to capture relationships:

- `flavor_x_region` - Interaction between flavor and region
- `flavor_x_roast` - Interaction between flavor and roast level
- `acid_x_roast` - Interaction between acidity and roast level
- `aroma_x_region` - Interaction between aroma and region
- `aroma_x_roast` - Interaction between aroma and roast level
- `flavor_x_type` - Interaction between flavor and coffee type

Target Variable Transformation

The rating column was log-transformed to reduce skewness and stabilize variance.

Feature Scaling

Selected numerical and interaction features were standardized using StandardScaler as part of the preprocessing pipeline.

5.1.2 Modeling

Model Choice

A GradientBoostingRegressor was selected for its ability to capture non-linear relationships and handle interaction features effectively.

Data Splitting

The dataset was split into training and testing sets, using the log-transformed rating as the target variable.

Model Performance

- Mean Squared Error (MSE): 0.0022
- R-Squared : 0.9113

The high R-Squared indicates that the model explains most of the variance in the target variable. The low MSE confirms strong predictive accuracy.

Predicted vs Actual Plot

Predictions align well with actual values, especially within the 2.0–2.4 log-scale range (corresponding to original ratings above ~7.5). At lower rating values (1.0–1.6 log-scale, or original ratings below 5), prediction spread increases. This performance drop at low ratings likely results from data imbalance, as low-rating samples are underrepresented in the dataset.

Residual Plot

Centered around zero: Indicates no systematic bias.

Funnel-shaped pattern: Suggests heteroscedasticity, with higher error variance at lower rating values, again tied to sparse data in this range.

Slight negative skew at high predicted values: May reflect a minor underestimation of the highest-rated coffees.

Feature Importance

- `flavor_x_type` - 0.480 - Most predictive; flavor varies significantly by type
- `flavor_x_region` - 0.212 - Regional flavor differences are key
- `aroma_x_roast` - 0.086 - Roast significantly influences aroma
- `aroma_x_region` - 0.076 - Aroma characteristics vary across regions
- `acid_x_roast` - 0.064 - Perceived acidity is affected by roast level
- `flavor_x_roast` - 0.060 - Flavor responds to roasting variations
- `roast` - 0.015 - Weak individual predictor
- `type` - 0.005 - Minimal standalone contribution
- `region` - 0.003 - Least important in isolation

Interaction features dominate, particularly those involving flavor, underscoring that flavor's impact is highly context-dependent.Raw features on their own offer limited predictive power unless part of an interaction.