EDS Theory Activity No. 1

Name:- Sumit Mohan Kadam

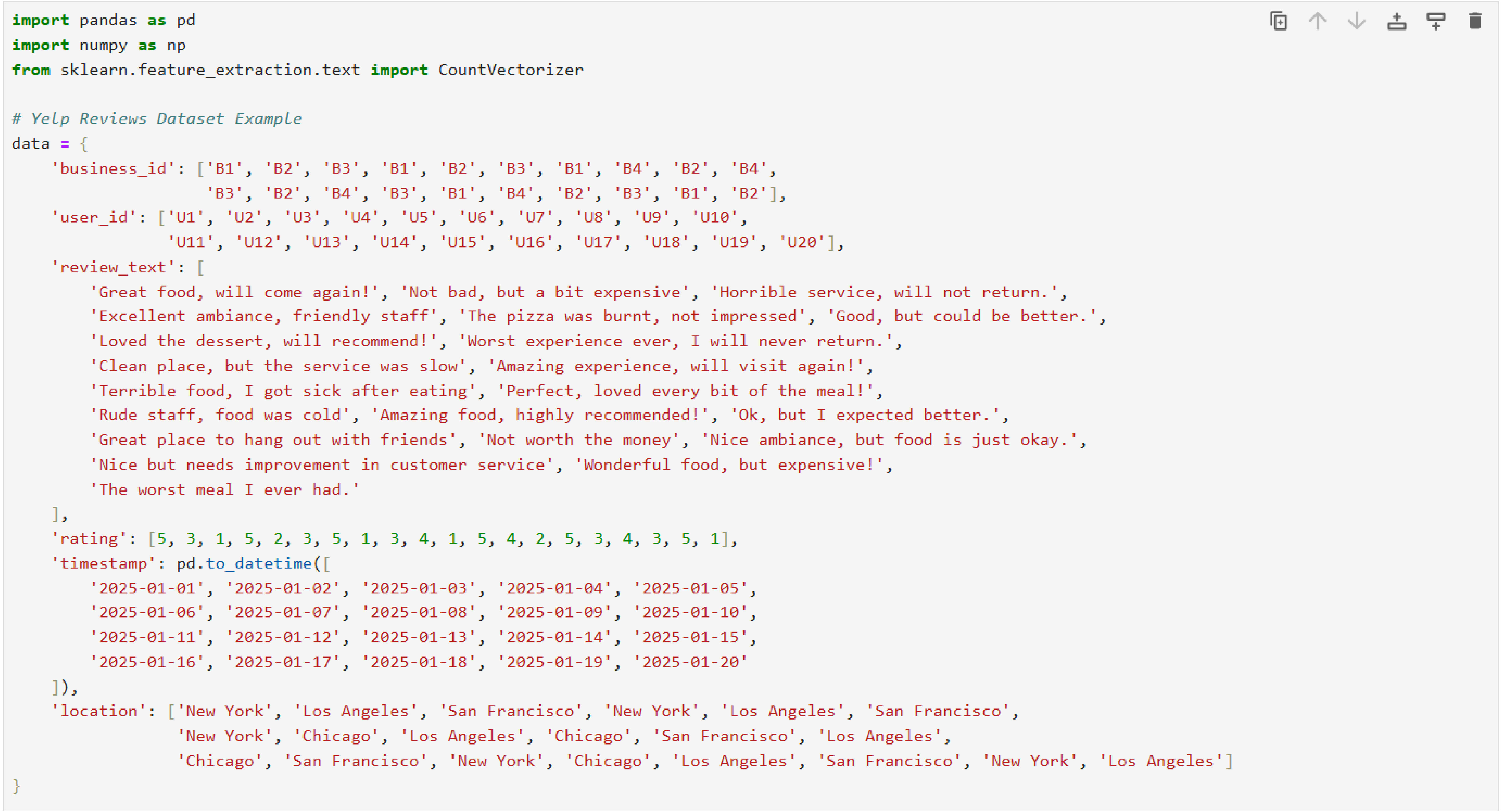
Class:- CS2

PRN:- 202401040363

Roll No:- CS2-39

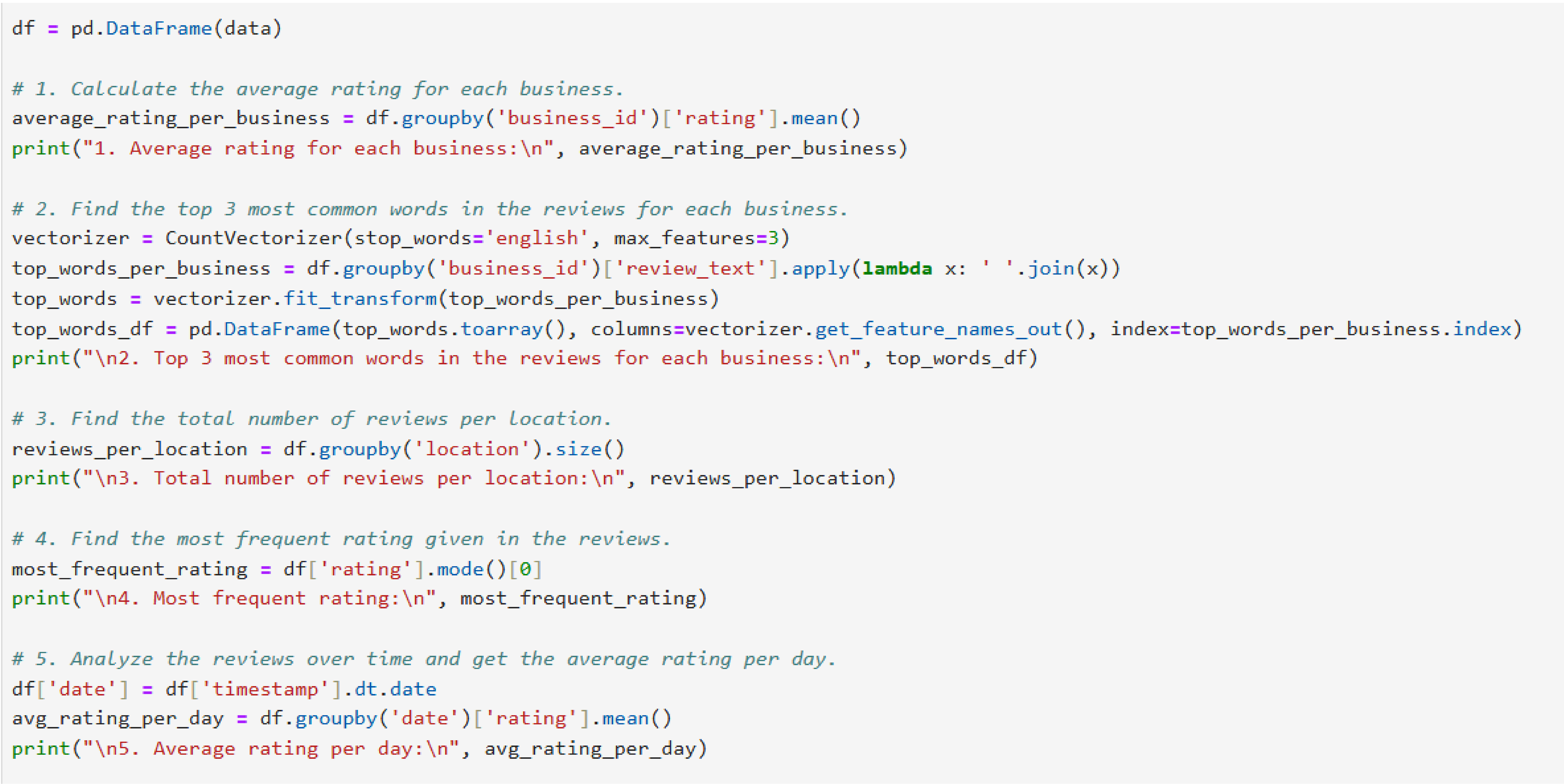
Yelp Reviews

Dataset:-

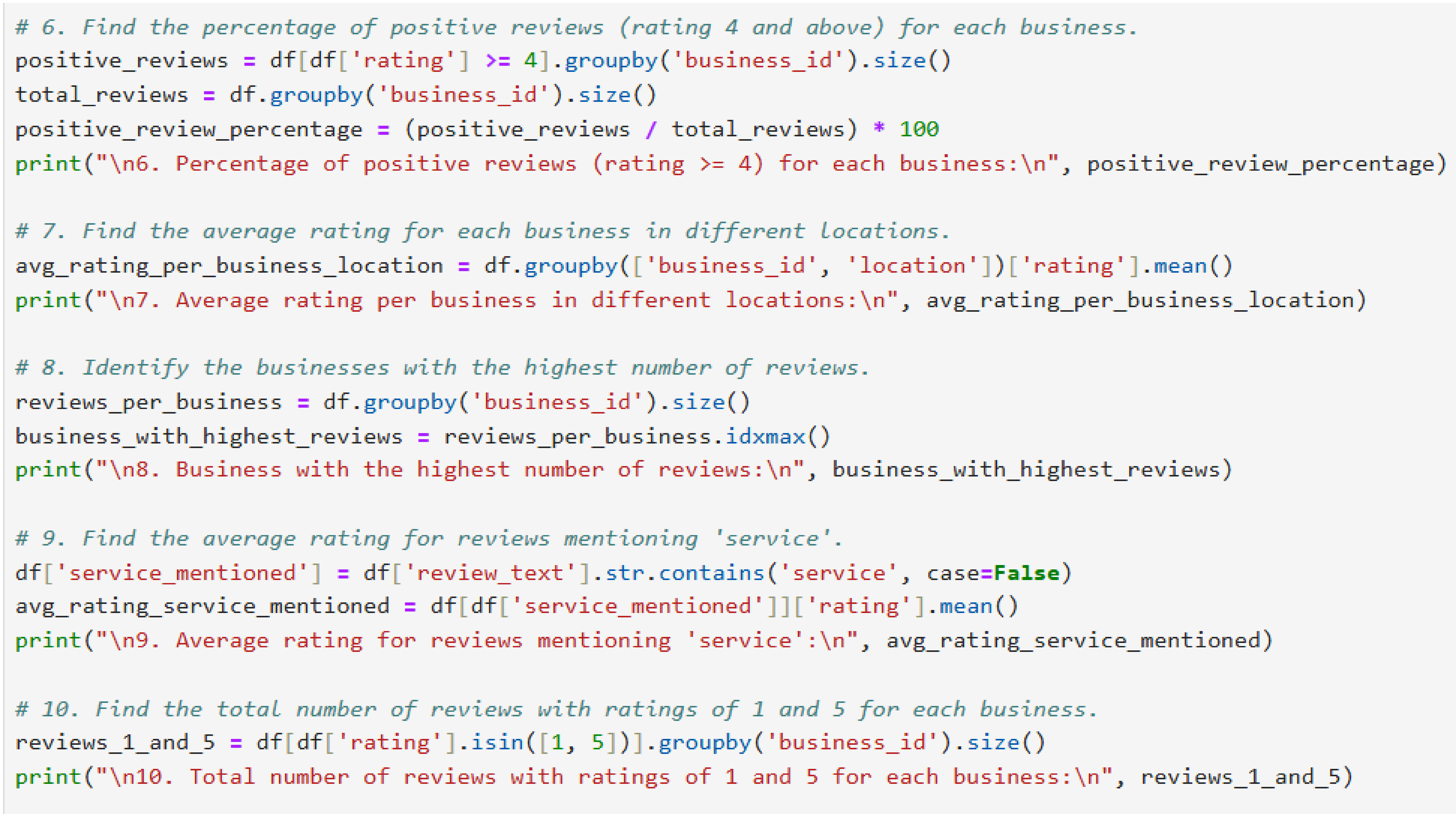


```python
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer

# Yelp Reviews Dataset Example
data = {
    'business_id': ['B1', 'B2', 'B3', 'B1', 'B2', 'B3', 'B1', 'B4', 'B2', 'B4',
                    'B3', 'B2', 'B4', 'B3', 'B1', 'B4', 'B2', 'B3', 'B1', 'B2'],
    'user_id': ['U1', 'U2', 'U3', 'U4', 'U5', 'U6', 'U7', 'U8', 'U9', 'U10',
                'U11', 'U12', 'U13', 'U14', 'U15', 'U16', 'U17', 'U18', 'U19', 'U20'],
    'review_text': [
        'Great food, will come again!', 'Not bad, but a bit expensive', 'Horrible service, will not return.',
        'Excellent ambiance, friendly staff', 'The pizza was burnt, not impressed', 'Good, but could be better.',
        'Loved the dessert, will recommend!', 'Worst experience ever, I will never return.',
        'Clean place, but the service was slow', 'Amazing experience, will visit again!',
        'Terrible food, I got sick after eating', 'Perfect, loved every bit of the meal!',
        'Rude staff, food was cold', 'Amazing food, highly recommended!', 'Ok, but I expected better.',
        'Great place to hang out with friends', 'Not worth the money', 'Nice ambiance, but food is just okay.',
        'Nice but needs improvement in customer service', 'Wonderful food, but expensive!',
        'The worst meal I ever had.'
    ],
    'rating': [5, 3, 1, 5, 2, 3, 5, 1, 3, 4, 1, 5, 4, 2, 5, 3, 4, 3, 5, 1],
    'timestamp': pd.to_datetime([
        '2025-01-01', '2025-01-02', '2025-01-03', '2025-01-04', '2025-01-05',
        '2025-01-06', '2025-01-07', '2025-01-08', '2025-01-09', '2025-01-10',
        '2025-01-11', '2025-01-12', '2025-01-13', '2025-01-14', '2025-01-15',
        '2025-01-16', '2025-01-17', '2025-01-18', '2025-01-19', '2025-01-20'
    ]),
    'location': ['New York', 'Los Angeles', 'San Francisco', 'New York', 'Los Angeles', 'San Francisco',
                 'New York', 'Chicago', 'Los Angeles', 'Chicago', 'San Francisco', 'Los Angeles',
                 'Chicago', 'San Francisco', 'New York', 'Chicago', 'Los Angeles', 'San Francisco', 'New York', 'Los Angeles']
}
```

Problem Statements:-



```python
df = pd.DataFrame(data)

# 1. Calculate the average rating for each business.
average_rating_per_business = df.groupby('business_id')['rating'].mean()
print("1. Average rating for each business:\n", average_rating_per_business)

# 2. Find the top 3 most common words in the reviews for each business.
vectorizer = CountVectorizer(stop_words='english', max_features=3)
top_words_per_business = df.groupby('business_id')['review_text'].apply(lambda x: ' '.join(x))
top_words = vectorizer.fit_transform(top_words_per_business)
top_words_df = pd.DataFrame(top_words.toarray(), columns=vectorizer.get_feature_names_out(), index=top_words_per_business.index)
print("\n2. Top 3 most common words in the reviews for each business:\n", top_words_df)

# 3. Find the total number of reviews per location.
reviews_per_location = df.groupby('location').size()
print("\n3. Total number of reviews per location:\n", reviews_per_location)

# 4. Find the most frequent rating given in the reviews.
most_frequent_rating = df['rating'].mode()[0]
print("\n4. Most frequent rating:\n", most_frequent_rating)

# 5. Analyze the reviews over time and get the average rating per day.
df['date'] = df['timestamp'].dt.date
avg_rating_per_day = df.groupby('date')['rating'].mean()
print("\n5. Average rating per day:\n", avg_rating_per_day)
```



```python
# 6. Find the percentage of positive reviews (rating 4 and above) for each business.
positive_reviews = df[df['rating'] >= 4].groupby('business_id').size()
total_reviews = df.groupby('business_id').size()
positive_review_percentage = (positive_reviews / total_reviews) * 100
print("\n6. Percentage of positive reviews (rating >= 4) for each business:\n", positive_review_percentage)

# 7. Find the average rating for each business in different locations.
avg_rating_per_business_location = df.groupby(['business_id', 'location'])['rating'].mean()
print("\n7. Average rating per business in different locations:\n", avg_rating_per_business_location)

# 8. Identify the businesses with the highest number of reviews.
reviews_per_business = df.groupby('business_id').size()
business_with_highest_reviews = reviews_per_business.idxmax()
print("\n8. Business with the highest number of reviews:\n", business_with_highest_reviews)

# 9. Find the average rating for reviews mentioning 'service'.
df['service_mentioned'] = df['review_text'].str.contains('service', case=False)
avg_rating_service_mentioned = df[df['service_mentioned']]['rating'].mean()
print("\n9. Average rating for reviews mentioning 'service':\n", avg_rating_service_mentioned)

# 10. Find the total number of reviews with ratings of 1 and 5 for each business.
reviews_1_and_5 = df[df['rating'].isin([1, 5])].groupby('business_id').size()
print("\n10. Total number of reviews with ratings of 1 and 5 for each business:\n", reviews_1_and_5)
```

```python
# 11. Find the average review length for each business.
df['review_length'] = df['review_text'].apply(len)
avg_review_length_per_business = df.groupby('business_id')['review_length'].mean()
print("\n11. Average review length for each business:\n", avg_review_length_per_business)

# 12. Find businesses that received only negative reviews (rating < 3).
negative_reviews = df[df['rating'] < 3].groupby('business_id').size()
businesses_with_negative_reviews = negative_reviews[negative_reviews == negative_reviews.max()]
print("\n12. Businesses that received only negative reviews:\n", businesses_with_negative_reviews)

# 13. Calculate the rating distribution (percentage of each rating) for each business.
rating_distribution_per_business = df.groupby('business_id')['rating'].value_counts(normalize=True) * 100
print("\n13. Rating distribution (percentage of each rating) for each business:\n", rating_distribution_per_business)

# 14. Find businesses with the lowest average rating.
lowest_avg_rating = df.groupby('business_id')['rating'].mean().idxmin()
print("\n14. Business with the lowest average rating:\n", lowest_avg_rating)

# 15. Count the number of reviews per rating category (e.g., 1, 2, 3, 4, 5).
reviews_per_rating = df.groupby('rating').size()
print("\n15. Number of reviews per rating category:\n", reviews_per_rating)
```

```python
# 16. Find the businesses with the highest average rating.
highest_avg_rating = df.groupby('business_id')['rating'].mean().idxmax()
print("\n16. Business with the highest average rating:\n", highest_avg_rating)

# 17. Find the review that contains the word 'amazing' and the corresponding rating.
df['amazing_mentioned'] = df['review_text'].str.contains('amazing', case=False)
amazing_reviews = df[df['amazing_mentioned']]
print("\n17. Reviews mentioning 'amazing' and their corresponding ratings:\n", amazing_reviews[['review_text', 'rating']])

# 18. Find the top 3 users who have written the most reviews.
reviews_per_user = df.groupby('user_id').size()
top_3_users = reviews_per_user.nlargest(3)
print("\n18. Top 3 users who have written the most reviews:\n", top_3_users)

# 19. Calculate the average rating per user across all businesses.
avg_rating_per_user = df.groupby('user_id')['rating'].mean()
print("\n19. Average rating per user across all businesses:\n", avg_rating_per_user)

# 20. Find the number of unique businesses reviewed by each user.
unique_businesses_per_user = df.groupby('user_id')['business_id'].nunique()
print("\n20. Number of unique businesses reviewed by each user:\n", unique_businesses_per_user)
```