Name : Atharva Kadam

Div : CS7

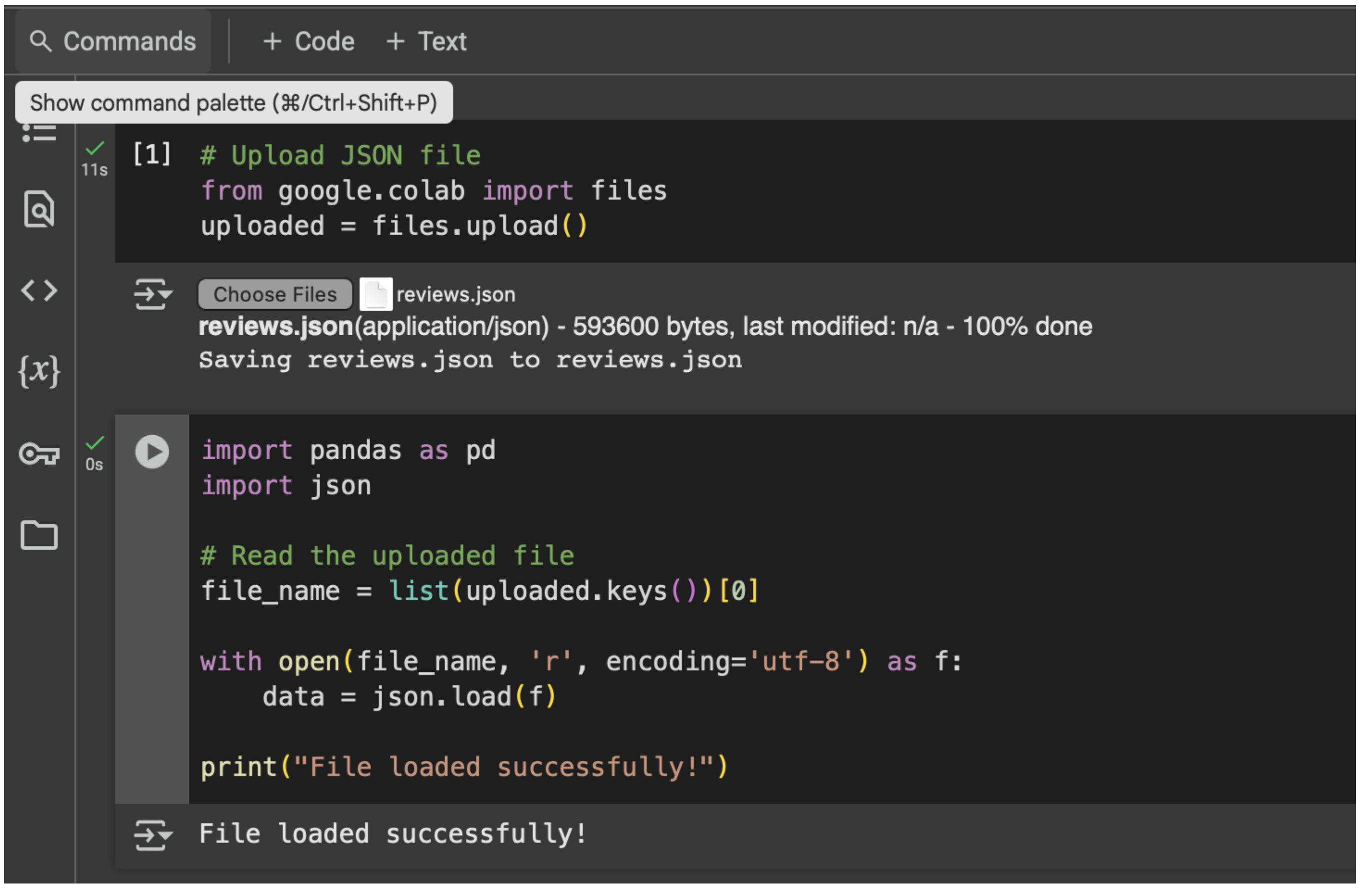
Roll no : 62

PRN : 202401110067

Batch : CS73

Colab link: https://colab.research.google.com/drive/1M-qFRubPRXH0PKnSUwRxnBlIbKidnpcW

URL Link: https://www.kaggle.com/datasets/fnbalves/paper-reviews-data-set

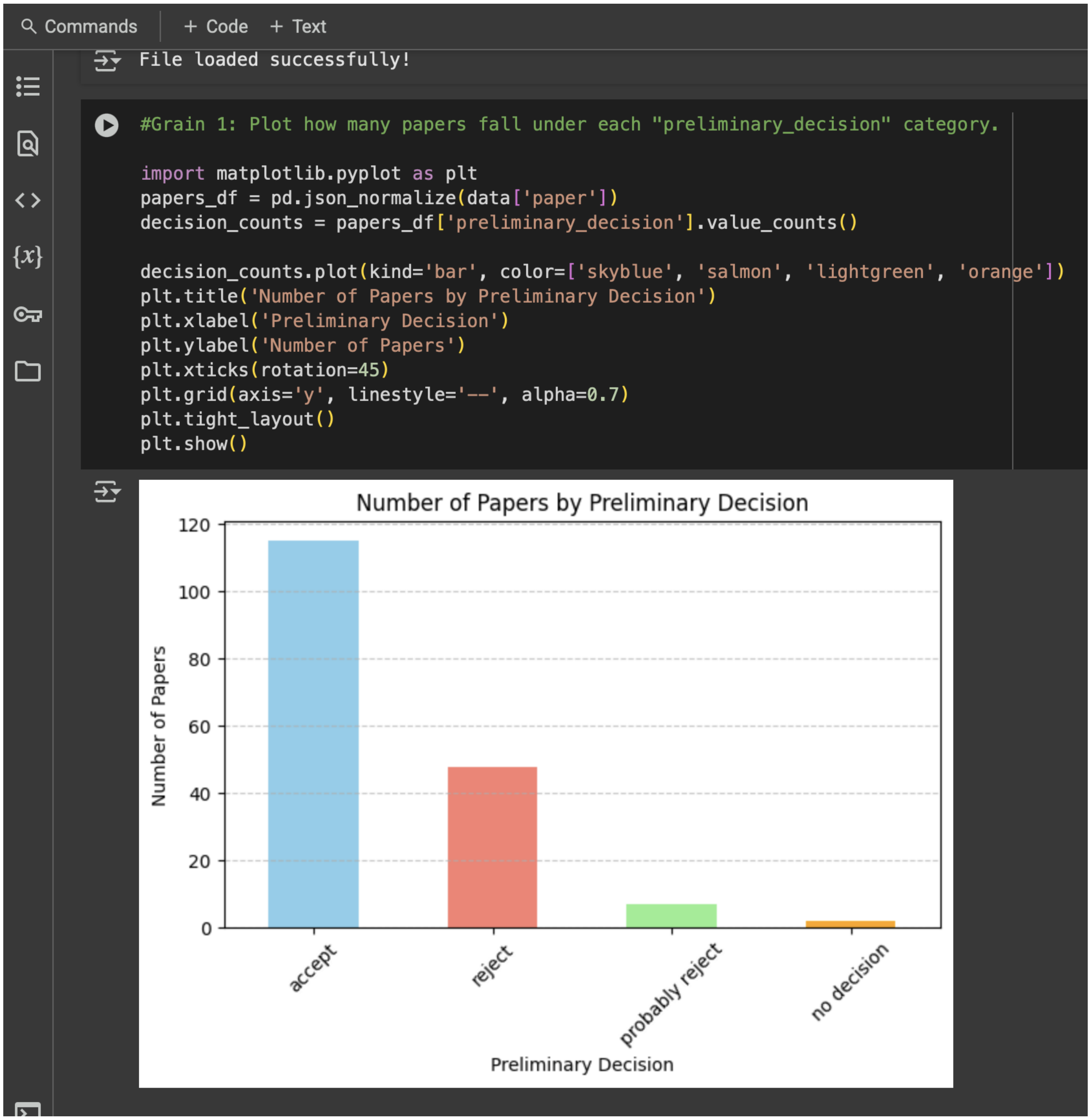


```
[1] # Upload JSON file
    from google.colab import files
    uploaded = files.upload()
```

```
reviews.json
reviews.json(application/json) - 593600 bytes, last modified: n/a - 100% done
Saving reviews.json to reviews.json
```

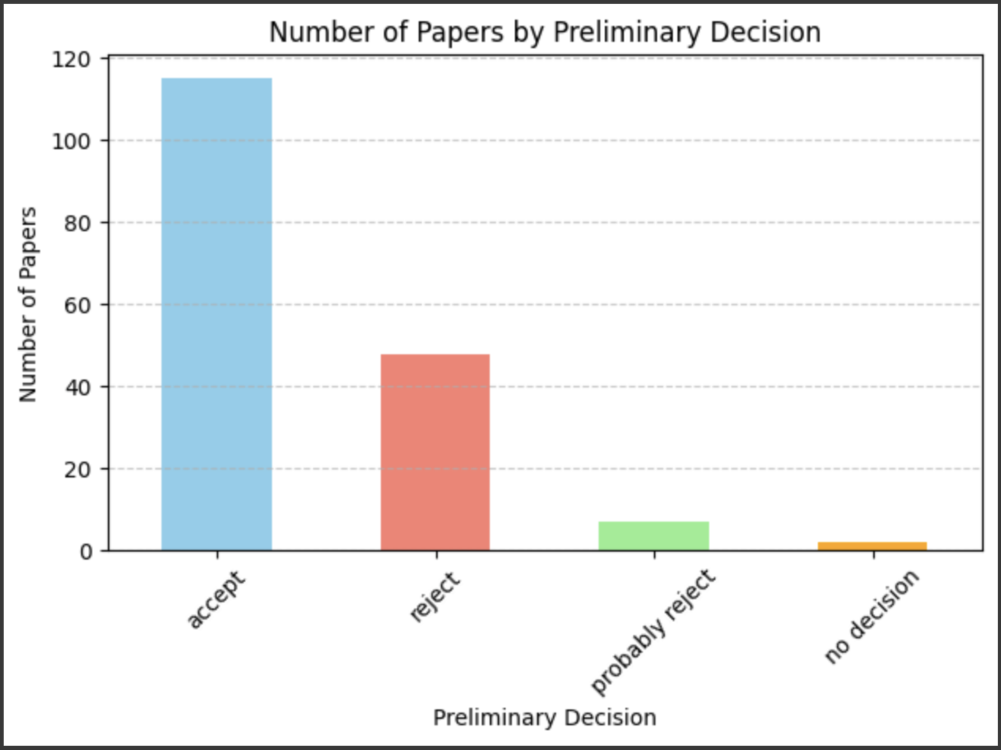
```
import pandas as pd
import json

# Read the uploaded file
file_name = list(uploaded.keys())[0]

with open(file_name, 'r', encoding='utf-8') as f:
    data = json.load(f)

print("File loaded successfully!")
```

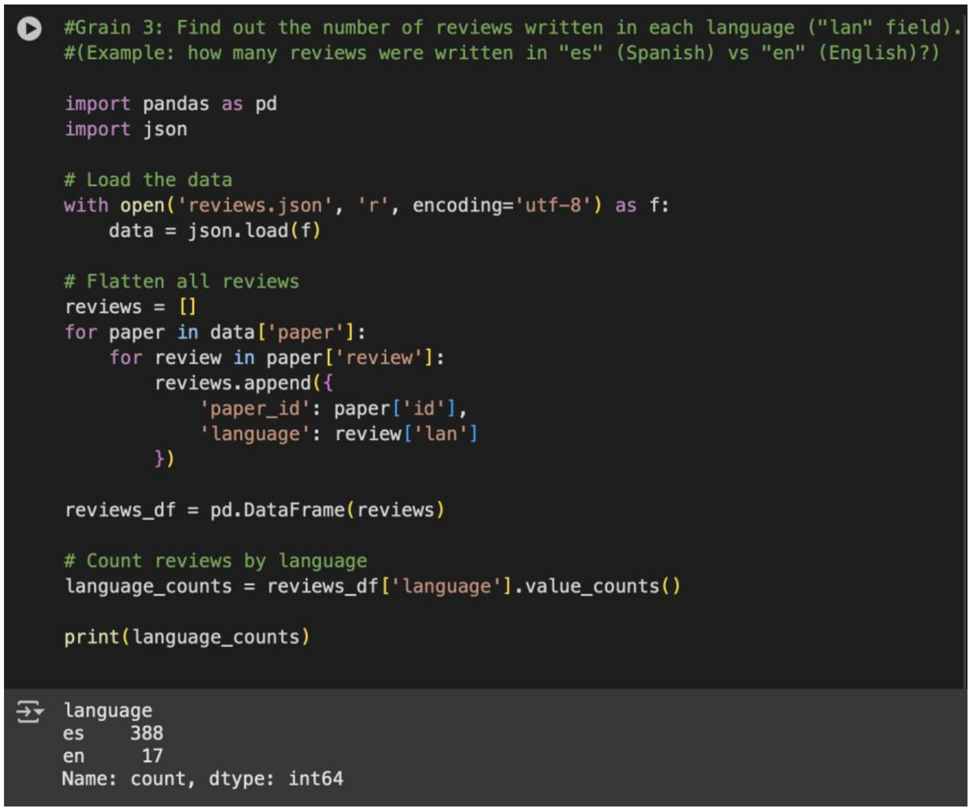
```
File loaded successfully!
```



```
File loaded successfully!
```

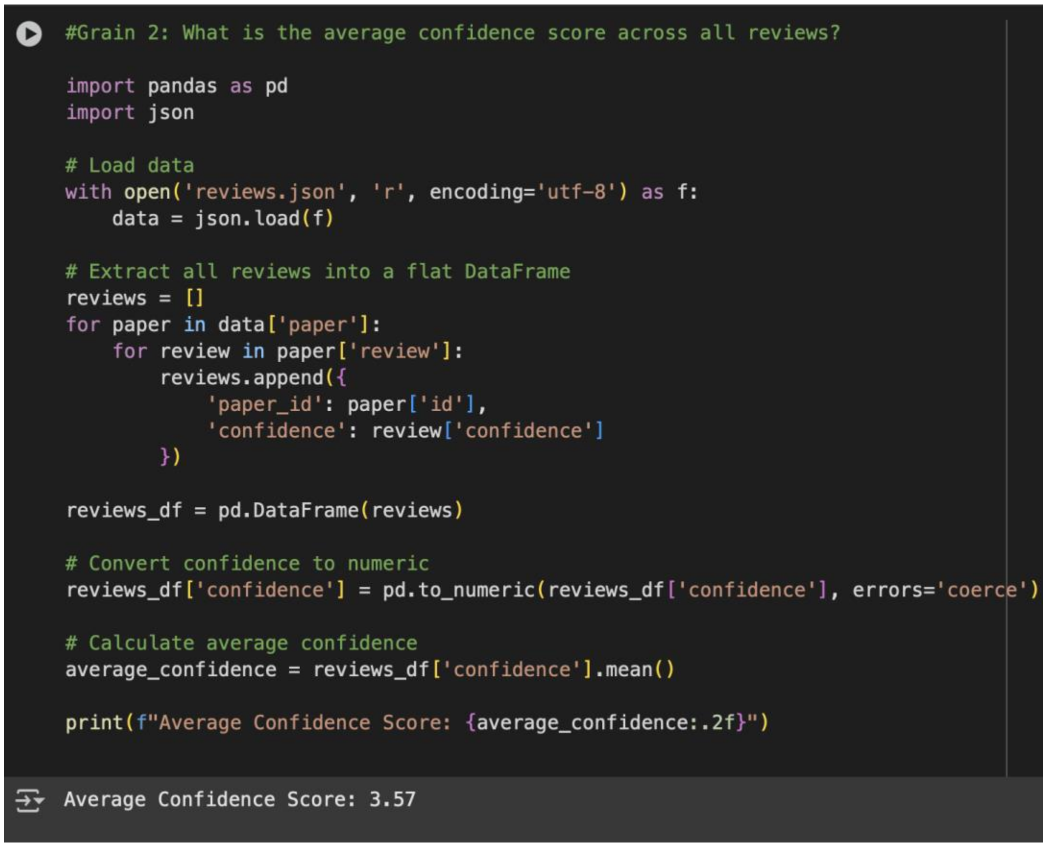
```
#Grain 1: Plot how many papers fall under each "preliminary_decision" category.

import matplotlib.pyplot as plt
papers_df = pd.json_normalize(data['paper'])
decision_counts = papers_df['preliminary_decision'].value_counts()

decision_counts.plot(kind='bar', color=['skyblue', 'salmon', 'lightgreen', 'orange'])
plt.title('Number of Papers by Preliminary Decision')
plt.xlabel('Preliminary Decision')
plt.ylabel('Number of Papers')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```





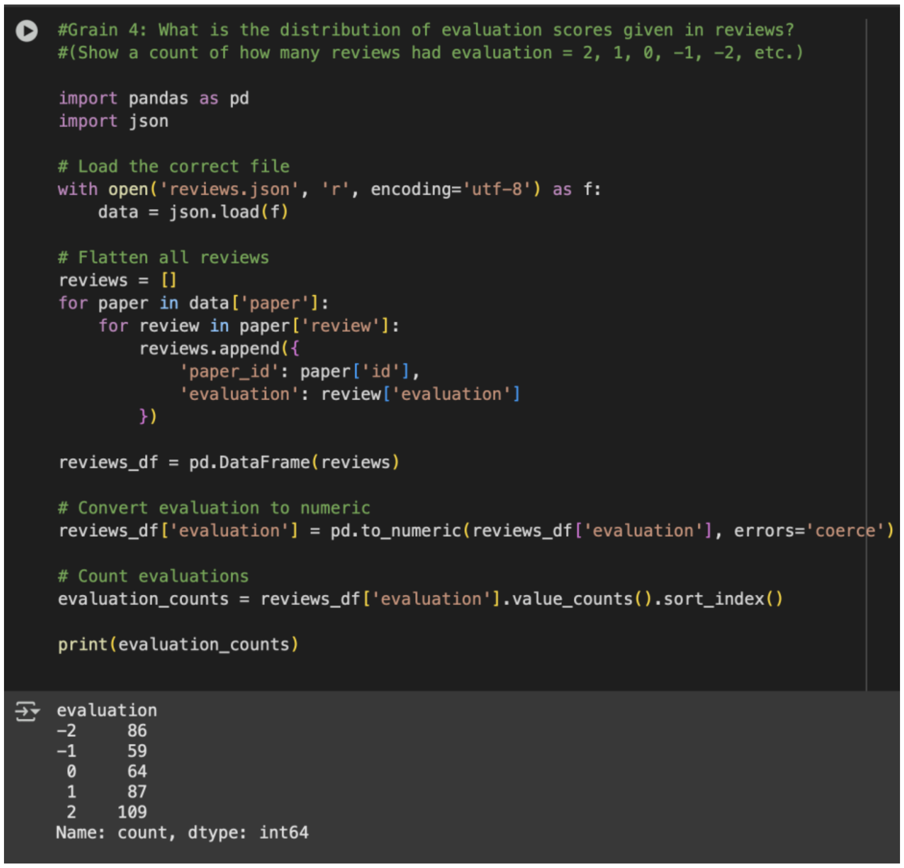
```
#Grain 3: Find out the number of reviews written in each language ("lan" field).
#(Example: how many reviews were written in "es" (Spanish) vs "en" (English)?)

import pandas as pd
import json

# Load the data
with open('reviews.json', 'r', encoding='utf-8') as f:
    data = json.load(f)

# Flatten all reviews
reviews = []
for paper in data['paper']:
    for review in paper['review']:
        reviews.append({
            'paper_id': paper['id'],
            'language': review['lan']
        })

reviews_df = pd.DataFrame(reviews)

# Count reviews by language
language_counts = reviews_df['language'].value_counts()

print(language_counts)
```

```
language
es    388
en     17
Name: count, dtype: int64
```



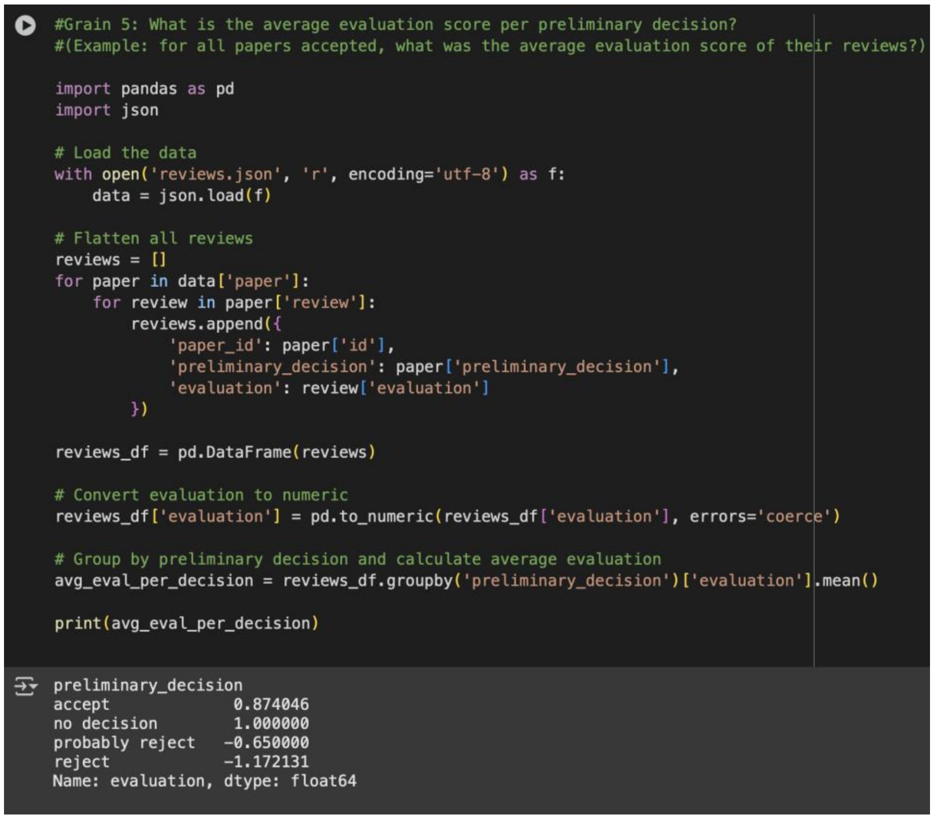
```
#Grain 2: What is the average confidence score across all reviews?

import pandas as pd
import json

# Load data
with open('reviews.json', 'r', encoding='utf-8') as f:
    data = json.load(f)

# Extract all reviews into a flat DataFrame
reviews = []
for paper in data['paper']:
    for review in paper['review']:
        reviews.append({
            'paper_id': paper['id'],
            'confidence': review['confidence']
        })

reviews_df = pd.DataFrame(reviews)

# Convert confidence to numeric
reviews_df['confidence'] = pd.to_numeric(reviews_df['confidence'], errors='coerce')

# Calculate average confidence
average_confidence = reviews_df['confidence'].mean()

print(f"Average Confidence Score: {average_confidence:.2f}")
```

```
Average Confidence Score: 3.57
```



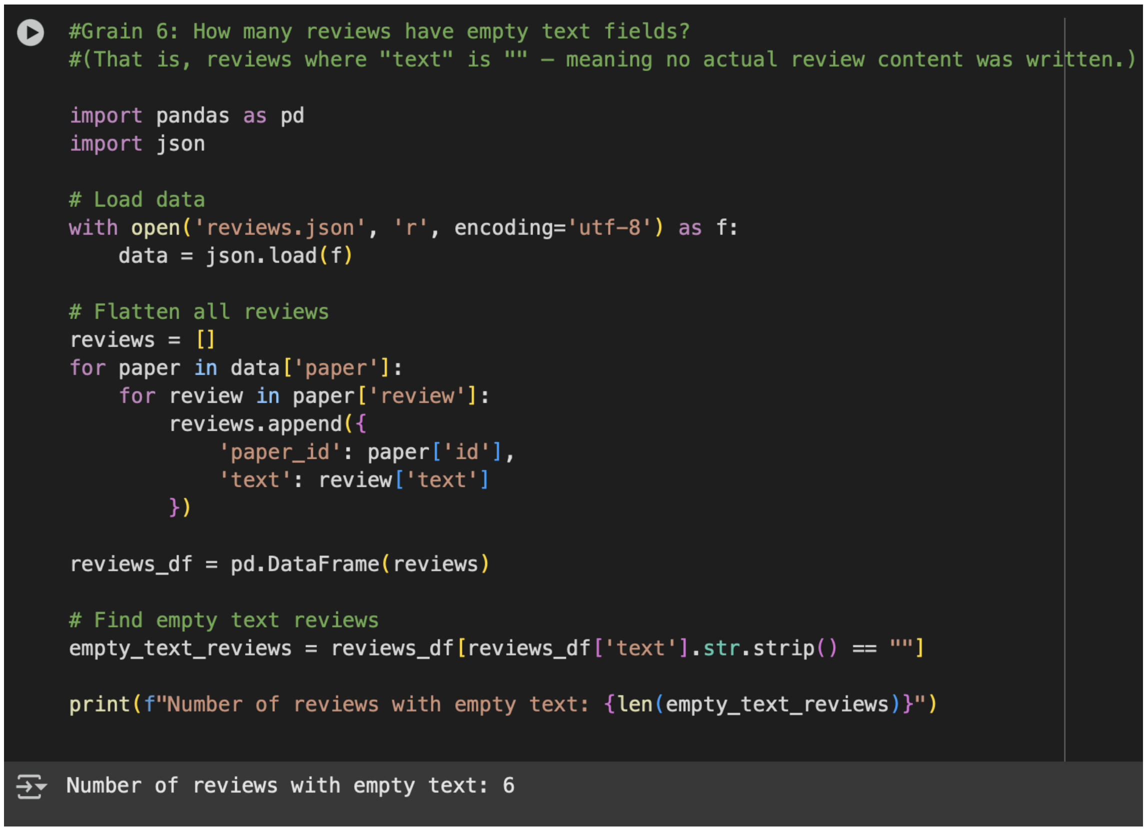
```
#Grain 4: What is the distribution of evaluation scores given in reviews?
#(Show a count of how many reviews had evaluation = 2, 1, 0, -1, -2, etc.)

import pandas as pd
import json

# Load the correct file
with open('reviews.json', 'r', encoding='utf-8') as f:
    data = json.load(f)

# Flatten all reviews
reviews = []
for paper in data['paper']:
    for review in paper['review']:
        reviews.append({
            'paper_id': paper['id'],
            'evaluation': review['evaluation']
        })

reviews_df = pd.DataFrame(reviews)

# Convert evaluation to numeric
reviews_df['evaluation'] = pd.to_numeric(reviews_df['evaluation'], errors='coerce')

# Count evaluations
evaluation_counts = reviews_df['evaluation'].value_counts().sort_index()

print(evaluation_counts)
```

```
evaluation
-2     86
-1     59
 0     64
 1     87
 2    109
Name: count, dtype: int64
```



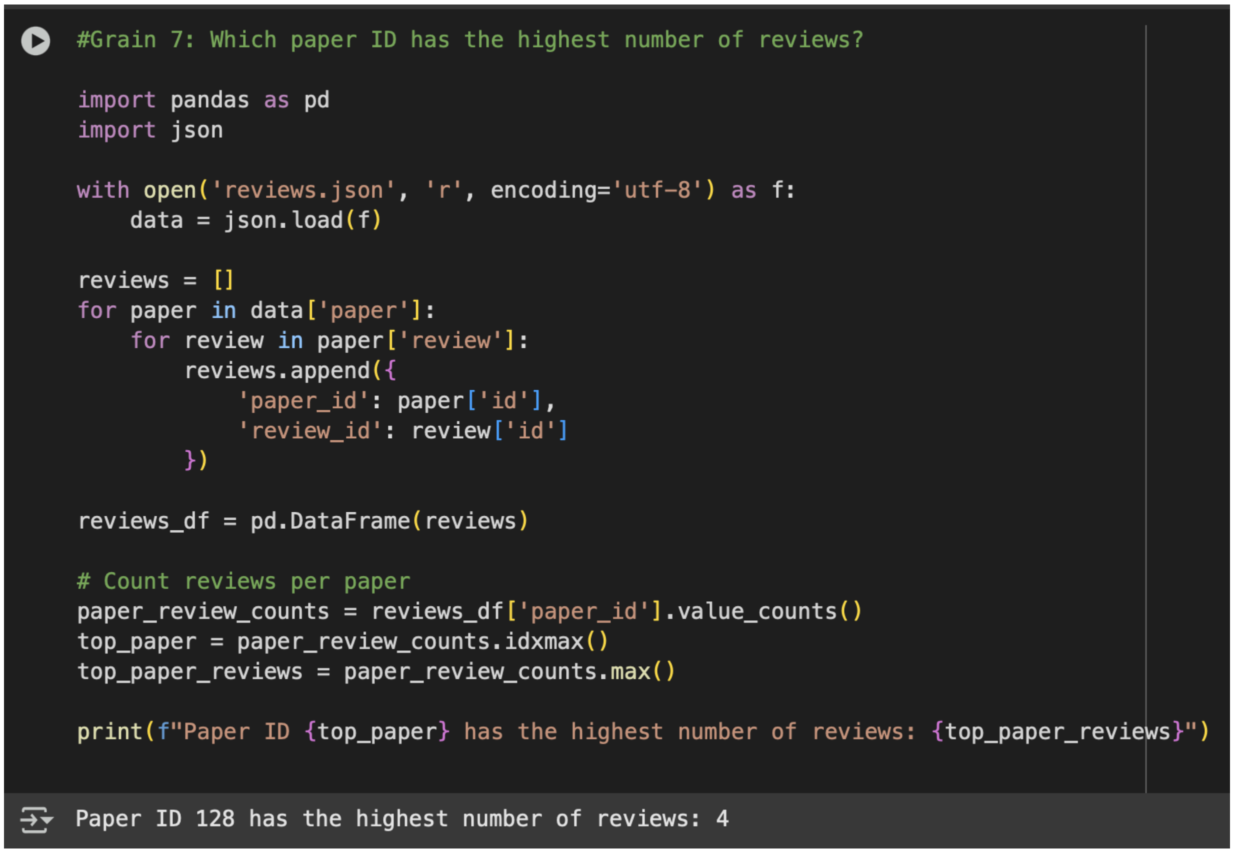
```
#Grain 5: What is the average evaluation score per preliminary decision?
#(Example: for all papers accepted, what was the average evaluation score of their reviews?)

import pandas as pd
import json

# Load the data
with open('reviews.json', 'r', encoding='utf-8') as f:
    data = json.load(f)

# Flatten all reviews
reviews = []
for paper in data['paper']:
    for review in paper['review']:
        reviews.append({
            'paper_id': paper['id'],
            'preliminary_decision': paper['preliminary_decision'],
            'evaluation': review['evaluation']
        })

reviews_df = pd.DataFrame(reviews)

# Convert evaluation to numeric
reviews_df['evaluation'] = pd.to_numeric(reviews_df['evaluation'], errors='coerce')

# Group by preliminary decision and calculate average evaluation
avg_eval_per_decision = reviews_df.groupby('preliminary_decision')['evaluation'].mean()

print(avg_eval_per_decision)
```

```
preliminary_decision
accept             0.874046
no decision        1.000000
probably reject   -0.650000
reject            -1.172131
Name: evaluation, dtype: float64
```



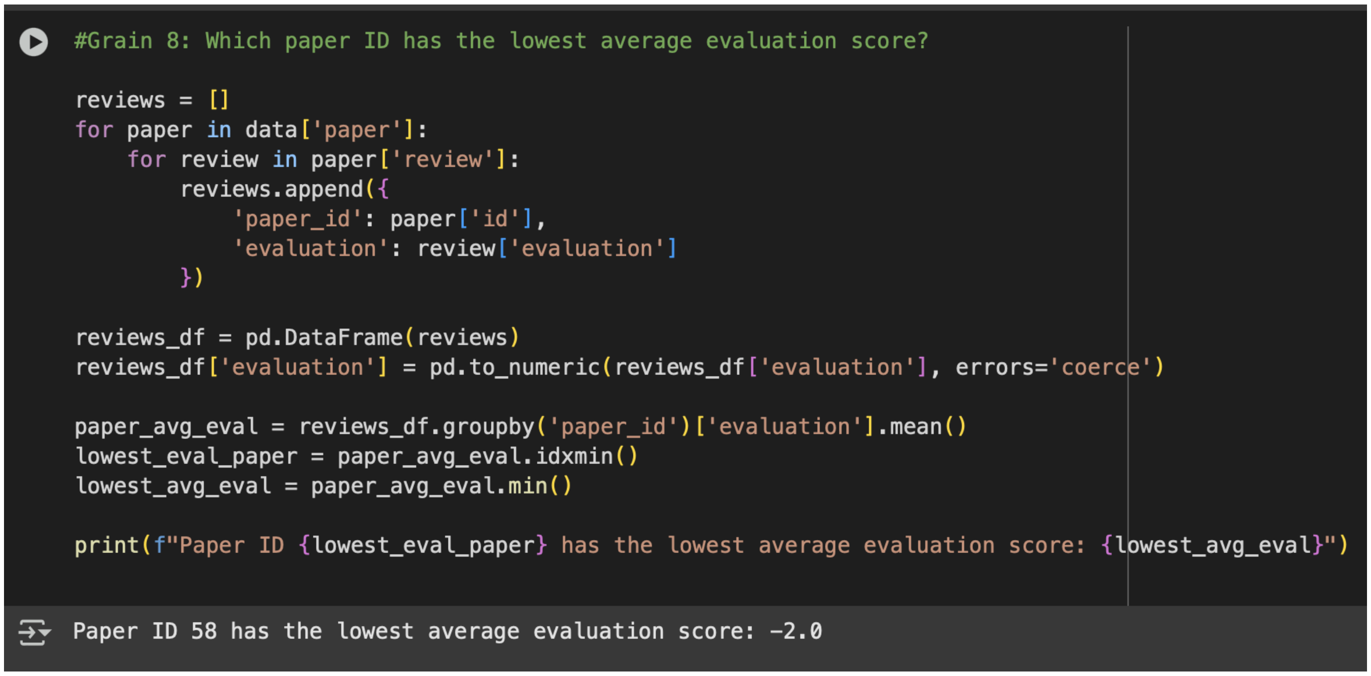
```python
#Grain 6: How many reviews have empty text fields?
#(That is, reviews where "text" is "" — meaning no actual review content was written.)

import pandas as pd
import json

# Load data
with open('reviews.json', 'r', encoding='utf-8') as f:
    data = json.load(f)

# Flatten all reviews
reviews = []
for paper in data['paper']:
    for review in paper['review']:
        reviews.append({
            'paper_id': paper['id'],
            'text': review['text']
        })

reviews_df = pd.DataFrame(reviews)

# Find empty text reviews
empty_text_reviews = reviews_df[reviews_df['text'].str.strip() == ""]

print(f"Number of reviews with empty text: {len(empty_text_reviews)}")
```

```
Number of reviews with empty text: 6
```



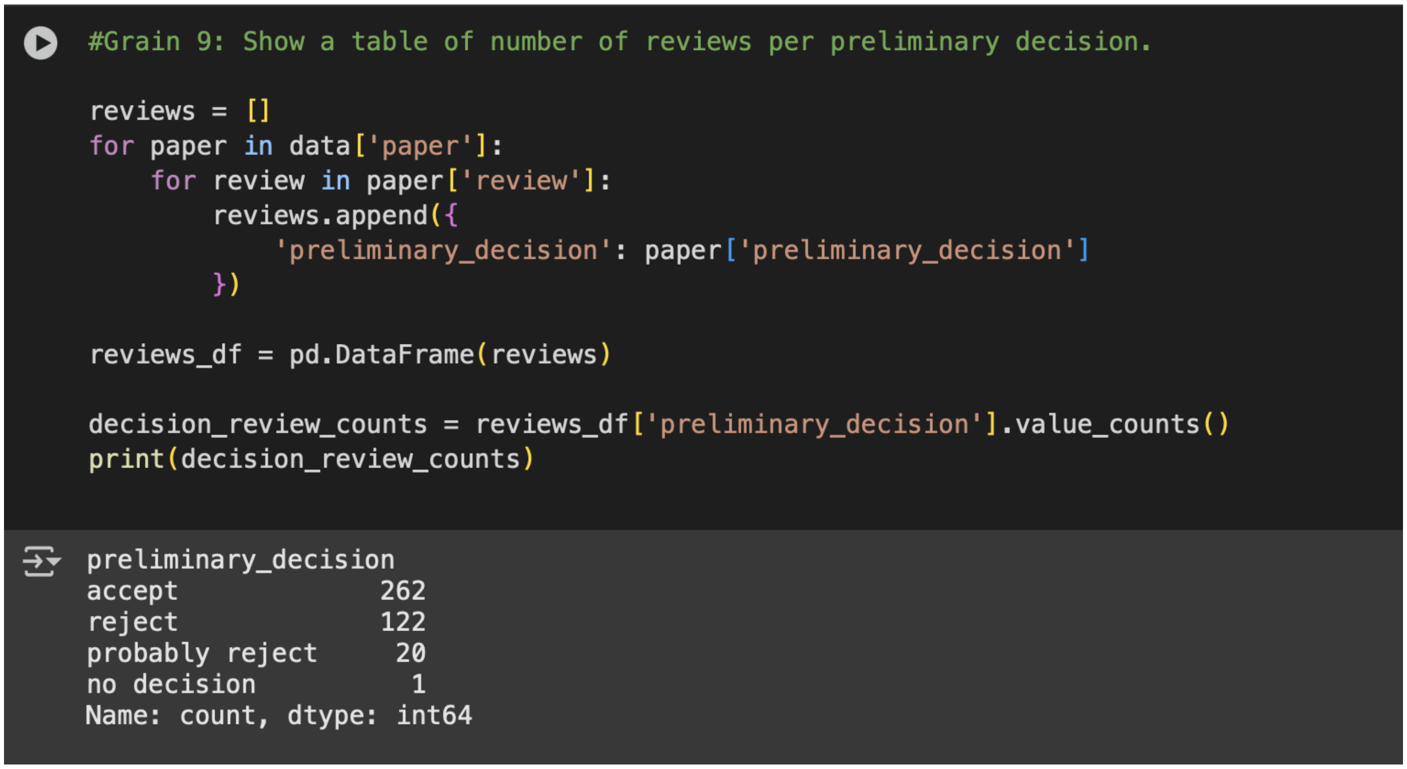
```python
#Grain 7: Which paper ID has the highest number of reviews?

import pandas as pd
import json

with open('reviews.json', 'r', encoding='utf-8') as f:
    data = json.load(f)

reviews = []
for paper in data['paper']:
    for review in paper['review']:
        reviews.append({
            'paper_id': paper['id'],
            'review_id': review['id']
        })

reviews_df = pd.DataFrame(reviews)

# Count reviews per paper
paper_review_counts = reviews_df['paper_id'].value_counts()
top_paper = paper_review_counts.idxmax()
top_paper_reviews = paper_review_counts.max()

print(f"Paper ID {top_paper} has the highest number of reviews: {top_paper_reviews}")
```

```
Paper ID 128 has the highest number of reviews: 4
```



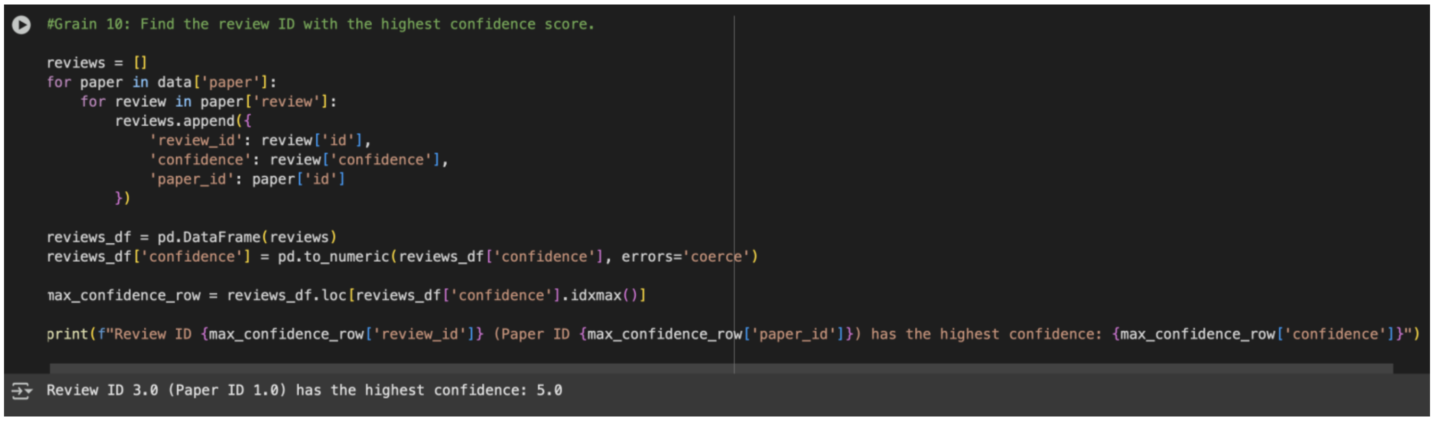
```python
#Grain 8: Which paper ID has the lowest average evaluation score?

reviews = []
for paper in data['paper']:
    for review in paper['review']:
        reviews.append({
            'paper_id': paper['id'],
            'evaluation': review['evaluation']
        })

reviews_df = pd.DataFrame(reviews)
reviews_df['evaluation'] = pd.to_numeric(reviews_df['evaluation'], errors='coerce')

paper_avg_eval = reviews_df.groupby('paper_id')['evaluation'].mean()
lowest_eval_paper = paper_avg_eval.idxmin()
lowest_avg_eval = paper_avg_eval.min()

print(f"Paper ID {lowest_eval_paper} has the lowest average evaluation score: {lowest_avg_eval}")
```

```
Paper ID 58 has the lowest average evaluation score: -2.0
```



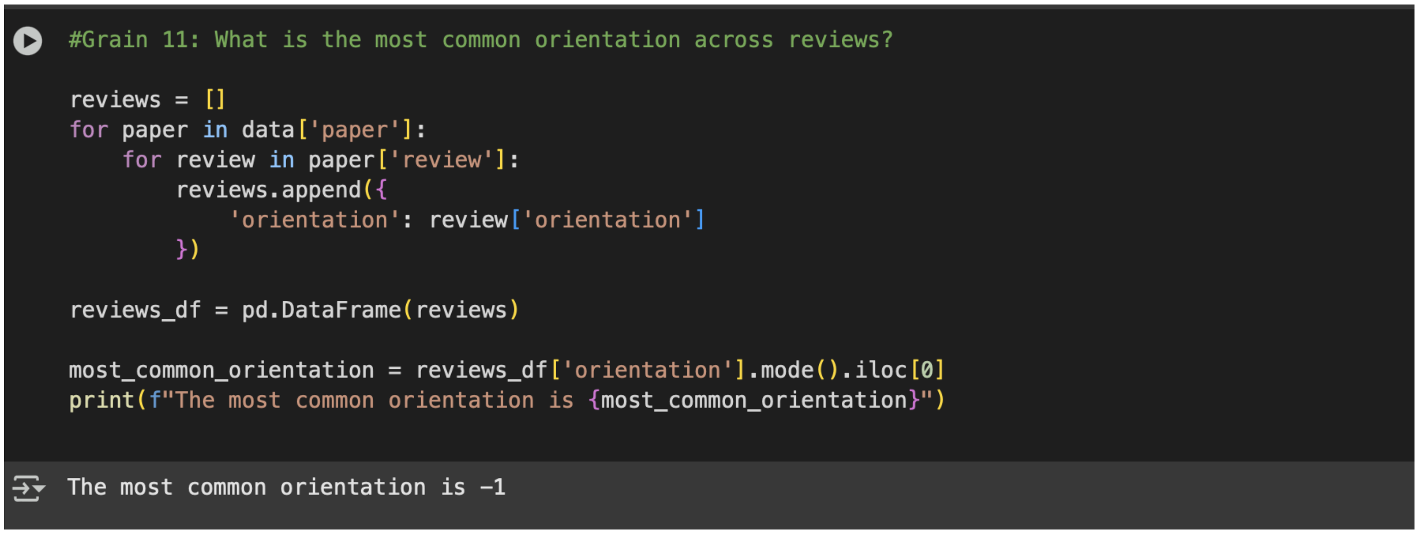
```python
#Grain 9: Show a table of number of reviews per preliminary decision.

reviews = []
for paper in data['paper']:
    for review in paper['review']:
        reviews.append({
            'preliminary_decision': paper['preliminary_decision']
        })

reviews_df = pd.DataFrame(reviews)

decision_review_counts = reviews_df['preliminary_decision'].value_counts()
print(decision_review_counts)
```

```
preliminary_decision
accept             262
reject             122
probably reject     20
no decision          1
Name: count, dtype: int64
```



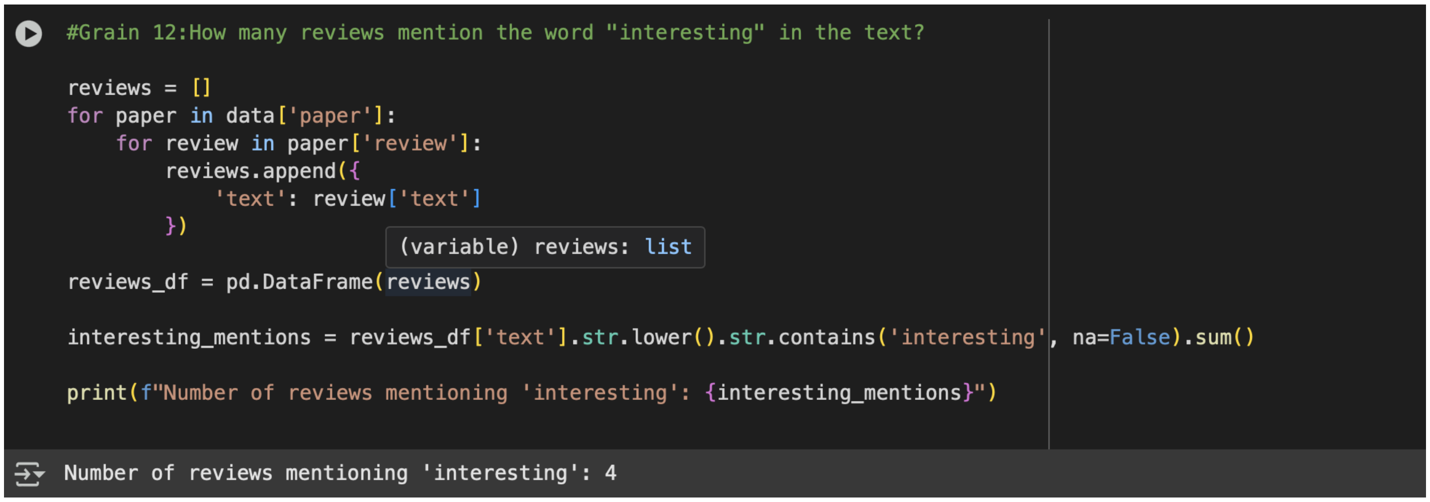
```
#Grain 10: Find the review ID with the highest confidence score.

reviews = []
for paper in data['paper']:
    for review in paper['review']:
        reviews.append({
            'review_id': review['id'],
            'confidence': review['confidence'],
            'paper_id': paper['id']
        })

reviews_df = pd.DataFrame(reviews)
reviews_df['confidence'] = pd.to_numeric(reviews_df['confidence'], errors='coerce')

max_confidence_row = reviews_df.loc[reviews_df['confidence'].idxmax()]

print(f"Review ID {max_confidence_row['review_id']} (Paper ID {max_confidence_row['paper_id']}) has the highest confidence: {max_confidence_row['confidence']}")
```

```
Review ID 3.0 (Paper ID 1.0) has the highest confidence: 5.0
```



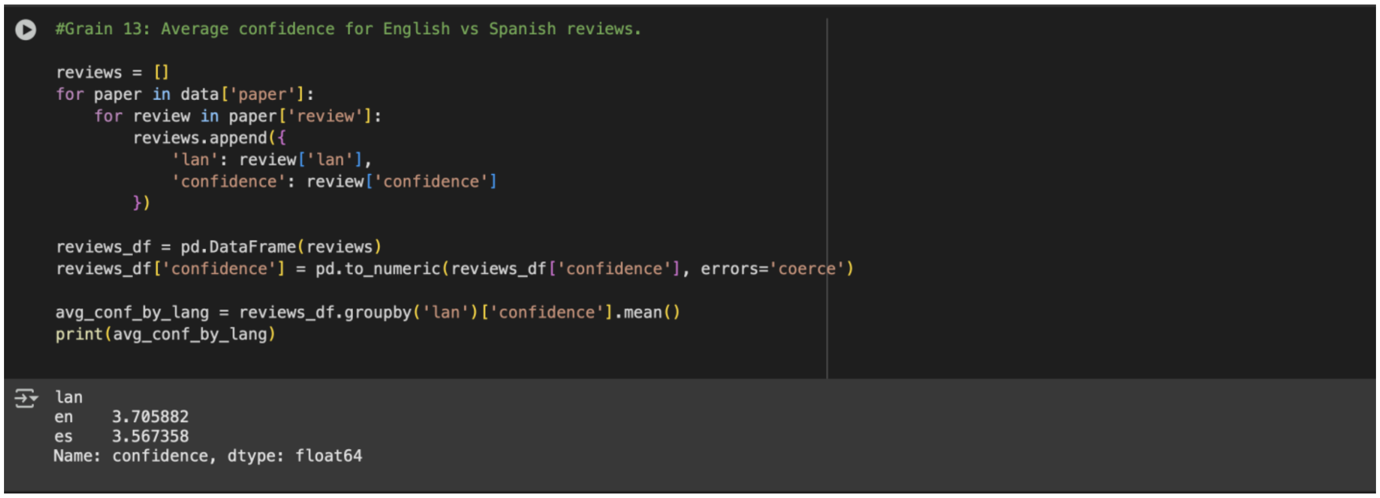
```
#Grain 11: What is the most common orientation across reviews?

reviews = []
for paper in data['paper']:
    for review in paper['review']:
        reviews.append({
            'orientation': review['orientation']
        })

reviews_df = pd.DataFrame(reviews)

most_common_orientation = reviews_df['orientation'].mode().iloc[0]
print(f"The most common orientation is {most_common_orientation}")
```

```
The most common orientation is -1
```



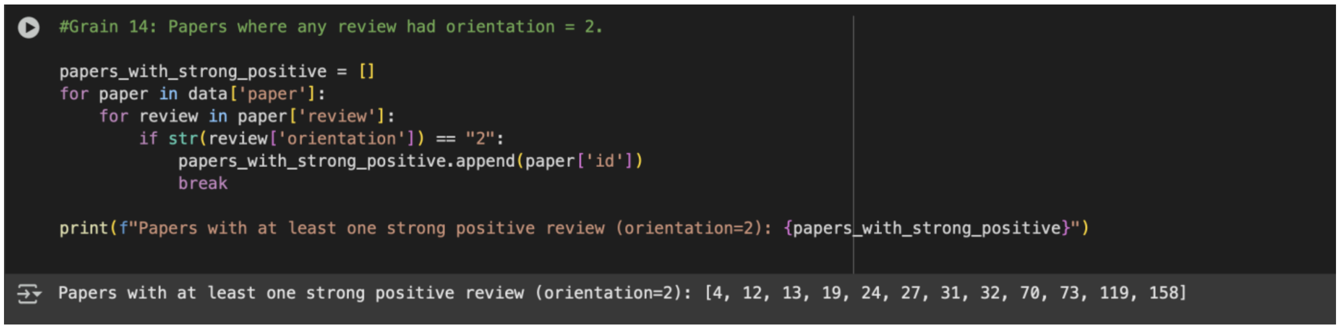
```
#Grain 12:How many reviews mention the word "interesting" in the text?

reviews = []
for paper in data['paper']:
    for review in paper['review']:
        reviews.append({
            'text': review['text']
        })

reviews_df = pd.DataFrame(reviews)

interesting_mentions = reviews_df['text'].str.lower().str.contains('interesting', na=False).sum()

print(f"Number of reviews mentioning 'interesting': {interesting_mentions}")
```

```
Number of reviews mentioning 'interesting': 4
```



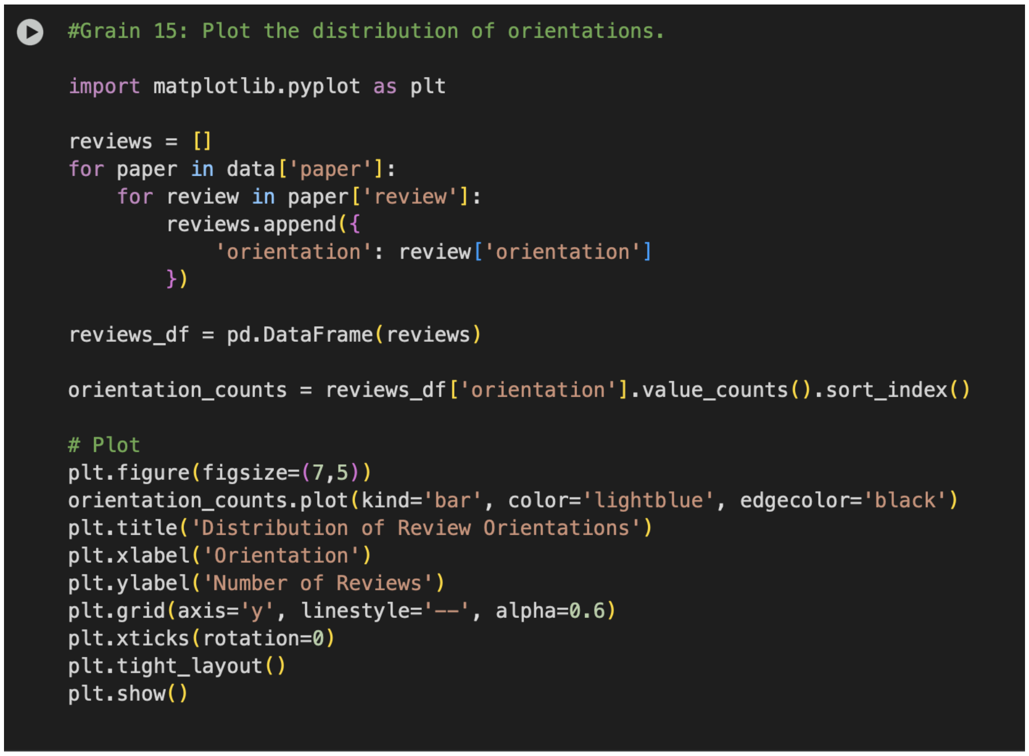
```
#Grain 13: Average confidence for English vs Spanish reviews.

reviews = []
for paper in data['paper']:
    for review in paper['review']:
        reviews.append({
            'lan': review['lan'],
            'confidence': review['confidence']
        })

reviews_df = pd.DataFrame(reviews)
reviews_df['confidence'] = pd.to_numeric(reviews_df['confidence'], errors='coerce')

avg_conf_by_lang = reviews_df.groupby('lan')['confidence'].mean()
print(avg_conf_by_lang)
```

```
lan
en    3.705882
es    3.567358
Name: confidence, dtype: float64
```

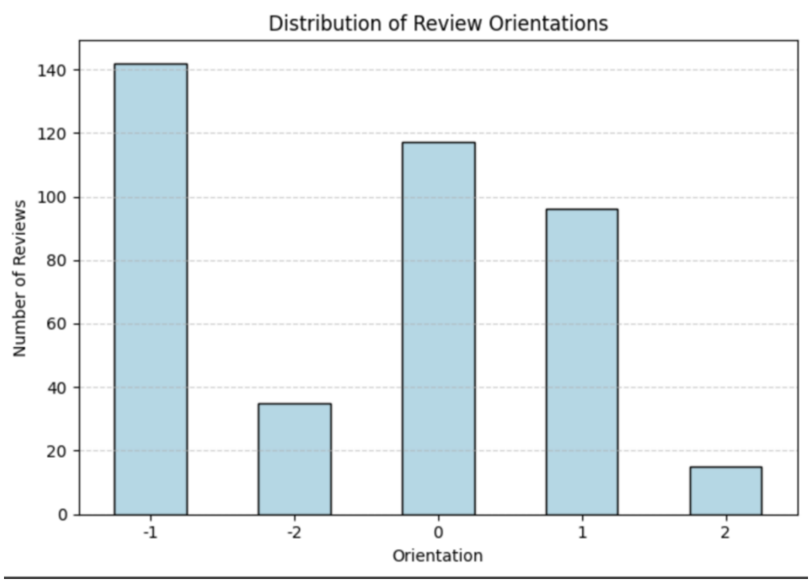


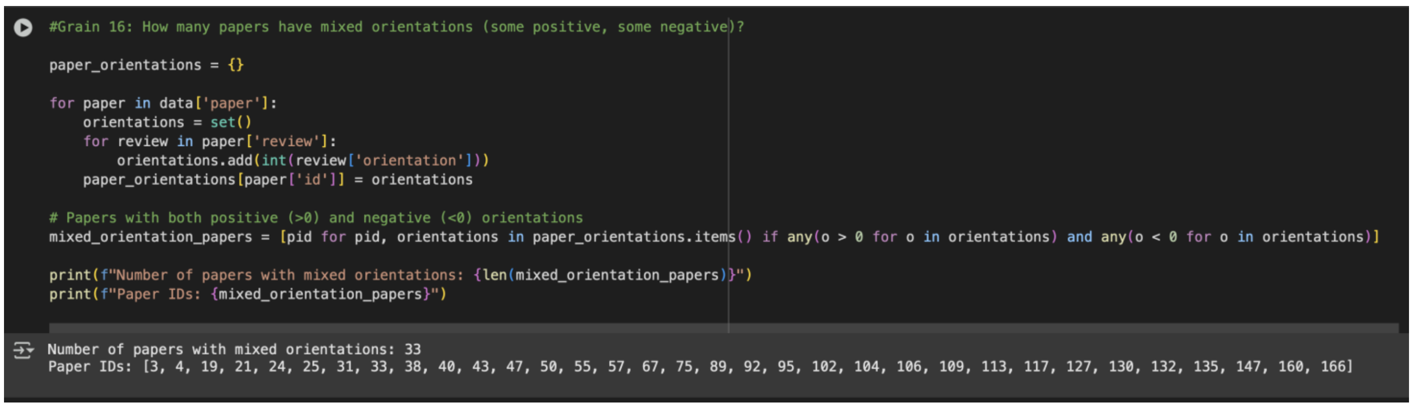
```
#Grain 14: Papers where any review had orientation = 2.

papers_with_strong_positive = []
for paper in data['paper']:
    for review in paper['review']:
        if str(review['orientation']) == "2":
            papers_with_strong_positive.append(paper['id'])
            break

print(f"Papers with at least one strong positive review (orientation=2): {papers_with_strong_positive}")
```

```
Papers with at least one strong positive review (orientation=2): [4, 12, 13, 19, 24, 27, 31, 32, 70, 73, 119, 158]
```



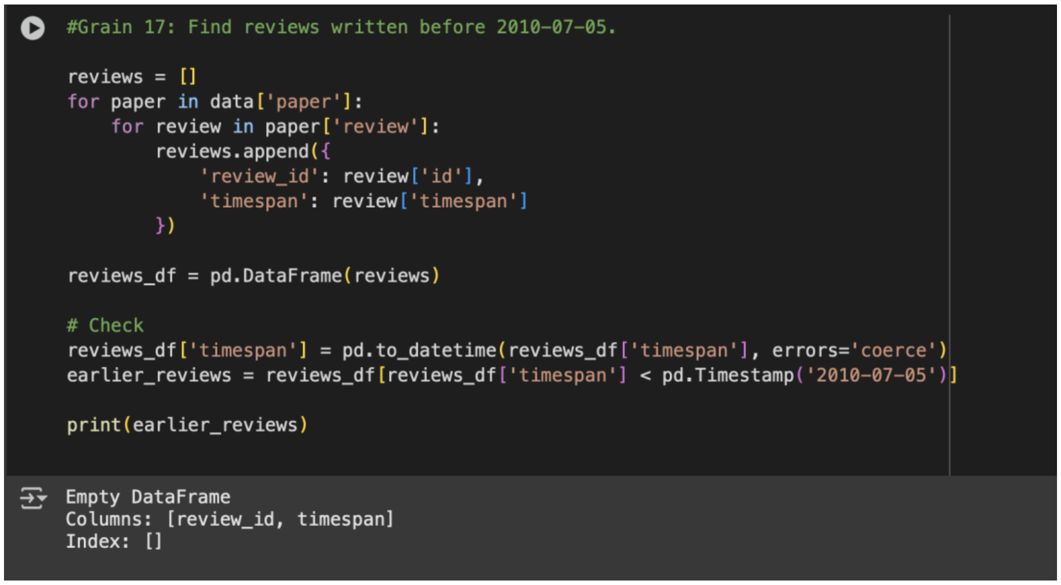
```
#Grain 15: Plot the distribution of orientations.

import matplotlib.pyplot as plt

reviews = []
for paper in data['paper']:
    for review in paper['review']:
        reviews.append({
            'orientation': review['orientation']
        })

reviews_df = pd.DataFrame(reviews)

orientation_counts = reviews_df['orientation'].value_counts().sort_index()

# Plot
plt.figure(figsize=(7,5))
orientation_counts.plot(kind='bar', color='lightblue', edgecolor='black')
plt.title('Distribution of Review Orientations')
plt.xlabel('Orientation')
plt.ylabel('Number of Reviews')
plt.grid(axis='y', linestyle='--', alpha=0.6)
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```





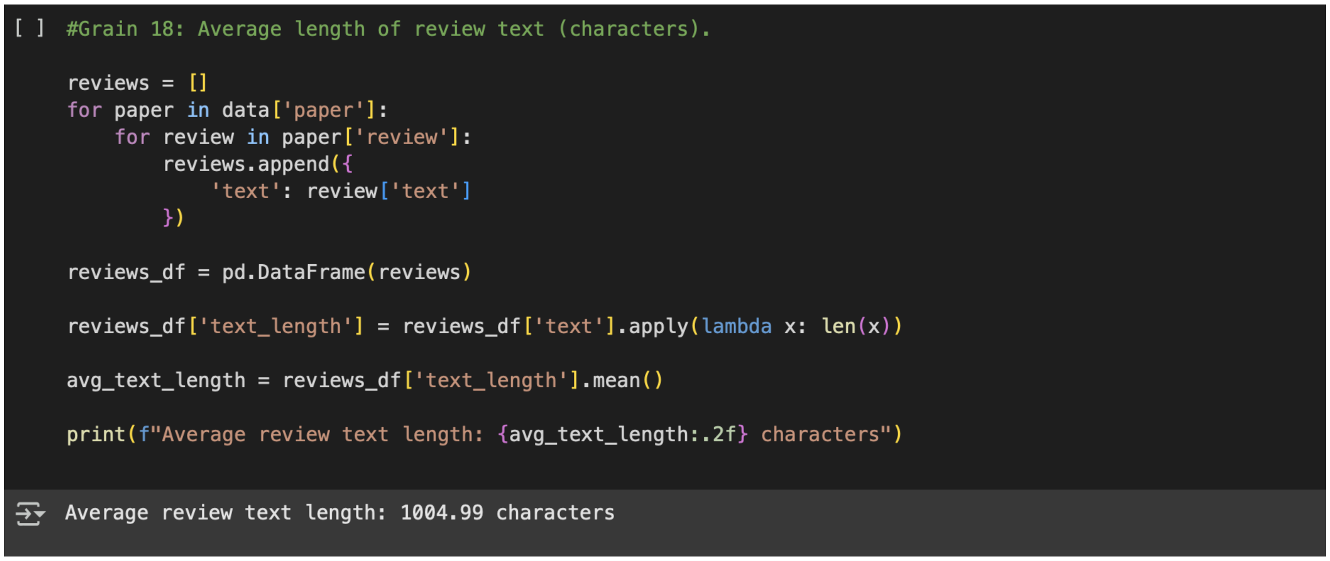
```
#Grain 16: How many papers have mixed orientations (some positive, some negative)?

paper_orientations = {}

for paper in data['paper']:
    orientations = set()
    for review in paper['review']:
        orientations.add(int(review['orientation']))
    paper_orientations[paper['id']] = orientations

# Papers with both positive (>0) and negative (<0) orientations
mixed_orientation_papers = [pid for pid, orientations in paper_orientations.items() if any(o > 0 for o in orientations) and any(o < 0 for o in orientations)]

print(f"Number of papers with mixed orientations: {len(mixed_orientation_papers)}")
print(f"Paper IDs: {mixed_orientation_papers}")
```

```
Number of papers with mixed orientations: 33
Paper IDs: [3, 4, 19, 21, 24, 25, 31, 33, 38, 40, 43, 47, 50, 55, 57, 67, 75, 89, 92, 95, 102, 104, 106, 109, 113, 117, 127, 130, 132, 135, 147, 160, 166]
```



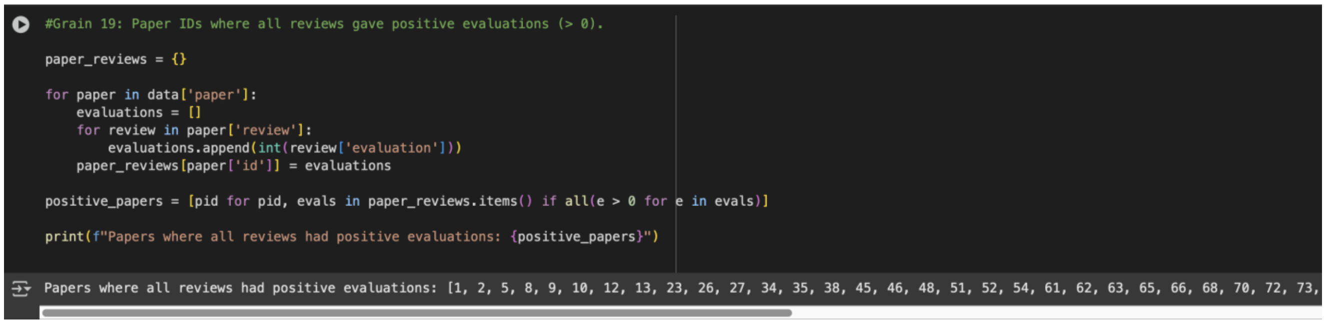
```
#Grain 17: Find reviews written before 2010-07-05.

reviews = []
for paper in data['paper']:
    for review in paper['review']:
        reviews.append({
            'review_id': review['id'],
            'timespan': review['timespan']
        })

reviews_df = pd.DataFrame(reviews)

# Check
reviews_df['timespan'] = pd.to_datetime(reviews_df['timespan'], errors='coerce')
earlier_reviews = reviews_df[reviews_df['timespan'] < pd.Timestamp('2010-07-05')]

print(earlier_reviews)
```

```
Empty DataFrame
Columns: [review_id, timespan]
Index: []
```



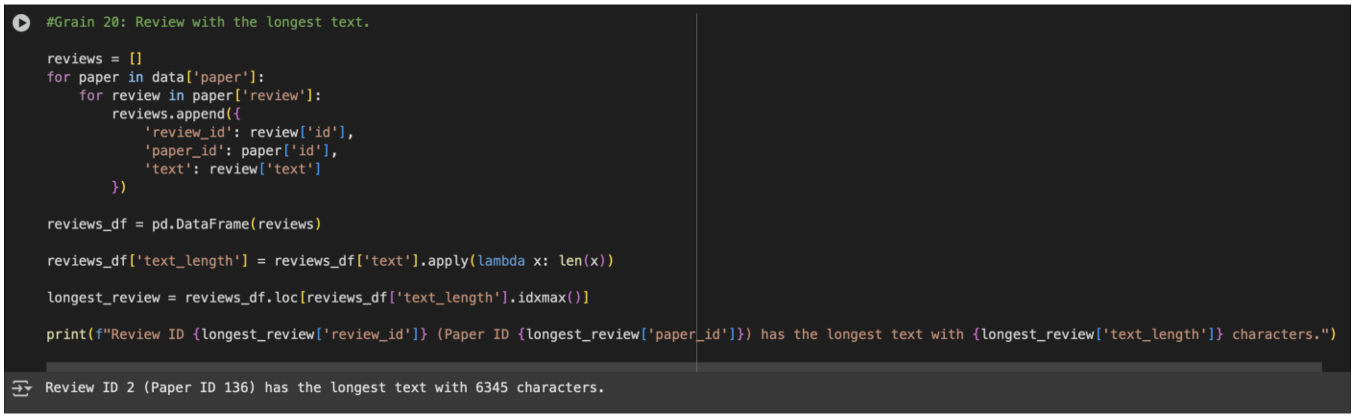
```
[ ] #Grain 18: Average length of review text (characters).

    reviews = []
    for paper in data['paper']:
        for review in paper['review']:
            reviews.append({
                'text': review['text']
            })

    reviews_df = pd.DataFrame(reviews)

    reviews_df['text_length'] = reviews_df['text'].apply(lambda x: len(x))

    avg_text_length = reviews_df['text_length'].mean()

    print(f"Average review text length: {avg_text_length:.2f} characters")
```

```
Average review text length: 1004.99 characters
```



```
#Grain 19: Paper IDs where all reviews gave positive evaluations (> 0).

paper_reviews = {}

for paper in data['paper']:
    evaluations = []
    for review in paper['review']:
        evaluations.append(int(review['evaluation']))
    paper_reviews[paper['id']] = evaluations

positive_papers = [pid for pid, evals in paper_reviews.items() if all(e > 0 for e in evals)]

print(f"Papers where all reviews had positive evaluations: {positive_papers}")
```

```
Papers where all reviews had positive evaluations: [1, 2, 5, 8, 9, 10, 12, 13, 23, 26, 27, 34, 35, 38, 45, 46, 48, 51, 52, 54, 61, 62, 63, 65, 66, 68, 70, 72, 73,
```



```
#Grain 20: Review with the longest text.

reviews = []
for paper in data['paper']:
    for review in paper['review']:
        reviews.append({
            'review_id': review['id'],
            'paper_id': paper['id'],
            'text': review['text']
        })

reviews_df = pd.DataFrame(reviews)

reviews_df['text_length'] = reviews_df['text'].apply(lambda x: len(x))

longest_review = reviews_df.loc[reviews_df['text_length'].idxmax()]

print(f"Review ID {longest_review['review_id']} (Paper ID {longest_review['paper_id']}) has the longest text with {longest_review['text_length']} characters.")
```

```
Review ID 2 (Paper ID 136) has the longest text with 6345 characters.
```