```
!pip install pandas tqdm
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-packages (2.2.2)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (4.67.1)
Requirement already satisfied: numpy>=1.26.0 in /usr/local/lib/python3.12/dist-packages (from pandas) (2.0.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas) (2025.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.2->pandas) (1.17.0
```

```
import requests
import gzip
import shutil

# Example: Electronics reviews
url = "http://snap.stanford.edu/data/amazon/productGraph/categoryFiles/reviews_Electronics_5.json.gz"
local_gz = "reviews_Electronics_5.json.gz"

# Download file
with requests.get(url, stream=True) as r:
    r.raise_for_status()
    with open(local_gz, 'wb') as f:
        for chunk in r.iter_content(chunk_size=10*1024*1024):
            if chunk:
                f.write(chunk)

print("✅ Download complete!")
```

```
✅ Download complete!
```

```
local_json = "reviews_Electronics_5.json"

with gzip.open(local_gz, 'rb') as f_in:
    with open(local_json, 'wb') as f_out:
        shutil.copyfileobj(f_in, f_out)

print("✅ Extraction complete!")
```

```
✅ Extraction complete!
```

```
import pandas as pd
from tqdm import tqdm
import json

input_file = local_json
output_file = "reviews_1M.json"

n_records = 1_000_000
count = 0
data_to_save = []

with open(input_file, 'r', encoding='utf-8') as f:
    for line in tqdm(f, total=n_records):
        data_to_save.append(json.loads(line))
        count += 1
        if count >= n_records:
            break

# Save as smaller JSON file
with open(output_file, 'w', encoding='utf-8') as f:
    for record in data_to_save:
        f.write(json.dumps(record) + "\n")

print(f"✅ Saved {n_records} reviews into {output_file}")
```

```
100%|██████████| 999999/1000000 [00:16<00:00, 58984.01it/s]
✅ Saved 1000000 reviews into reviews_1M.json
```

```
from google.colab import files
files.download("reviews_1M.json")
```

```
Downloading "reviews_1M.json":
```

```
import pandas as pd
import json

# Just to test while full file downloads
sample_file = "reviews_Electronics_5.json"

# Load only first 1000 lines
sample_data = []
with open(sample_file, 'r', encoding='utf-8') as f:
    for i, line in enumerate(f):
        if i >= 1000:
            break
        sample_data.append(json.loads(line))

# Convert to DataFrame
df = pd.DataFrame(sample_data)
print(df.head())
```

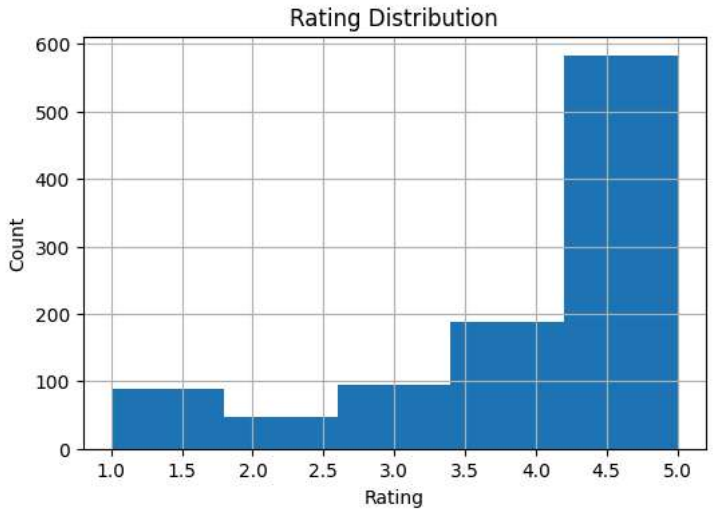
```
      reviewerID        asin              reviewerName   helpful  \
0   AO94DHGC771SJ  0528881469                   amazdnu    [0, 0]
1   AMO214LNFCEI4  0528881469           Amazon Customer  [12, 15]
2  A3N7T0DY83Y4IG  0528881469             C. A. Freeman  [43, 45]
3  A1H8PY3QHMQQA0  0528881469  Dave M. Shaw "mack dave"   [9, 10]
4  A24EV6RXELQZ63  0528881469               Wayne Smith    [0, 0]

                                        reviewText  overall  \
0  We got this GPS for my husband who is an (OTR)...      5.0
1  I'm a professional OTR truck driver, and I bou...      1.0
2  Well, what can I say.  I've had this unit in m...      3.0
3  Not going to write a long review, even thought...      2.0
4  I've had mine for a year and here's what we go...      1.0

                              summary  unixReviewTime   reviewTime
0                       Gotta have GPS!      1370131200   06 2, 2013
1                     Very Disappointed      1290643200  11 25, 2010
2                        1st impression      1283990400   09 9, 2010
3                Great grafics, POOR GPS      1290556800  11 24, 2010
4  Major issues, only excuses for support      1317254400  09 29, 2011
```

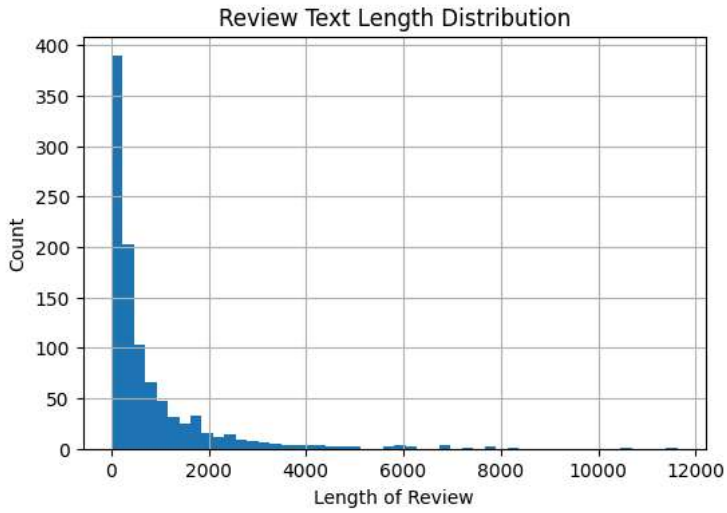
```
# Drop reviews with missing text
df = df.dropna(subset=['reviewText'])

# Lowercase
df['reviewText'] = df['reviewText'].str.lower().str.strip()

# Create unique ID if not present
if 'reviewerID' not in df.columns:
    df['reviewID'] = range(1, len(df)+1)
else:
    df['reviewID'] = df['reviewerID']

print(df[['reviewID', 'reviewText']].head())
```

```
        reviewID                                         reviewText
0   AO94DHGC771SJ  we got this gps for my husband who is an (otr)...
1   AMO214LNFCEI4  i'm a professional otr truck driver, and i bou...
2  A3N7T0DY83Y4IG  well, what can i say.  i've had this unit in m...
3  A1H8PY3QHMQQA0  not going to write a long review, even thought...
4  A24EV6RXELQZ63  i've had mine for a year and here's what we go...
```

```
import matplotlib.pyplot as plt

# Convert rating to numeric
df['overall'] = pd.to_numeric(df['overall'], errors='coerce')

# Plot rating distribution
plt.figure(figsize=(6,4))
df['overall'].hist(bins=5)
plt.title('Rating Distribution')
plt.xlabel('Rating')
```

```
plt.ylabel('Count')
plt.show()

# Plot review length distribution
df['review_len'] = df['reviewText'].apply(len)
plt.figure(figsize=(6,4))
df['review_len'].hist(bins=50)
plt.title('Review Text Length Distribution')
plt.xlabel('Length of Review')
plt.ylabel('Count')
plt.show()
```





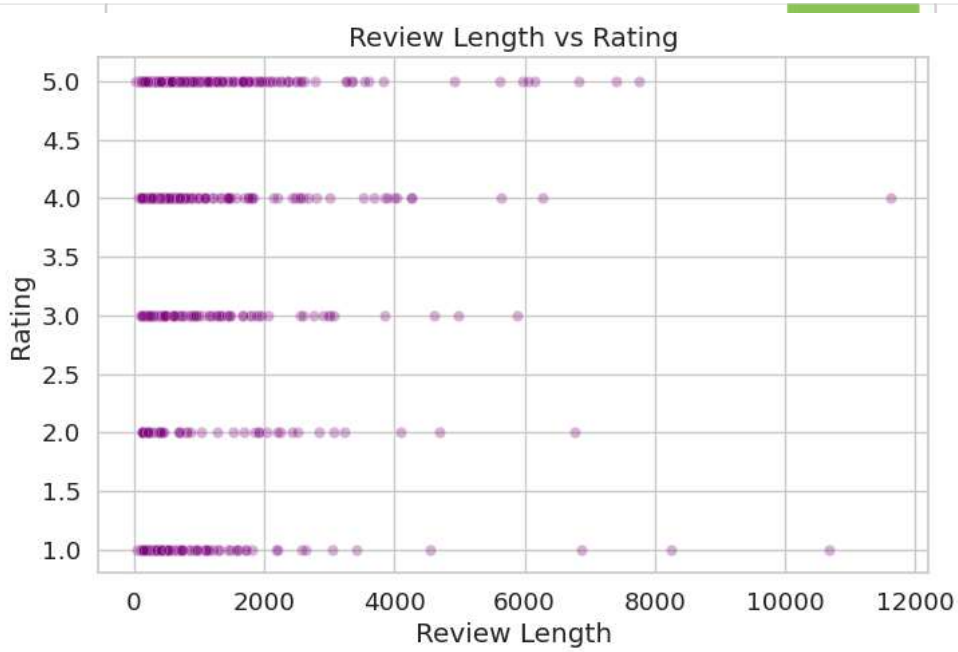

```
df = df[df['review_len'] > 10]
```

```
df = df.drop_duplicates(subset=['reviewID'])
```

```
import matplotlib.pyplot as plt
import seaborn as sns

sns.set(style="whitegrid", palette="muted", font_scale=1.2)

# 1️⃣ Rating Distribution
plt.figure(figsize=(8,5))
sns.countplot(x='overall', data=df, palette='viridis')
plt.title('Rating Distribution')
plt.xlabel('Rating')
plt.ylabel('Count')
plt.show()

# 2️⃣ Review Length Distribution
plt.figure(figsize=(8,5))
sns.histplot(df['review_len'], bins=50, kde=True, color='skyblue')
```

```
plt.title('Review Text Length Distribution')
plt.xlabel('Length of Review')
plt.ylabel('Count')
plt.show()

# 3️⃣ Heatmap: Correlation between numeric features
numeric_cols = ['overall', 'review_len']
plt.figure(figsize=(6,4))
sns.heatmap(df[numeric_cols].corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()

# 4️⃣ Top 20 Most Frequent Words (after cleaning)
from collections import Counter
import re

all_text = ' '.join(df['reviewText'].tolist())
words = re.findall(r'\b\w+\b', all_text)
counter = Counter(words)
top_words = counter.most_common(20)

words_df = pd.DataFrame(top_words, columns=['Word','Count'])

plt.figure(figsize=(10,5))
sns.barplot(x='Count', y='Word', data=words_df, palette='magma')
plt.title('Top 20 Most Frequent Words in Reviews')
plt.show()
```

```
plt.figure(figsize=(8,5))
sns.scatterplot(x='review_len', y='overall', data=df, alpha=0.3, color='purple')
plt.title('Review Length vs Rating')
plt.xlabel('Review Length')
plt.ylabel('Rating')
plt.show()
```




```
df[['reviewID','reviewText']].to_json("reviews_1M_clean.json", orient='records', lines=True)
print("✅ Cleaned JSON saved for Merkle Tree")
```

```
✅ Cleaned JSON saved for Merkle Tree
```

```
from google.colab import drive
import shutil
import os

# Mount Google Drive
drive.mount('/content/drive')

# Destination folder in your Drive
dest_dir = "/content/drive/MyDrive/AmazonDataset"
os.makedirs(dest_dir, exist_ok=True)

# Move the cleaned file there
shutil.move("reviews_1M_clean.json", f"{dest_dir}/reviews_1M_clean.json")
print(f"✅ File saved to {dest_dir}")
```

```
Mounted at /content/drive
✅ File saved to /content/drive/MyDrive/AmazonDataset
```

Start coding or generate with AI.