In [2]:
```
import dask.dataframe as dd
import pandas as pd
import matplotlib.pyplot as plt
```

In [4]:
```
df = dd.read_csv("kindle_reviews.csv", dtype={'reviewerName': 'object'})
```

In [6]:
```
df.head()
```

Out[6]:

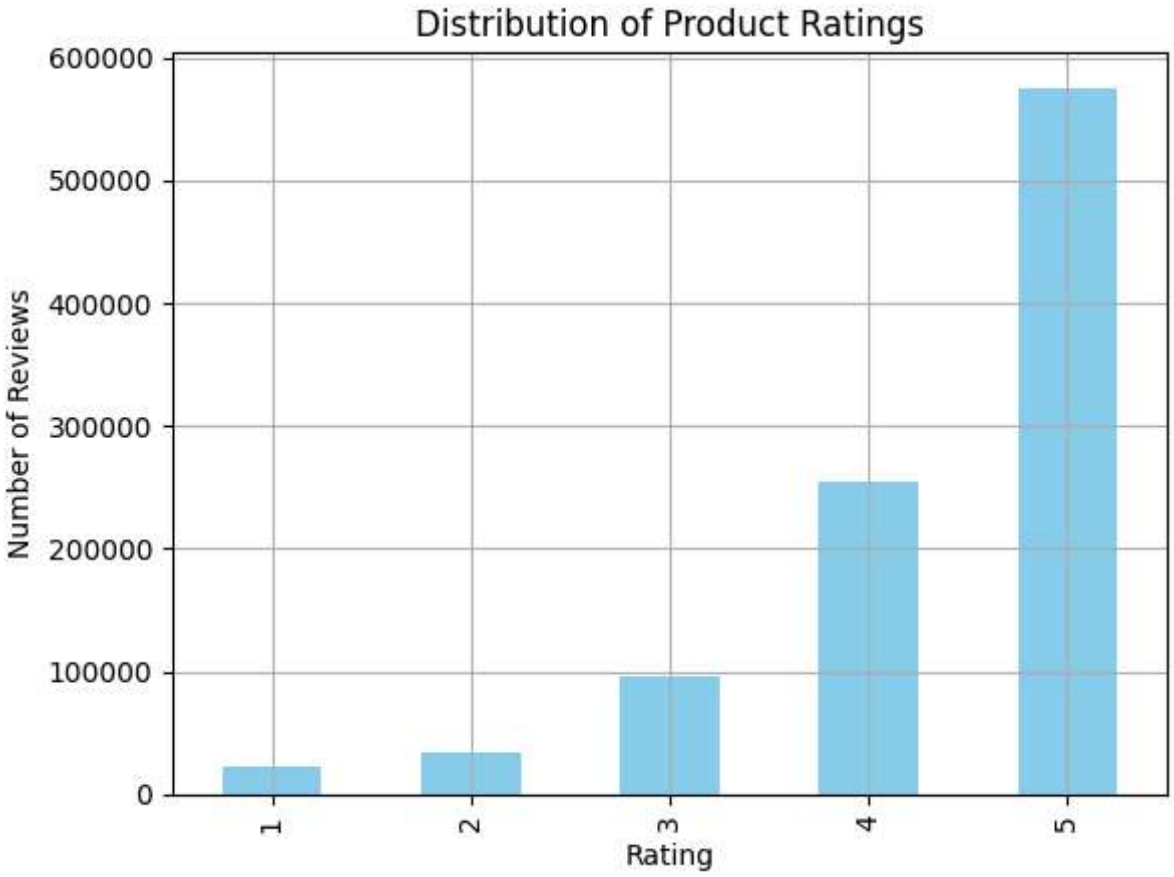
	Unnamed: 0	asin	helpful	overall	reviewText	reviewTime	reviewerID	re
0	0	B000F83SZQ	[0, 0]	5	I enjoy vintage books and movies so I enjoyed ...	05 5, 2014	A1F6404F1VG29J	
1	1	B000F83SZQ	[2, 2]	4	This book is a reissue of an old one; the auth...	01 6, 2014	AN0N05A9LIJEQ	
2	2	B000F83SZQ	[2, 2]	4	This was a fairly interesting read. It had ol...	04 4, 2014	A795DMNCJILA6	
3	3	B000F83SZQ	[1, 1]	5	I'd never read any of the Amy Brewster mysteri...	02 19, 2014	A1FV0SX13TWVXQ	El
4	4	B000F83SZQ	[0, 1]	4	If you like period pieces - clothing, lingo, y...	03 19, 2014	A3SPTOKDG7WBLN	Fa

In [8]:
```
# Total reviews
total_reviews = df.shape[0].compute()
print(f"Total reviews: {total_reviews}")

# Unique products and reviewers
unique_products = df['asin'].nunique().compute()
unique_reviewers = df['reviewerID'].nunique().compute()
print(f"Unique products: {unique_products}")
print(f"Unique reviewers: {unique_reviewers}")
```

```
Total reviews: 982619
Unique products: 61934
Unique reviewers: 68223
```

In [9]:
```python
# Distribution of ratings
rating_counts = df['overall'].value_counts().compute().sort_index()
print("Rating counts:")
print(rating_counts)

# Plot rating distribution
rating_counts.plot(kind='bar', color='skyblue')
plt.title("Distribution of Product Ratings")
plt.xlabel("Rating")
plt.ylabel("Number of Reviews")
plt.grid(True)
plt.show()
```

```
Rating counts:
overall
1     23018
2     34130
3     96194
4    254013
5    575264
Name: count, dtype: int64
```



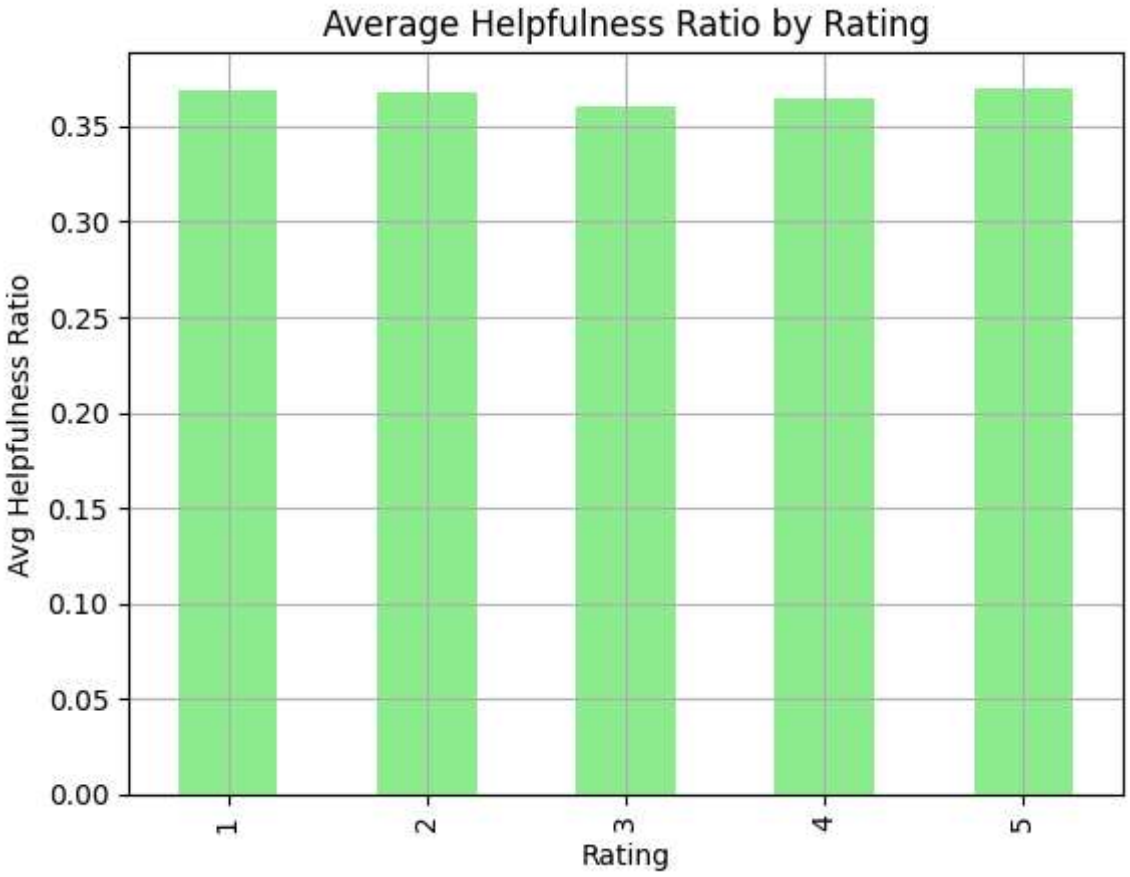
In [16]:
```python
import ast

# Function to safely parse the helpful column
def parse_helpful_list(x):
```

```
    try:
        values = ast.literal_eval(x)
        if isinstance(values, list) and len(values) == 2:
            return values[0], values[1]
    except:
        return 0, 0
    return 0, 0

# Map using Dask
helpful_parsed = df['helpful'].map(parse_helpful_list, meta=('helpful', 'object')).
helpful_df = pd.DataFrame(helpful_parsed.tolist(), columns=['helpful_num', 'helpful

# Reassign to original df
df = df.assign(helpful_num=helpful_df['helpful_num'], helpful_den=helpful_df['helpf
```

In [17]:
```
# Add helpfulness ratio safely
df['helpfulness_ratio'] = df.apply(
    lambda row: row.helpful_num / row.helpful_den if row.helpful_den > 0 else 0,
    axis=1,
    meta=('helpfulness_ratio', 'float')
)
```

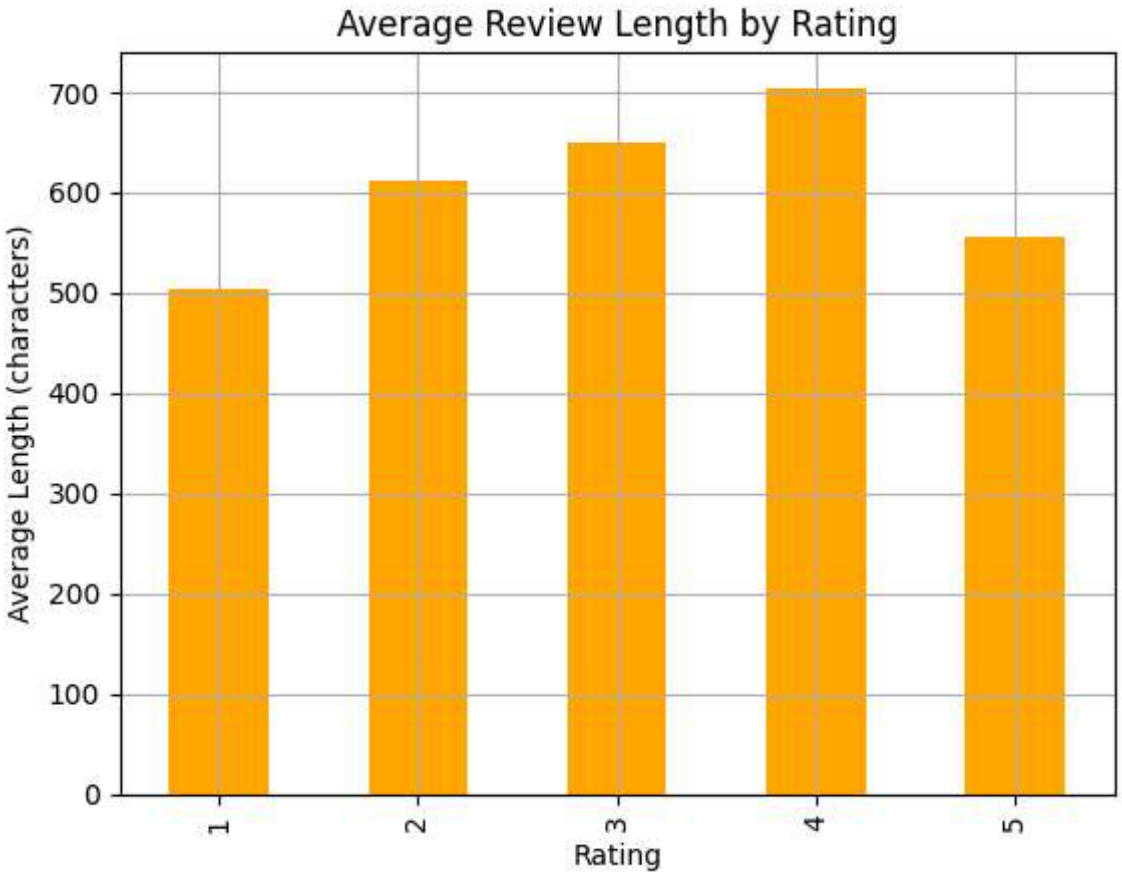
In [20]:
```
# Mean helpfulness ratio by rating
helpfulness_by_rating = df.groupby('overall')['helpfulness_ratio'].mean().compute()
print(helpfulness_by_rating)

# Plot
helpfulness_by_rating.plot(kind='bar', color='lightgreen')
plt.title("Average Helpfulness Ratio by Rating")
plt.xlabel("Rating")
plt.ylabel("Avg Helpfulness Ratio")
plt.grid(True)
plt.show()
```

```
overall
1    0.369004
2    0.367690
3    0.360499
4    0.364891
5    0.369964
Name: helpfulness_ratio, dtype: float64
```



In [21]:
```
# Compute review text length
df['review_length'] = df['reviewText'].str.len()

# Average review length by rating
avg_length_by_rating = df.groupby('overall')['review_length'].mean().compute()
print("Average review length by rating:")
print(avg_length_by_rating)

# Plot
avg_length_by_rating.plot(kind='bar', color='orange')
plt.title("Average Review Length by Rating")
plt.xlabel("Rating")
plt.ylabel("Average Length (characters)")
plt.grid(True)
plt.show()
```

```
Average review length by rating:
overall
1    504.446564
2    611.172986
3    650.021436
4    704.150364
5    555.573188
Name: review_length, dtype: Float64
```



In [22]:
```
# Top 10 most helpful reviews with at least 10 votes
top_helpful_reviews = df[df['helpful_den'] >= 10].nlargest(10, 'helpfulness_ratio')
print("Top 10 helpful reviews:")
top_helpful_reviews
```

```
Top 10 helpful reviews:
```

Out[22]:

	asin	reviewerID	overall	helpfulness_ratio
158	B00JKZZ8K8	AX0ZTNX6KVJ40	5	1.0
1274	B00JLZ5J7E	A354MOJ56YP5Y8	5	1.0
1616	B00JM2HDOS	A2I58O8E2AVBZM	5	1.0
1910	B00JMF2RPU	A2DRRMQPPWXB62	5	1.0
2070	B00JMG2RU4	A2UKWLNK4R52YN	5	1.0
4358	B00JO1PSX0	A1XIQBGKVH6G8B	5	1.0
6407	B00JPJY43Q	A1YCJXNHARGRVR	1	1.0
7025	B00JPW10QC	AR35CG3TE4S4D	3	1.0
7688	B00JQQD394	A34HRQ4GESJ158	5	1.0
8473	B00JRC0BCY	AAXG649HM1S5Q	5	1.0

In [24]:
```
# Group and aggregate
product_stats = df.groupby('asin').agg({'overall': ['mean', 'count']}).compute()
```

```
product_stats.columns = ['avg_rating', 'review_count']

# Filter and sort
top_products = product_stats[product_stats['review_count'] >= 50].sort_values(by='a
print("Top 10 best-rated products (min 50 reviews):")
top_products
```

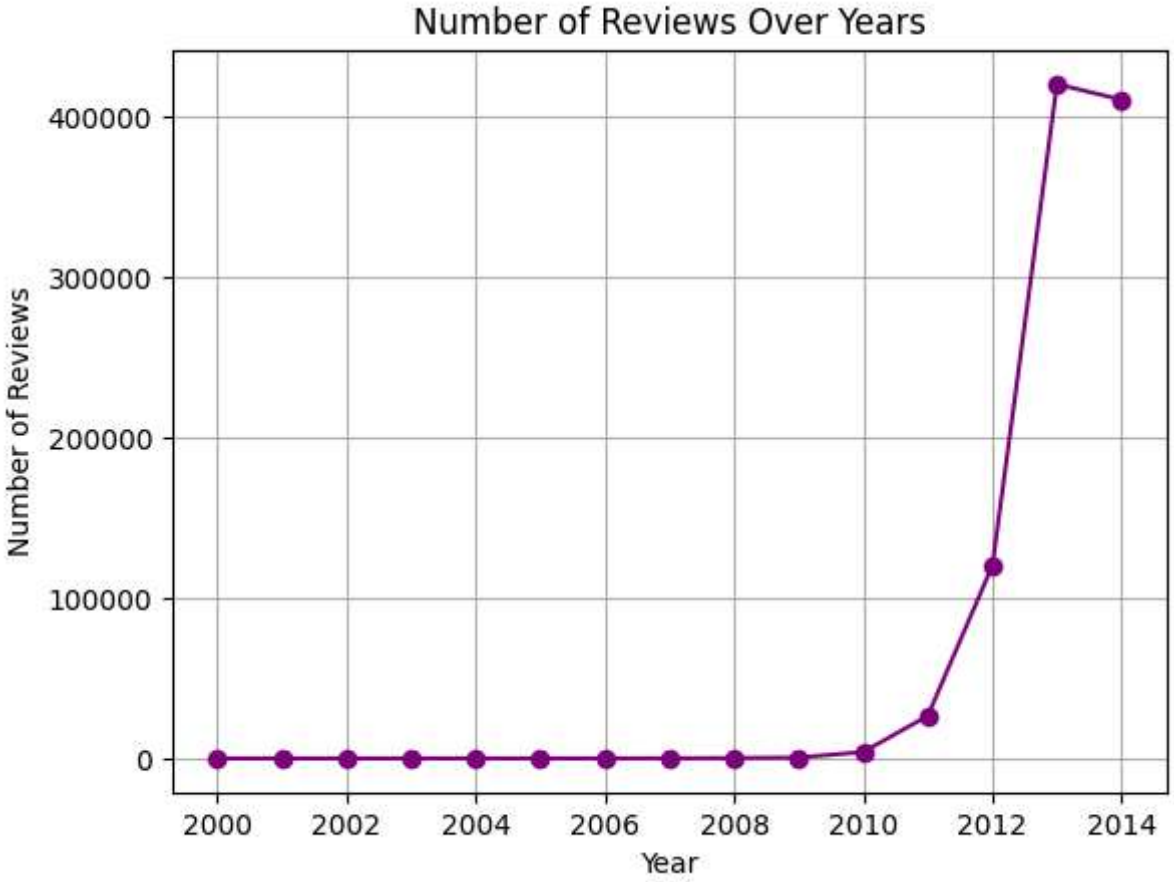
```
Top 10 best-rated products (min 50 reviews):
```

Out[24]:

asin	avg_rating	review_count
B00ID7K5CA	5.000000	70
B00JMORESG	5.000000	59
B00GN8K6CU	4.975309	81
B00DRN5X72	4.969388	98
B0073999SU	4.968254	63
B00LDI41Z8	4.964912	57
B00KP5P2EY	4.963636	55
B00IJBDBG2	4.961039	77
B00DRERCRA	4.957143	70
B00C44PW5I	4.951456	103

In [28]:

```
# Convert Unix time to datetime
df['review_datetime'] = dd.to_datetime(df['unixReviewTime'], unit='s')
df['review_year'] = df['review_datetime'].dt.year

# Reviews per year
reviews_per_year = df.groupby('review_year').size().compute().sort_index()

# Plot
reviews_per_year.plot(kind='line', marker='o', color='purple')
plt.title("Number of Reviews Over Years")
plt.xlabel("Year")
plt.ylabel("Number of Reviews")
plt.grid(True)
plt.show()
```


Number of Reviews Over Years
Number of Reviews
400000
300000
200000
100000
0
2000
2002
2004
2006
2008
2010
2012
2014
Year