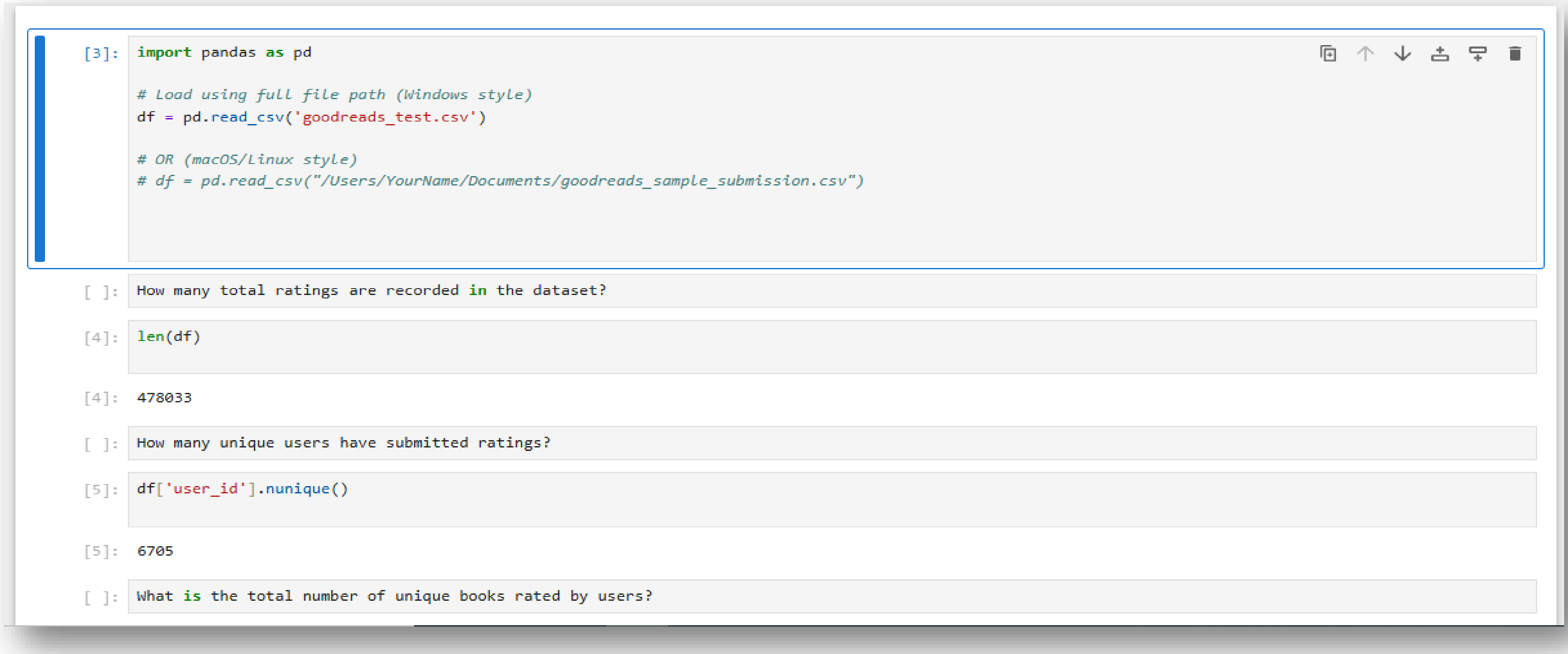
NAME : Parth Somnath Ghule

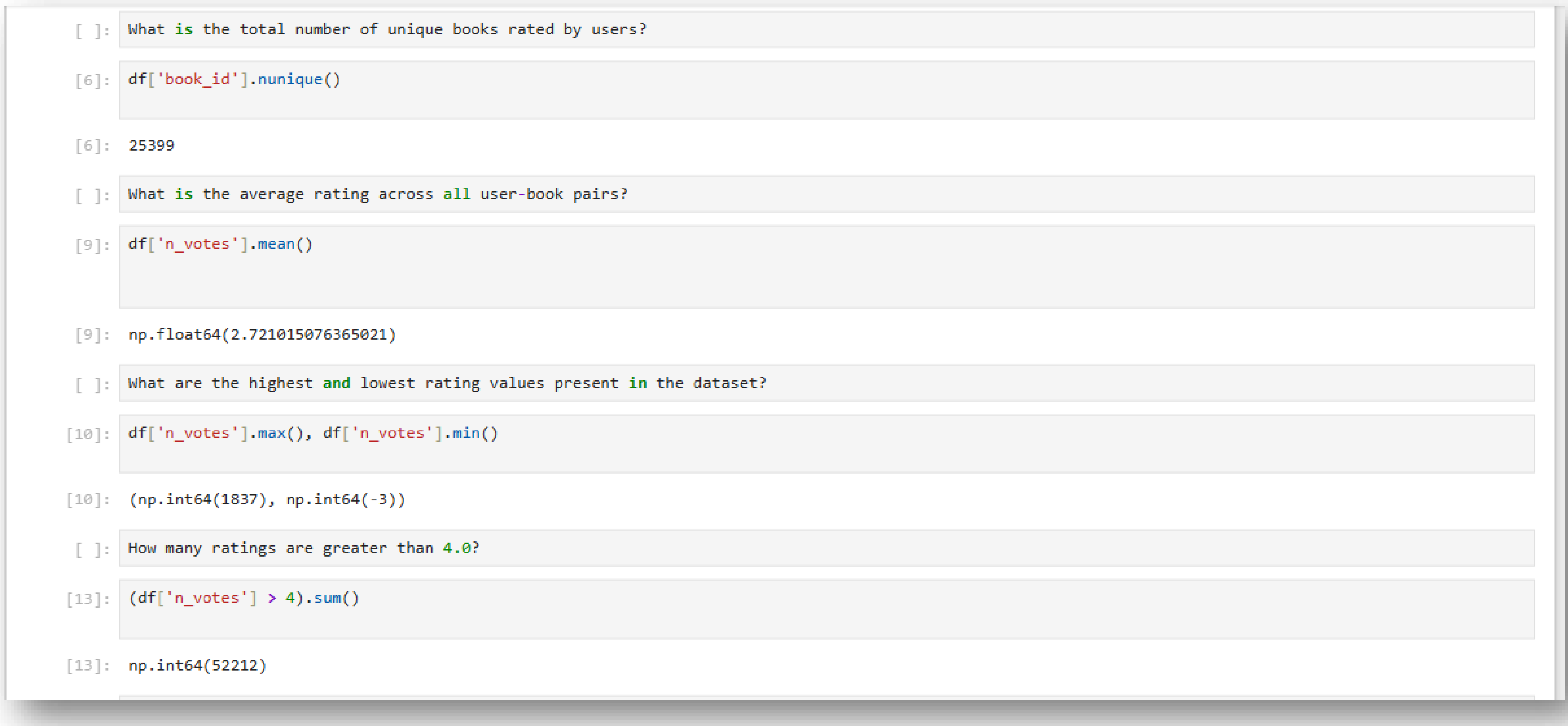
Prn : 202401040276

Division : CS-2   Roll no. :71

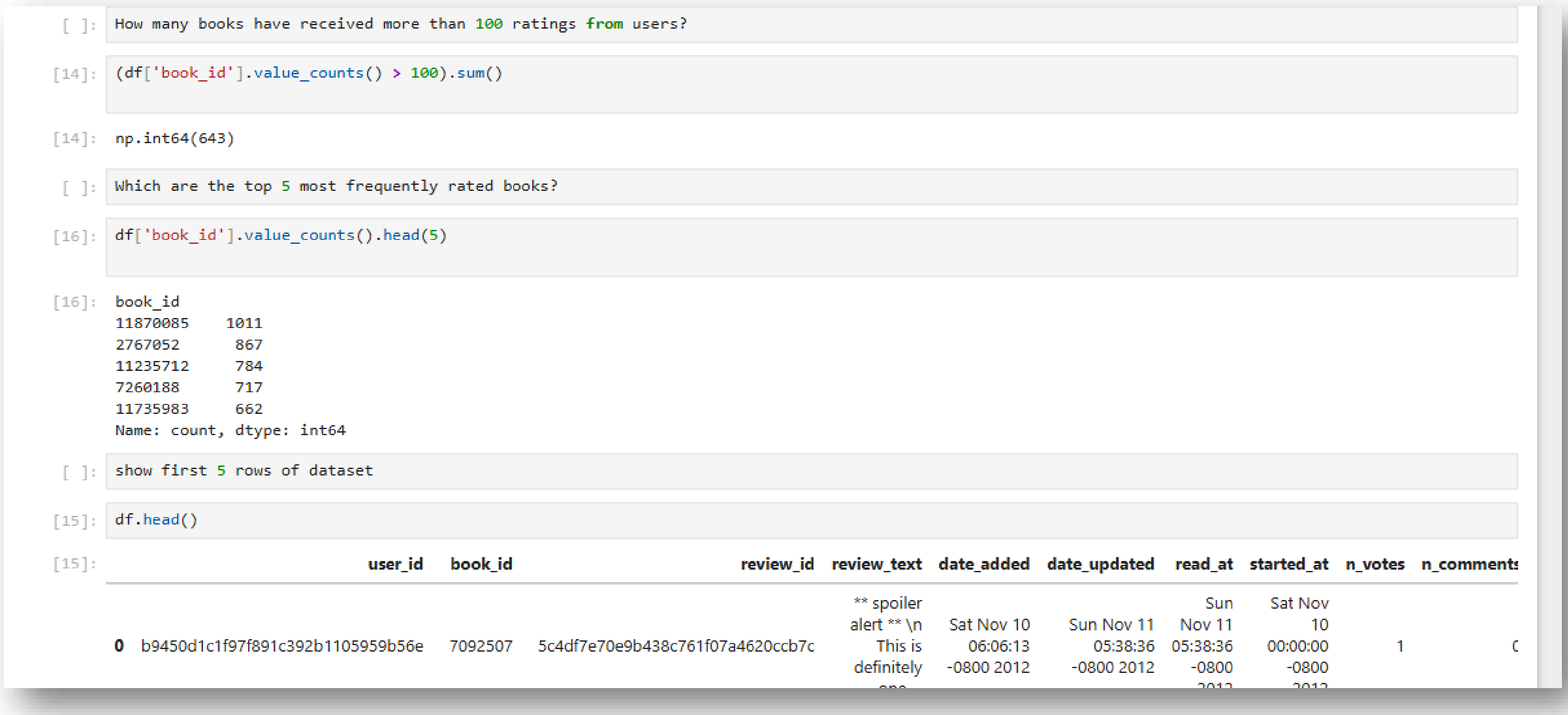
EDS activity 1



```
[3]: import pandas as pd

     # Load using full file path (Windows style)
     df = pd.read_csv('goodreads_test.csv')

     # OR (macOS/Linux style)
     # df = pd.read_csv("/Users/YourName/Documents/goodreads_sample_submission.csv")

[ ]: How many total ratings are recorded in the dataset?

[4]: len(df)

[4]: 478033

[ ]: How many unique users have submitted ratings?

[5]: df['user_id'].nunique()

[5]: 6705

[ ]: What is the total number of unique books rated by users?
```



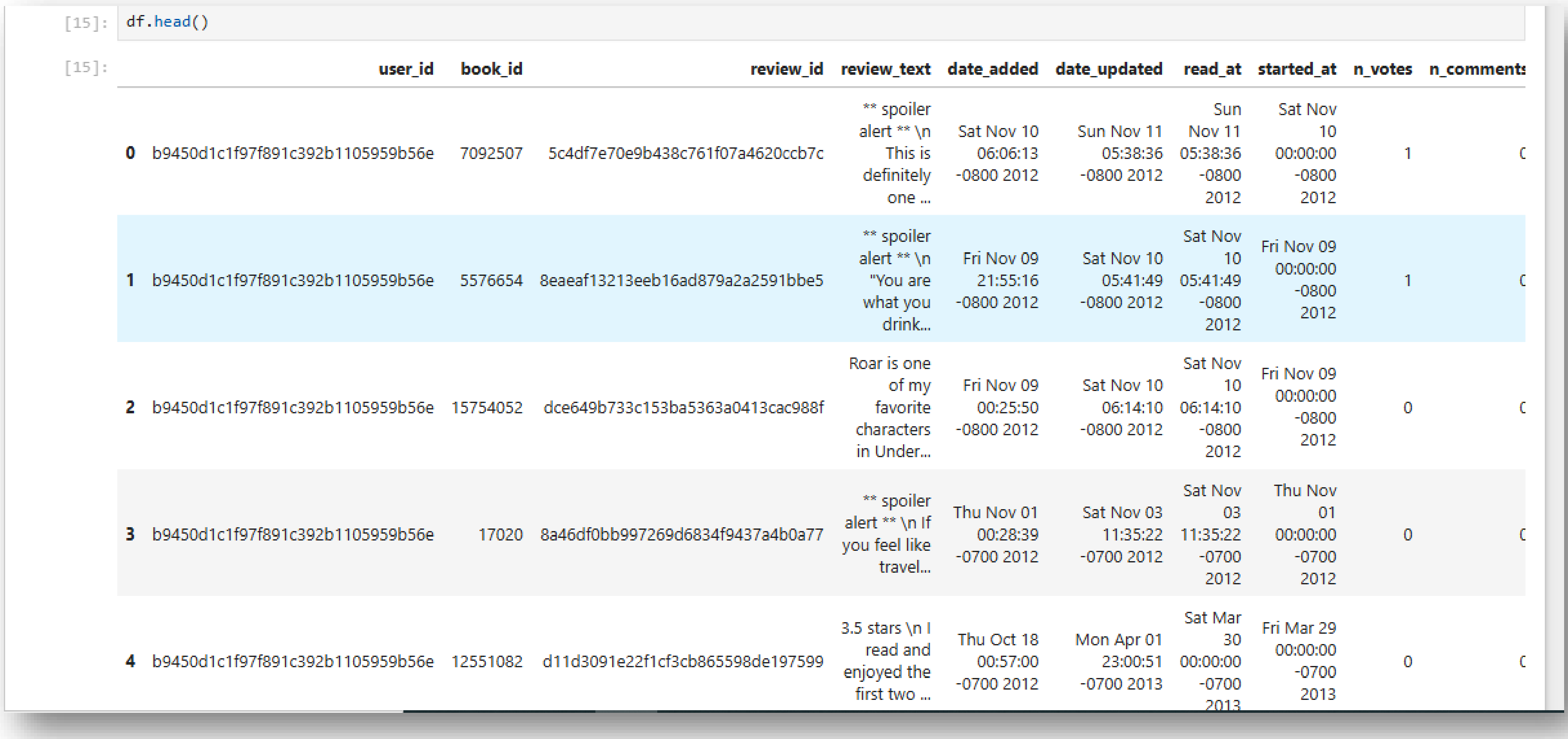
```
[ ]: What is the total number of unique books rated by users?

[6]: df['book_id'].nunique()

[6]: 25399

[ ]: What is the average rating across all user-book pairs?

[9]: df['n_votes'].mean()

[9]: np.float64(2.721015076365021)

[ ]: What are the highest and lowest rating values present in the dataset?

[10]: df['n_votes'].max(), df['n_votes'].min()

[10]: (np.int64(1837), np.int64(-3))

[ ]: How many ratings are greater than 4.0?

[13]: (df['n_votes'] > 4).sum()

[13]: np.int64(52212)
```



```
[ ]: How many books have received more than 100 ratings from users?

[14]: (df['book_id'].value_counts() > 100).sum()

[14]: np.int64(643)

[ ]: Which are the top 5 most frequently rated books?

[16]: df['book_id'].value_counts().head(5)

[16]: book_id
      11870085    1011
      2767052      867
      11235712     784
      7260188      717
      11735983     662
      Name: count, dtype: int64

[ ]: show first 5 rows of dataset

[15]: df.head()
```

[15]:

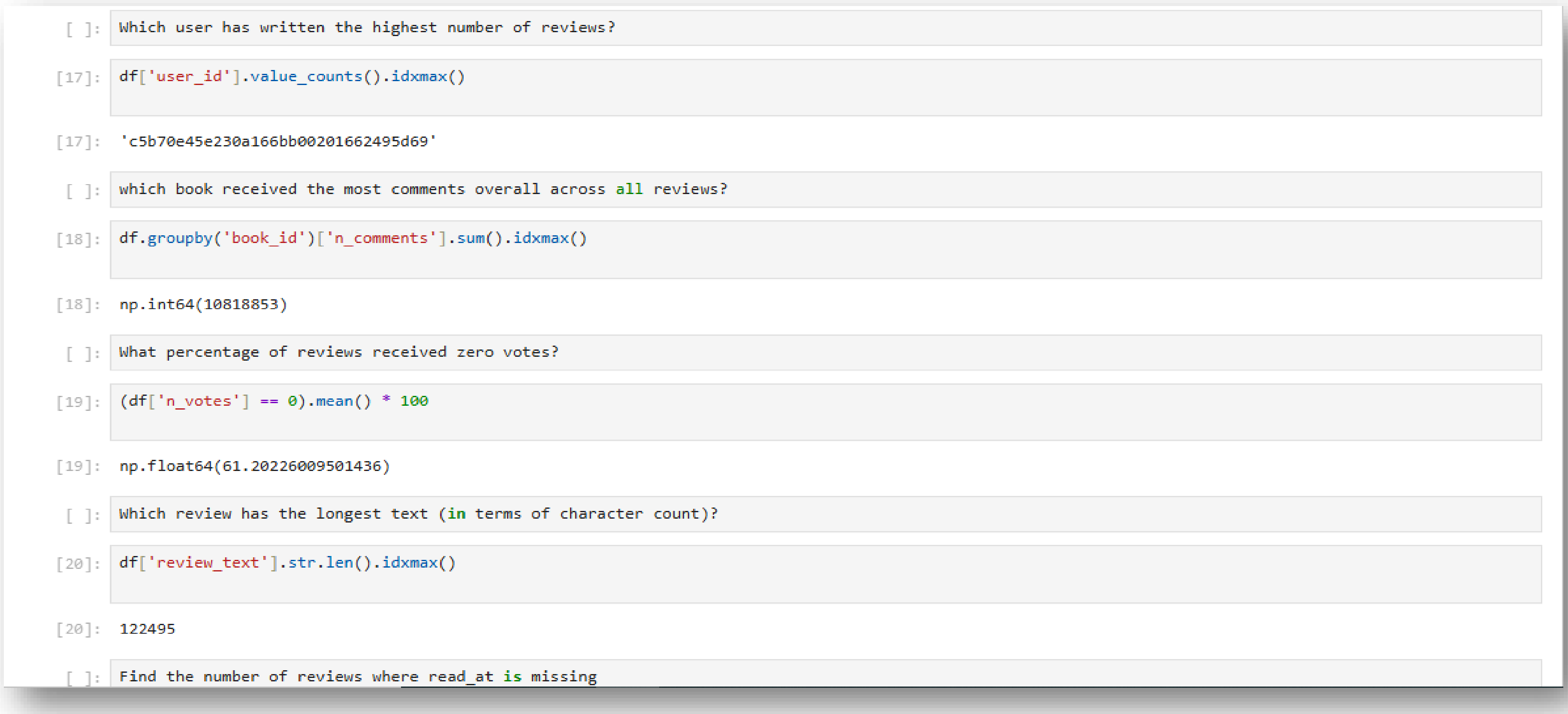
	user_id	book_id	review_id	review_text	date_added	date_updated	read_at	started_at	n_votes	n_comments
0	b9450d1c1f97f891c392b1105959b56e	7092507	5c4df7e70e9b438c761f07a4620ccb7c	** spoiler alert ** \n This is definitely	Sat Nov 10 06:06:13 -0800 2012	Sun Nov 11 05:38:36 -0800 2012	Sun Nov 11 05:38:36 -0800	Sat Nov 10 00:00:00 -0800	1	[illegible]



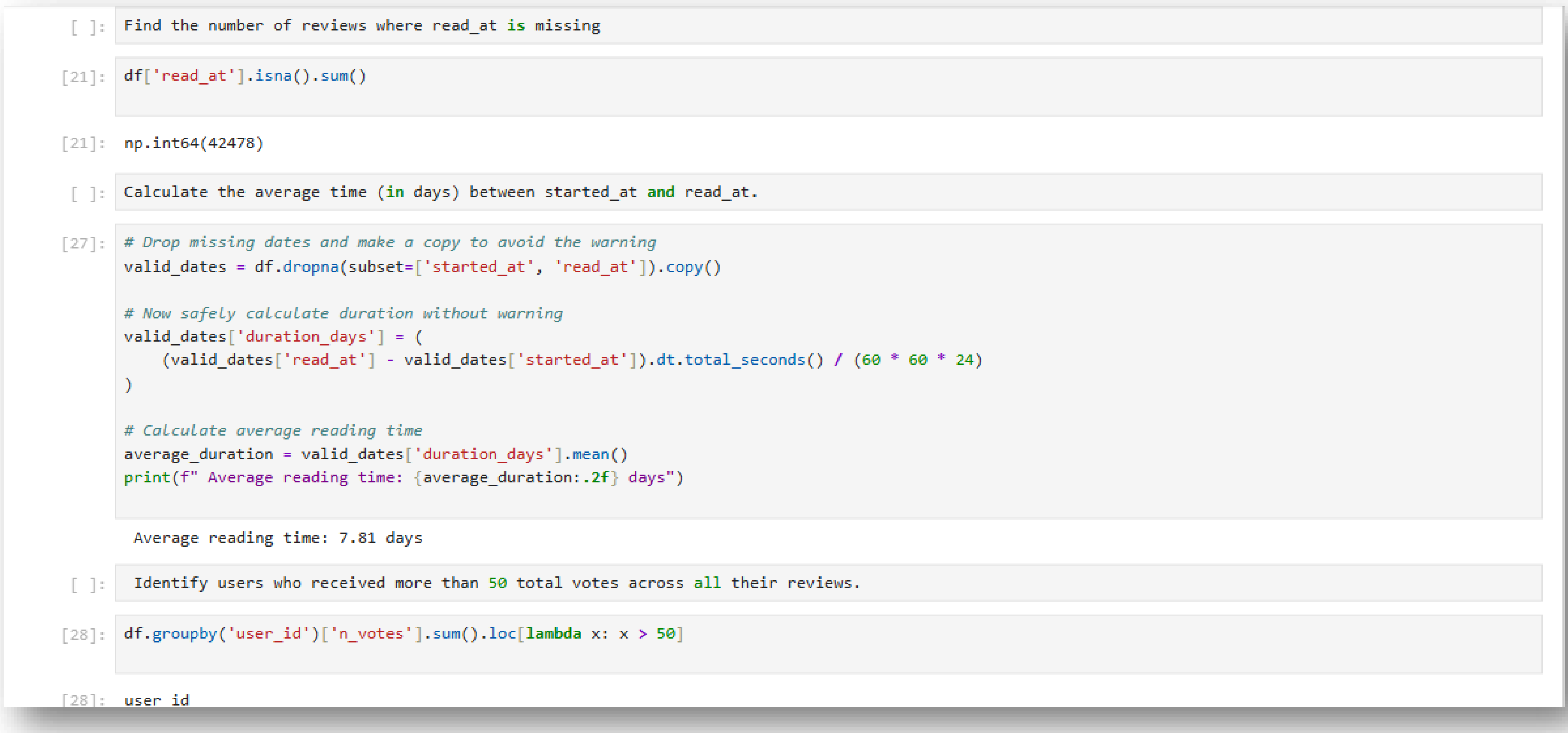
```
[15]: df.head()
```

[15]:

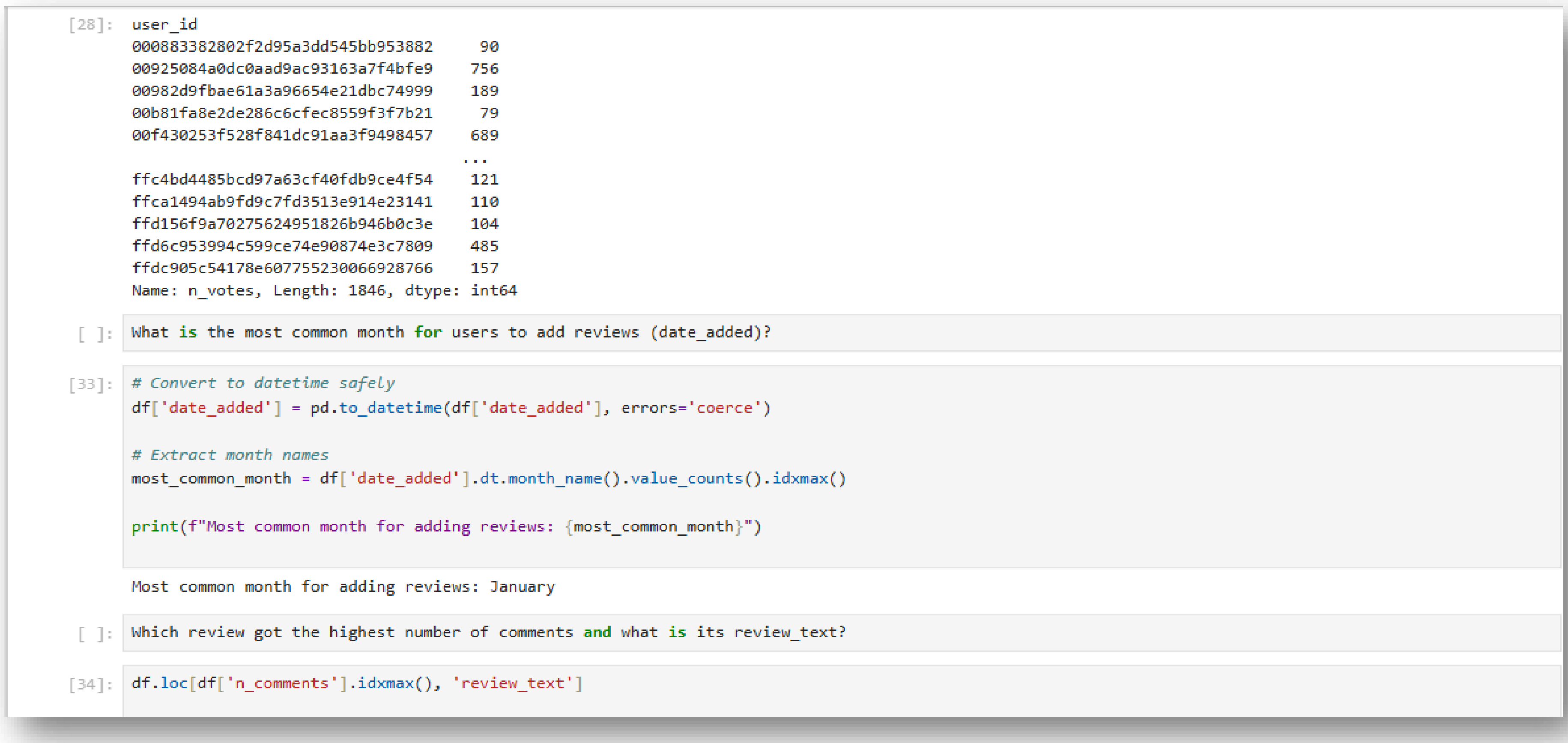
	user_id	book_id	review_id	review_text	date_added	date_updated	read_at	started_at	n_votes	n_comments
0	b9450d1c1f97f891c392b1105959b56e	7092507	5c4df7e70e9b438c761f07a4620ccb7c	** spoiler alert ** \n This is definitely one ...	Sat Nov 10 06:06:13 -0800 2012	Sun Nov 11 05:38:36 -0800 2012	Sun Nov 11 05:38:36 -0800 2012	Sat Nov 10 00:00:00 -0800 2012	1	[illegible]
1	b9450d1c1f97f891c392b1105959b56e	5576654	8eaeaf13213eeb16ad879a2a2591bbe5	** spoiler alert ** \n "You are what you drink...	Fri Nov 09 21:55:16 -0800 2012	Sat Nov 10 05:41:49 -0800 2012	Sat Nov 10 05:41:49 -0800 2012	Fri Nov 09 00:00:00 -0800 2012	1	[illegible]
2	b9450d1c1f97f891c392b1105959b56e	15754052	dce649b733c153ba5363a0413cac988f	Roar is one of my favorite characters in Under...	Fri Nov 09 00:25:50 -0800 2012	Sat Nov 10 06:14:10 -0800 2012	Sat Nov 10 06:14:10 -0800 2012	Fri Nov 09 00:00:00 -0800 2012	0	[illegible]
3	b9450d1c1f97f891c392b1105959b56e	17020	8a46df0bb997269d6834f9437a4b0a77	** spoiler alert ** \n If you feel like travel...	Thu Nov 01 00:28:39 -0700 2012	Sat Nov 03 11:35:22 -0700 2012	Sat Nov 03 11:35:22 -0700 2012	Thu Nov 01 00:00:00 -0700 2012	0	[illegible]
4	b9450d1c1f97f891c392b1105959b56e	12551082	d11d3091e22f1cf3cb865598de197599	3.5 stars \n I read and enjoyed the first two ...	Thu Oct 18 00:57:00 -0700 2012	Mon Apr 01 23:00:51 -0700 2013	Sat Mar 30 00:00:00 -0700 2013	Fri Mar 29 00:00:00 -0700 2013	0	[illegible]



```
[ ]:  Which user has written the highest number of reviews?

[17]: df['user_id'].value_counts().idxmax()

[17]: 'c5b70e45e230a166bb00201662495d69'

[ ]:  which book received the most comments overall across all reviews?

[18]: df.groupby('book_id')['n_comments'].sum().idxmax()

[18]: np.int64(10818853)

[ ]:  What percentage of reviews received zero votes?

[19]: (df['n_votes'] == 0).mean() * 100

[19]: np.float64(61.20226009501436)

[ ]:  Which review has the longest text (in terms of character count)?

[20]: df['review_text'].str.len().idxmax()

[20]: 122495

[ ]:  Find the number of reviews where read_at is missing
```



```
[ ]:  Find the number of reviews where read_at is missing

[21]: df['read_at'].isna().sum()

[21]: np.int64(42478)

[ ]:  Calculate the average time (in days) between started_at and read_at.

[27]: # Drop missing dates and make a copy to avoid the warning
      valid_dates = df.dropna(subset=['started_at', 'read_at']).copy()

      # Now safely calculate duration without warning
      valid_dates['duration_days'] = (
          (valid_dates['read_at'] - valid_dates['started_at']).dt.total_seconds() / (60 * 60 * 24)
      )

      # Calculate average reading time
      average_duration = valid_dates['duration_days'].mean()
      print(f" Average reading time: {average_duration:.2f} days")

       Average reading time: 7.81 days

[ ]:   Identify users who received more than 50 total votes across all their reviews.

[28]: df.groupby('user_id')['n_votes'].sum().loc[lambda x: x > 50]

[28]: user_id
```



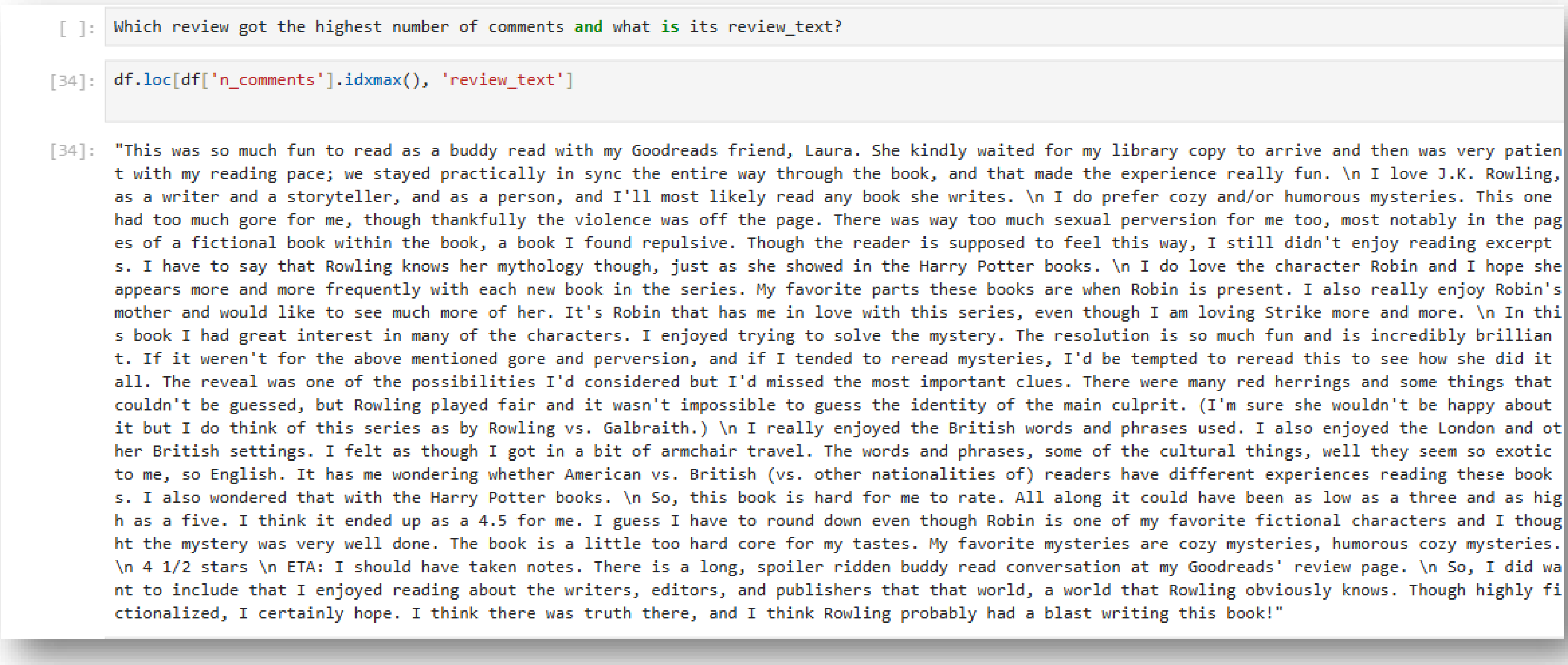
```
[28]: user_id
      000883382802f2d95a3dd545bb953882     90
      00925084a0dc0aad9ac93163a7f4bfe9    756
      00982d9fbae61a3a96654e21dbc74999    189
      00b81fa8e2de286c6cfec8559f3f7b21     79
      00f430253f528f841dc91aa3f9498457    689
                                         ...
      ffc4bd4485bcd97a63cf40fdb9ce4f54    121
      ffca1494ab9fd9c7fd3513e914e23141    110
      ffd156f9a70275624951826b946b0c3e    104
      ffd6c953994c599ce74e90874e3c7809    485
      ffdc905c54178e607755230066928766    157
      Name: n_votes, Length: 1846, dtype: int64
```

```
[ ]: What is the most common month for users to add reviews (date_added)?
```

```
[33]: # Convert to datetime safely
      df['date_added'] = pd.to_datetime(df['date_added'], errors='coerce')

      # Extract month names
      most_common_month = df['date_added'].dt.month_name().value_counts().idxmax()

      print(f"Most common month for adding reviews: {most_common_month}")

      Most common month for adding reviews: January
```

```
[ ]: Which review got the highest number of comments and what is its review_text?
```

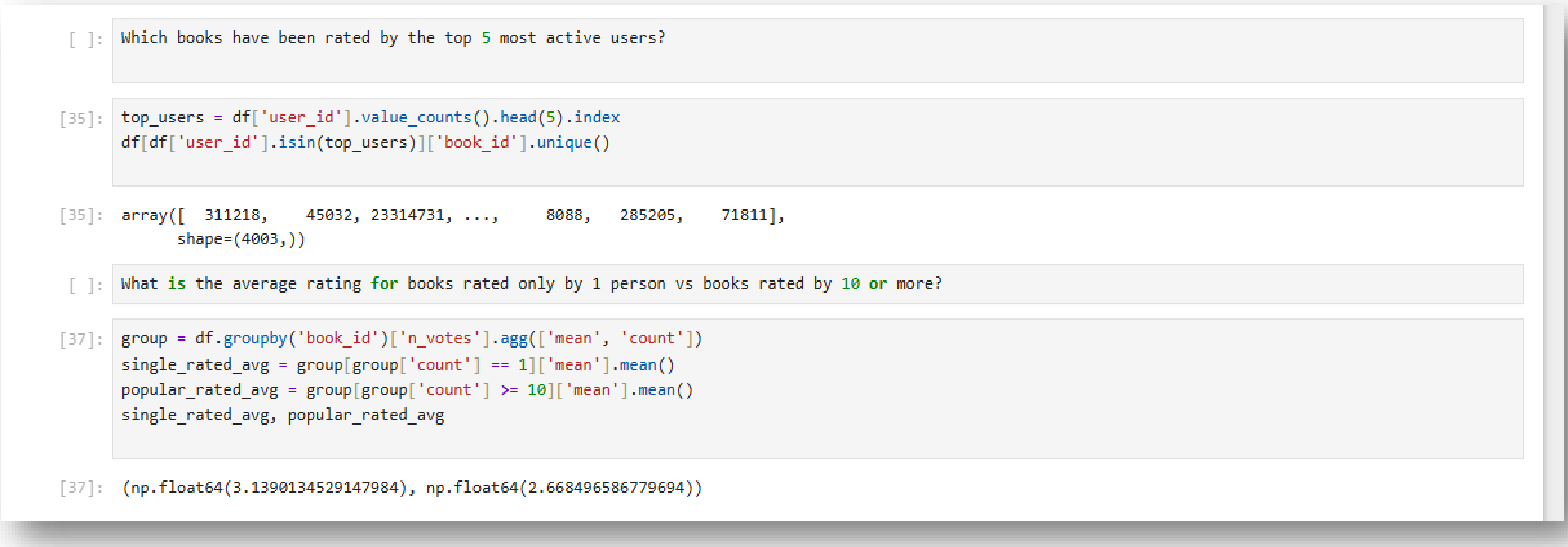
```
[34]: df.loc[df['n_comments'].idxmax(), 'review_text']
```



```
[ ]: Which review got the highest number of comments and what is its review_text?
```

```
[34]: df.loc[df['n_comments'].idxmax(), 'review_text']
```

```
[34]: "This was so much fun to read as a buddy read with my Goodreads friend, Laura. She kindly waited for my library copy to arrive and then was very patien
      t with my reading pace; we stayed practically in sync the entire way through the book, and that made the experience really fun. \n I love J.K. Rowling,
      as a writer and a storyteller, and as a person, and I'll most likely read any book she writes. \n I do prefer cozy and/or humorous mysteries. This one
      had too much gore for me, though thankfully the violence was off the page. There was way too much sexual perversion for me too, most notably in the pag
      es of a fictional book within the book, a book I found repulsive. Though the reader is supposed to feel this way, I still didn't enjoy reading excerpt
      s. I have to say that Rowling knows her mythology though, just as she showed in the Harry Potter books. \n I do love the character Robin and I hope she
      appears more and more frequently with each new book in the series. My favorite parts these books are when Robin is present. I also really enjoy Robin's
      mother and would like to see much more of her. It's Robin that has me in love with this series, even though I am loving Strike more and more. \n In thi
      s book I had great interest in many of the characters. I enjoyed trying to solve the mystery. The resolution is so much fun and is incredibly brillian
      t. If it weren't for the above mentioned gore and perversion, and if I tended to reread mysteries, I'd be tempted to reread this to see how she did it
      all. The reveal was one of the possibilities I'd considered but I'd missed the most important clues. There were many red herrings and some things that
      couldn't be guessed, but Rowling played fair and it wasn't impossible to guess the identity of the main culprit. (I'm sure she wouldn't be happy about
      it but I do think of this series as by Rowling vs. Galbraith.) \n I really enjoyed the British words and phrases used. I also enjoyed the London and ot
      her British settings. I felt as though I got in a bit of armchair travel. The words and phrases, some of the cultural things, well they seem so exotic
      to me, so English. It has me wondering whether American vs. British (vs. other nationalities of) readers have different experiences reading these book
      s. I also wondered that with the Harry Potter books. \n So, this book is hard for me to rate. All along it could have been as low as a three and as hig
      h as a five. I think it ended up as a 4.5 for me. I guess I have to round down even though Robin is one of my favorite fictional characters and I thoug
      ht the mystery was very well done. The book is a little too hard core for my tastes. My favorite mysteries are cozy mysteries, humorous cozy mysteries.
      \n 4 1/2 stars \n ETA: I should have taken notes. There is a long, spoiler ridden buddy read conversation at my Goodreads' review page. \n So, I did wa
      nt to include that I enjoyed reading about the writers, editors, and publishers that that world, a world that Rowling obviously knows. Though highly fi
      ctionalized, I certainly hope. I think there was truth there, and I think Rowling probably had a blast writing this book!"
```



```
[ ]: Which books have been rated by the top 5 most active users?
```

```
[35]: top_users = df['user_id'].value_counts().head(5).index
      df[df['user_id'].isin(top_users)]['book_id'].unique()
```

```
[35]: array([  311218,    45032, 23314731, ...,     8088,   285205,    71811],
            shape=(4003,))
```

```
[ ]: What is the average rating for books rated only by 1 person vs books rated by 10 or more?
```

```
[37]: group = df.groupby('book_id')['n_votes'].agg(['mean', 'count'])
      single_rated_avg = group[group['count'] == 1]['mean'].mean()
      popular_rated_avg = group[group['count'] >= 10]['mean'].mean()
      single_rated_avg, popular_rated_avg
```

```
[37]: (np.float64(3.1390134529147984), np.float64(2.668496586779694))
```