Name: Tejas Narayan Mahamuni

Roll No: CS2-15

PRN: 202401040036

Essentials of Data-Science

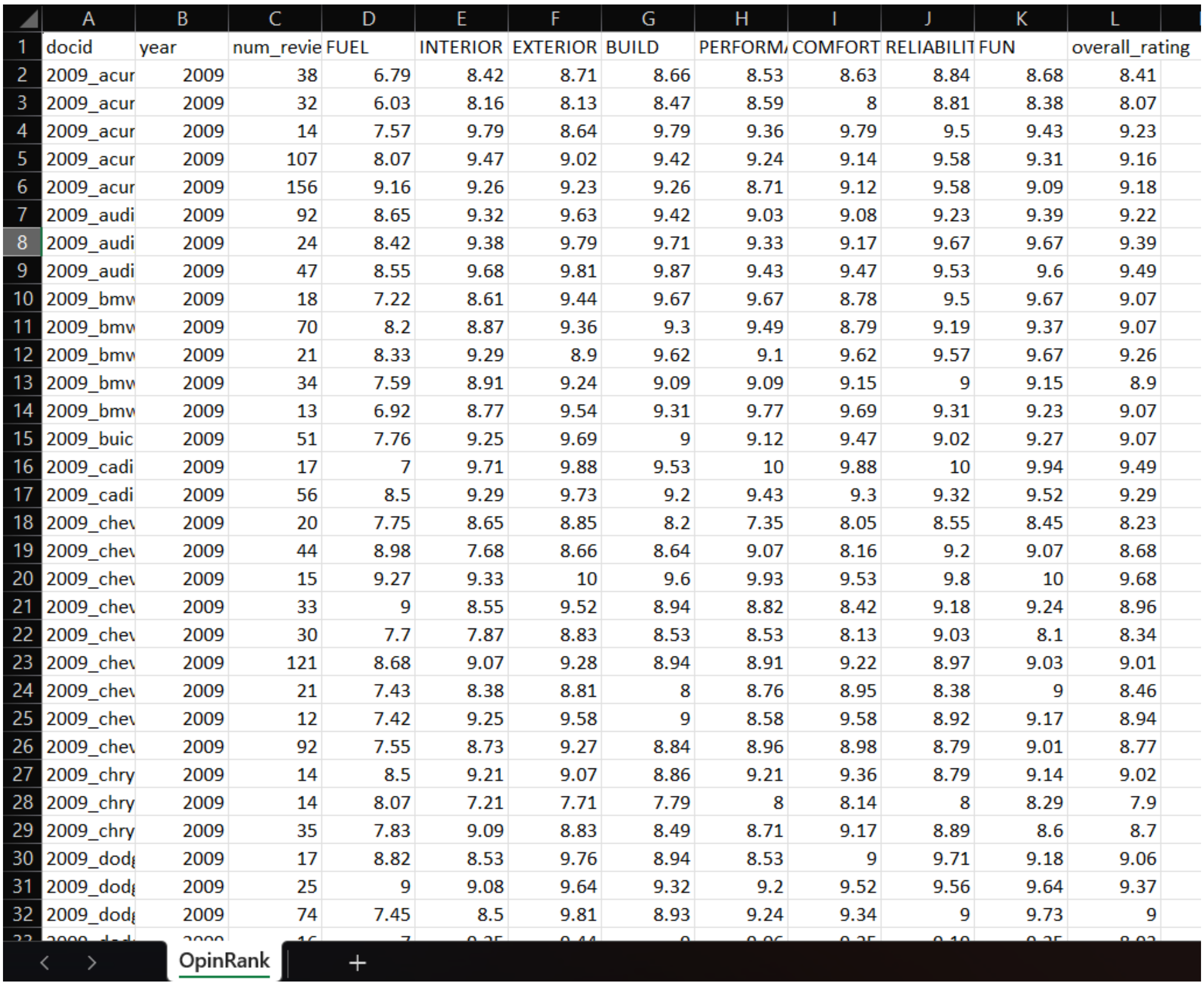
Theory Activity No. 1

Formulated 20 Problem Statements for "OpinRank.csv"

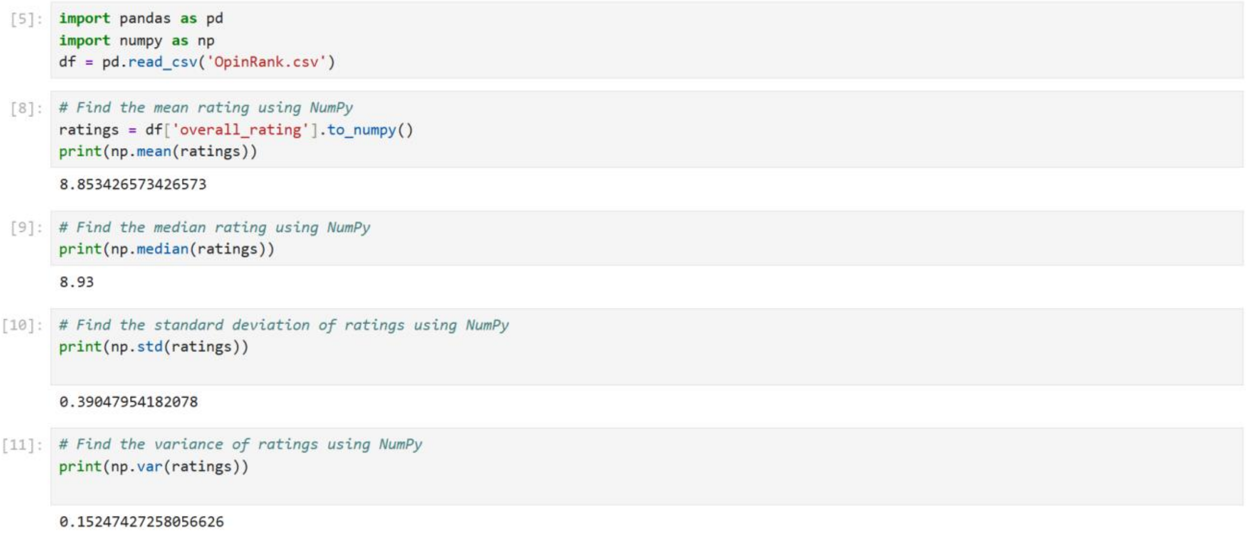
NumPy-Based Solutions (10 Problems)

Pandas-Based Solutions (10 Problems)

OpinRank.csv Dataset



	A	B	C	D	E	F	G	H	I	J	K	L
1	docid	year	num_revie	FUEL	INTERIOR	EXTERIOR	BUILD	PERFORM/	COMFORT	RELIABILIT	FUN	overall_rating
2	2009_acur	2009	38	6.79	8.42	8.71	8.66	8.53	8.63	8.84	8.68	8.41
3	2009_acur	2009	32	6.03	8.16	8.13	8.47	8.59	8	8.81	8.38	8.07
4	2009_acur	2009	14	7.57	9.79	8.64	9.79	9.36	9.79	9.5	9.43	9.23
5	2009_acur	2009	107	8.07	9.47	9.02	9.42	9.24	9.14	9.58	9.31	9.16
6	2009_acur	2009	156	9.16	9.26	9.23	9.26	8.71	9.12	9.58	9.09	9.18
7	2009_audi	2009	92	8.65	9.32	9.63	9.42	9.03	9.08	9.23	9.39	9.22
8	2009_audi	2009	24	8.42	9.38	9.79	9.71	9.33	9.17	9.67	9.67	9.39
9	2009_audi	2009	47	8.55	9.68	9.81	9.87	9.43	9.47	9.53	9.6	9.49
10	2009_bmw	2009	18	7.22	8.61	9.44	9.67	9.67	8.78	9.5	9.67	9.07
11	2009_bmw	2009	70	8.2	8.87	9.36	9.3	9.49	8.79	9.19	9.37	9.07
12	2009_bmw	2009	21	8.33	9.29	8.9	9.62	9.1	9.62	9.57	9.67	9.26
13	2009_bmw	2009	34	7.59	8.91	9.24	9.09	9.09	9.15	9	9.15	8.9
14	2009_bmw	2009	13	6.92	8.77	9.54	9.31	9.77	9.69	9.31	9.23	9.07
15	2009_buic	2009	51	7.76	9.25	9.69	9	9.12	9.47	9.02	9.27	9.07
16	2009_cadi	2009	17	7	9.71	9.88	9.53	10	9.88	10	9.94	9.49
17	2009_cadi	2009	56	8.5	9.29	9.73	9.2	9.43	9.3	9.32	9.52	9.29
18	2009_chev	2009	20	7.75	8.65	8.85	8.2	7.35	8.05	8.55	8.45	8.23
19	2009_chev	2009	44	8.98	7.68	8.66	8.64	9.07	8.16	9.2	9.07	8.68
20	2009_chev	2009	15	9.27	9.33	10	9.6	9.93	9.53	9.8	10	9.68
21	2009_chev	2009	33	9	8.55	9.52	8.94	8.82	8.42	9.18	9.24	8.96
22	2009_chev	2009	30	7.7	7.87	8.83	8.53	8.53	8.13	9.03	8.1	8.34
23	2009_chev	2009	121	8.68	9.07	9.28	8.94	8.91	9.22	8.97	9.03	9.01
24	2009_chev	2009	21	7.43	8.38	8.81	8	8.76	8.95	8.38	9	8.46
25	2009_chev	2009	12	7.42	9.25	9.58	9	8.58	9.58	8.92	9.17	8.94
26	2009_chev	2009	92	7.55	8.73	9.27	8.84	8.96	8.98	8.79	9.01	8.77
27	2009_chry	2009	14	8.5	9.21	9.07	8.86	9.21	9.36	8.79	9.14	9.02
28	2009_chry	2009	14	8.07	7.21	7.71	7.79	8	8.14	8	8.29	7.9
29	2009_chry	2009	35	7.83	9.09	8.83	8.49	8.71	9.17	8.89	8.6	8.7
30	2009_dodg	2009	17	8.82	8.53	9.76	8.94	8.53	9	9.71	9.18	9.06
31	2009_dodg	2009	25	9	9.08	9.64	9.32	9.2	9.52	9.56	9.64	9.37
32	2009_dodg	2009	74	7.45	8.5	9.81	8.93	9.24	9.34	9	9.73	9
33	2009_dodg	2009	16	7	9.25	9.44	9	9.06	9.35	9.19	9.35	8.93



```
[5]: import pandas as pd
     import numpy as np
     df = pd.read_csv('OpinRank.csv')
```

```
[8]: # Find the mean rating using NumPy
     ratings = df['overall_rating'].to_numpy()
     print(np.mean(ratings))
```

```
8.853426573426573
```

```
[9]: # Find the median rating using NumPy
     print(np.median(ratings))
```

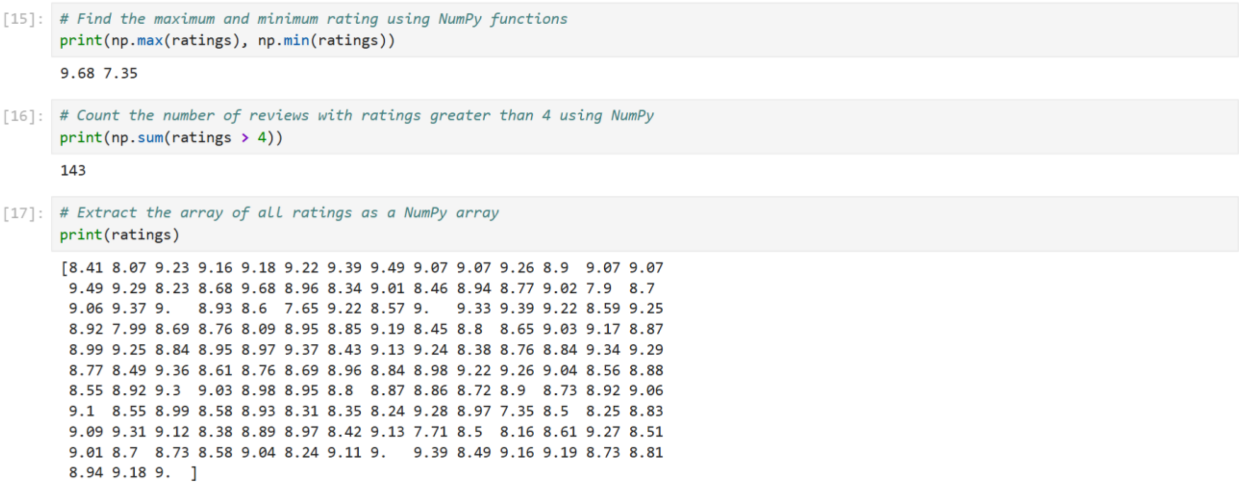
```
8.93
```

```
[10]: # Find the standard deviation of ratings using NumPy
      print(np.std(ratings))
```

```
0.39047954182078
```

```
[11]: # Find the variance of ratings using NumPy
      print(np.var(ratings))
```

```
0.15247427258056626
```



```
[15]: # Find the maximum and minimum rating using NumPy functions
      print(np.max(ratings), np.min(ratings))
```

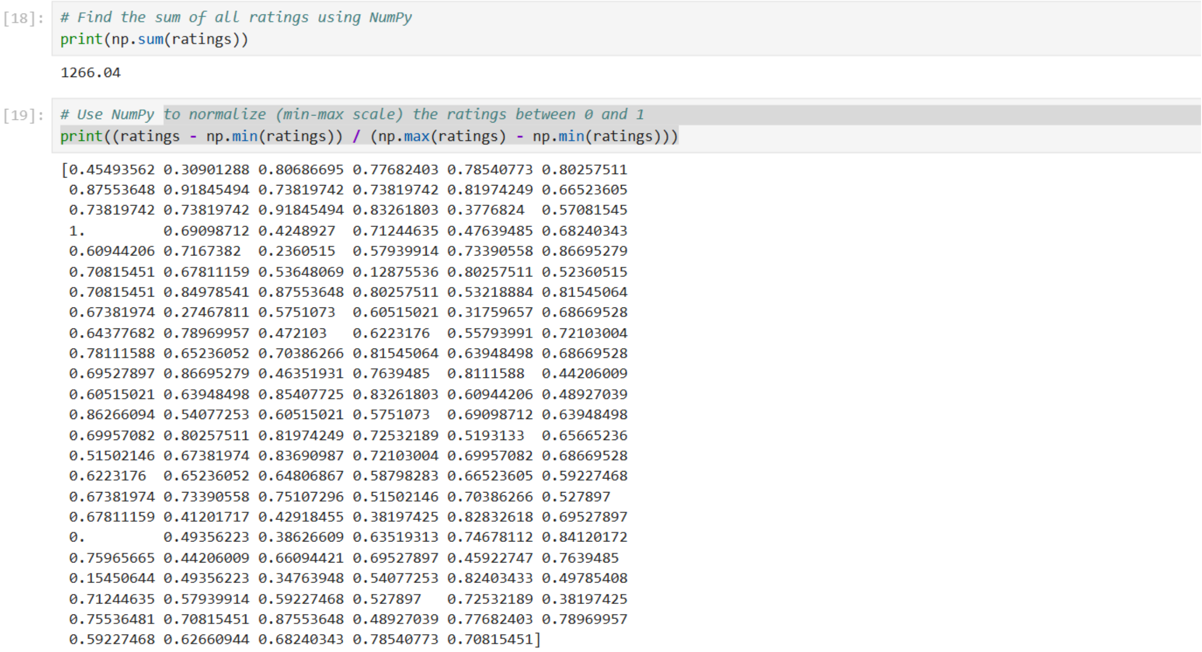
```
9.68 7.35
```

```
[16]: # Count the number of reviews with ratings greater than 4 using NumPy
      print(np.sum(ratings > 4))
```

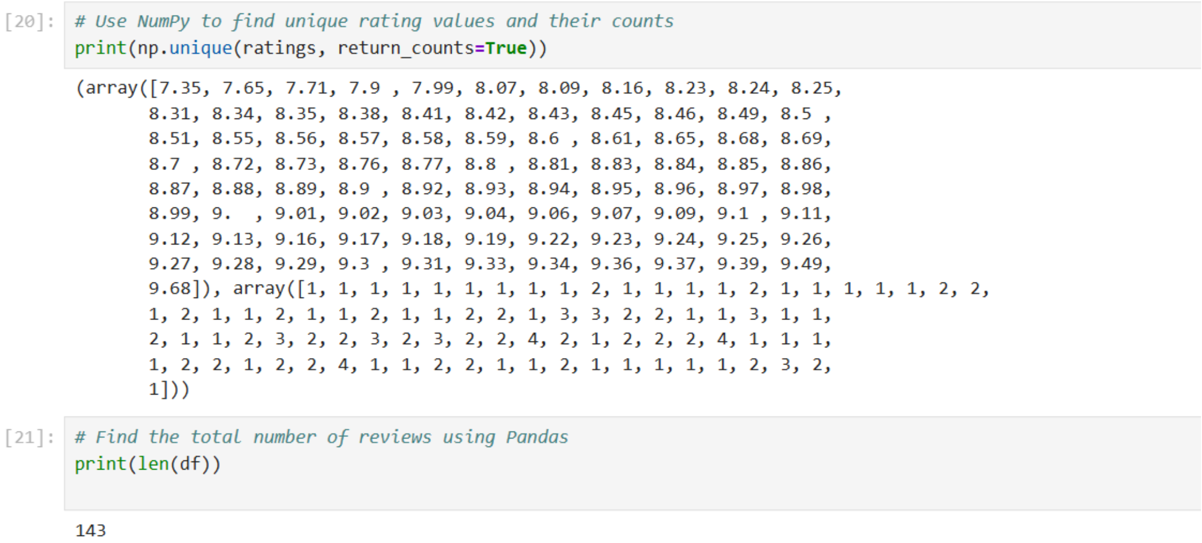
```
143
```

```
[17]: # Extract the array of all ratings as a NumPy array
      print(ratings)
```

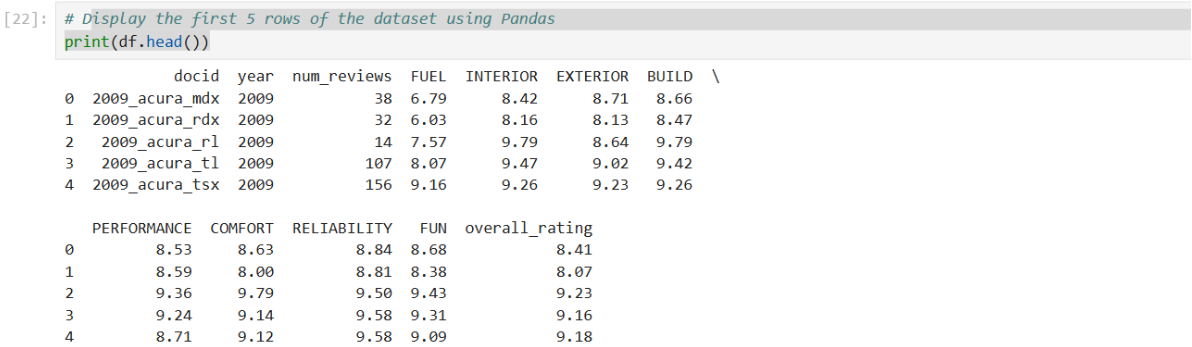
```
[8.41 8.07 9.23 9.16 9.18 9.22 9.39 9.49 9.07 9.07 9.26 8.9  9.07 9.07
 9.49 9.29 8.23 8.68 9.68 8.96 8.34 9.01 8.46 8.94 8.77 9.02 7.9  8.7
 9.06 9.37 9.   8.93 8.6  7.65 9.22 8.57 9.   9.33 9.39 9.22 8.59 9.25
 8.92 7.99 8.69 8.76 8.09 8.95 8.85 9.19 8.45 8.8  8.65 9.03 9.17 8.87
 8.99 9.25 8.84 8.95 8.97 9.37 8.43 9.13 9.24 8.38 8.76 8.84 9.34 9.29
 8.77 8.49 9.36 8.61 8.76 8.69 8.96 8.84 8.98 9.22 9.26 9.04 8.56 8.88
 8.55 8.92 9.3  9.03 8.98 8.95 8.8  8.87 8.86 8.72 8.9  8.73 8.92 9.06
 9.1  8.55 8.99 8.58 8.93 8.31 8.35 8.24 9.28 8.97 7.35 8.5  8.25 8.83
 9.09 9.31 9.12 8.38 8.89 8.97 8.42 9.13 7.71 8.5  8.16 8.61 9.27 8.51
 9.01 8.7  8.73 8.58 9.04 8.24 9.11 9.   9.39 8.49 9.16 9.19 8.73 8.81
 8.94 9.18 9.  ]
```



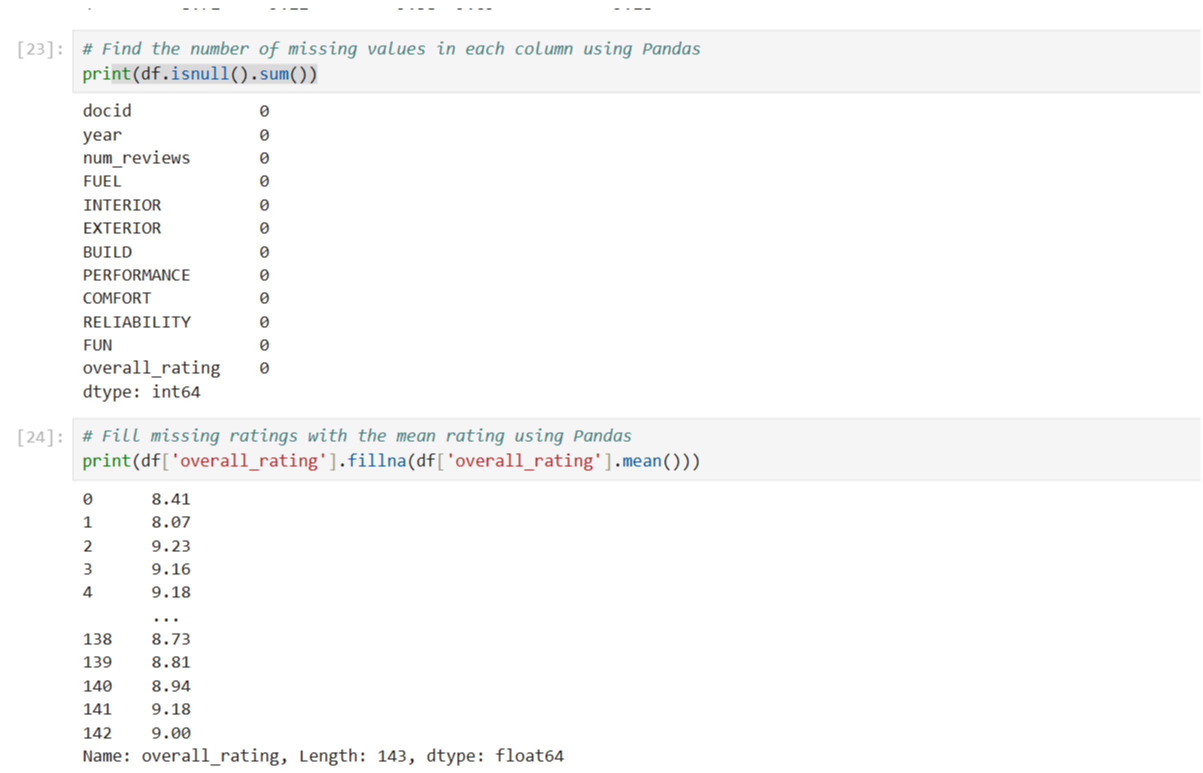
```
[18]: # Find the sum of all ratings using NumPy
      print(np.sum(ratings))

      1266.04

[19]: # Use NumPy to normalize (min-max scale) the ratings between 0 and 1
      print((ratings - np.min(ratings)) / (np.max(ratings) - np.min(ratings)))

      [0.45493562 0.30901288 0.80686695 0.77682403 0.78540773 0.80257511
       0.87553648 0.91845494 0.73819742 0.73819742 0.81974249 0.66523605
       0.73819742 0.73819742 0.91845494 0.83261803 0.3776824  0.57081545
       1.         0.69098712 0.4248927  0.71244635 0.47639485 0.68240343
       0.60944206 0.7167382  0.2360515  0.57939914 0.73390558 0.86695279
       0.70815451 0.67811159 0.53648069 0.12875536 0.80257511 0.52360515
       0.70815451 0.84978541 0.87553648 0.80257511 0.53218884 0.81545064
       0.67381974 0.27467811 0.5751073  0.60515021 0.31759657 0.68669528
       0.64377682 0.78969957 0.472103   0.6223176  0.55793991 0.72103004
       0.78111588 0.65236052 0.70386266 0.81545064 0.63948498 0.68669528
       0.69527897 0.86695279 0.46351931 0.7639485  0.8111588  0.44206009
       0.60515021 0.63948498 0.85407725 0.83261803 0.60944206 0.48927039
       0.86266094 0.54077253 0.60515021 0.5751073  0.69098712 0.63948498
       0.69957082 0.80257511 0.81974249 0.72532189 0.5193133  0.65665236
       0.51502146 0.67381974 0.83690987 0.72103004 0.69957082 0.68669528
       0.6223176  0.65236052 0.64806867 0.58798283 0.66523605 0.59227468
       0.67381974 0.73390558 0.75107296 0.51502146 0.70386266 0.527897
       0.67811159 0.41201717 0.42918455 0.38197425 0.82832618 0.69527897
       0.         0.49356223 0.38626609 0.63519313 0.74678112 0.84120172
       0.75965665 0.44206009 0.66094421 0.69527897 0.45922747 0.7639485
       0.15450644 0.49356223 0.34763948 0.54077253 0.82403433 0.49785408
       0.71244635 0.57939914 0.59227468 0.527897   0.72532189 0.38197425
       0.75536481 0.70815451 0.87553648 0.48927039 0.77682403 0.78969957
       0.59227468 0.62660944 0.68240343 0.78540773 0.70815451]
```



```
[20]: # Use NumPy to find unique rating values and their counts
      print(np.unique(ratings, return_counts=True))

      (array([7.35, 7.65, 7.71, 7.9 , 7.99, 8.07, 8.09, 8.16, 8.23, 8.24, 8.25,
             8.31, 8.34, 8.35, 8.38, 8.41, 8.42, 8.43, 8.45, 8.46, 8.49, 8.5 ,
             8.51, 8.55, 8.56, 8.57, 8.58, 8.59, 8.6 , 8.61, 8.65, 8.68, 8.69,
             8.7 , 8.72, 8.73, 8.76, 8.77, 8.8 , 8.81, 8.83, 8.84, 8.85, 8.86,
             8.87, 8.88, 8.89, 8.9 , 8.92, 8.93, 8.94, 8.95, 8.96, 8.97, 8.98,
             8.99, 9.  , 9.01, 9.02, 9.03, 9.04, 9.06, 9.07, 9.09, 9.1 , 9.11,
             9.12, 9.13, 9.16, 9.17, 9.18, 9.19, 9.22, 9.23, 9.24, 9.25, 9.26,
             9.27, 9.28, 9.29, 9.3 , 9.31, 9.33, 9.34, 9.36, 9.37, 9.39, 9.49,
             9.68]), array([1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 2,
             1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 2, 1, 3, 3, 2, 2, 1, 1, 3, 1, 1,
             2, 1, 1, 2, 3, 2, 2, 3, 2, 3, 2, 2, 4, 2, 1, 2, 2, 2, 4, 1, 1, 1,
             1, 2, 2, 1, 2, 2, 4, 1, 1, 2, 2, 1, 1, 2, 1, 1, 1, 1, 1, 2, 3, 2,
             1]))

[21]: # Find the total number of reviews using Pandas
      print(len(df))

      143
```



```
[22]: # Display the first 5 rows of the dataset using Pandas
      print(df.head())

                 docid  year  num_reviews  FUEL  INTERIOR  EXTERIOR  BUILD  \
      0  2009_acura_mdx  2009           38  6.79      8.42      8.71   8.66
      1  2009_acura_rdx  2009           32  6.03      8.16      8.13   8.47
      2   2009_acura_rl  2009           14  7.57      9.79      8.64   9.79
      3   2009_acura_tl  2009          107  8.07      9.47      9.02   9.42
      4  2009_acura_tsx  2009          156  9.16      9.26      9.23   9.26

         PERFORMANCE  COMFORT  RELIABILITY   FUN  overall_rating
      0         8.53     8.63         8.84  8.68            8.41
      1         8.59     8.00         8.81  8.38            8.07
      2         9.36     9.79         9.50  9.43            9.23
      3         9.24     9.14         9.58  9.31            9.16
      4         8.71     9.12         9.58  9.09            9.18
```

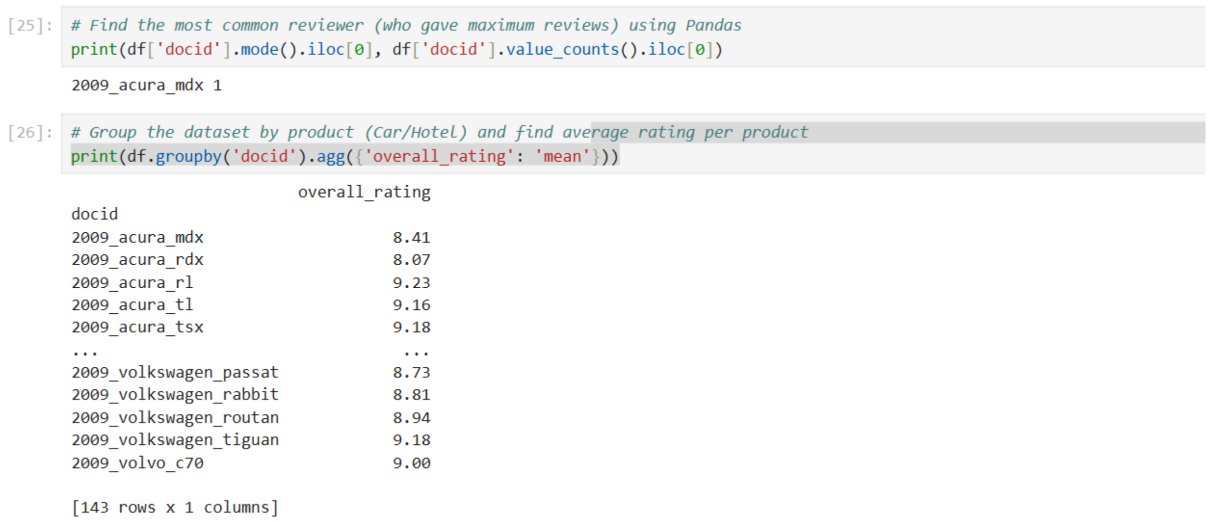


```
[23]: # Find the number of missing values in each column using Pandas
      print(df.isnull().sum())
```

```
docid             0
year              0
num_reviews       0
FUEL              0
INTERIOR          0
EXTERIOR          0
BUILD             0
PERFORMANCE       0
COMFORT           0
RELIABILITY       0
FUN               0
overall_rating    0
dtype: int64
```

```
[24]: # Fill missing ratings with the mean rating using Pandas
      print(df['overall_rating'].fillna(df['overall_rating'].mean()))
```

```
0      8.41
1      8.07
2      9.23
3      9.16
4      9.18
       ... 
138    8.73
139    8.81
140    8.94
141    9.18
142    9.00
Name: overall_rating, Length: 143, dtype: float64
```



```
[25]: # Find the most common reviewer (who gave maximum reviews) using Pandas
      print(df['docid'].mode().iloc[0], df['docid'].value_counts().iloc[0])
```

```
2009_acura_mdx 1
```

```
[26]: # Group the dataset by product (Car/Hotel) and find average rating per product
      print(df.groupby('docid').agg({'overall_rating': 'mean'}))
```

```
                       overall_rating
docid                                
2009_acura_mdx                   8.41
2009_acura_rdx                   8.07
2009_acura_rl                    9.23
2009_acura_tl                    9.16
2009_acura_tsx                   9.18
...                               ...
2009_volkswagen_passat           8.73
2009_volkswagen_rabbit           8.81
2009_volkswagen_routan           8.94
2009_volkswagen_tiguan           9.18
2009_volvo_c70                   9.00

[143 rows x 1 columns]
```

```
[27]: # Create a new column "Review_Length" showing number of words in Review_Text
      print(df['docid'].str.len())
```

```
0      14
1      14
2      13
3      13
4      14
       ..
138    22
139    22
140    22
141    22
142    14
Name: docid, Length: 143, dtype: int64
```

```
[28]: # Find the review with the maximum number of words in Review_Text using Pandas
      print(df.loc[df['docid'].str.len().idxmax()])
```

```
docid             2009_chrysler_town_and_country
year                                        2009
num_reviews                                   35
FUEL                                        7.83
INTERIOR                                    9.09
EXTERIOR                                    8.83
BUILD                                       8.49
PERFORMANCE                                 8.71
COMFORT                                     9.17
RELIABILITY                                 8.89
FUN                                          8.6
overall_rating                               8.7
Name: 27, dtype: object
```

```
[29]: # Sort all reviews by rating in descending order using Pandas
      print(df.sort_values('overall_rating', ascending=False))
```

```
                        docid  year  num_reviews  FUEL  INTERIOR  EXTERIOR  \
18      2009_chevrolet_corvette  2009           15  9.27      9.33     10.00
14          2009_cadillac_cts-v  2009           17  7.00      9.71      9.88
7                  2009_audi_q5  2009           47  8.55      9.68      9.81
38             2009_ford_f-150  2009           52  8.48      9.69      9.46
134         2009_volkswagen_cc  2009           94  8.96      9.53      9.86
..                         ...   ...          ...   ...       ...       ...
43            2009_ford_ranger  2009           21  7.33      7.71      8.24
26       2009_chrysler_sebring  2009           14  8.07      7.21      7.71
120   2009_suzuki_grand_vitara  2009           12  6.75      8.00      8.83
33          2009_dodge_journey  2009          150  6.91      7.63      8.66
108           2009_saturn_aura  2009           16  7.69      7.06      8.50

     BUILD  PERFORMANCE  COMFORT  RELIABILITY    FUN  overall_rating
18    9.60         9.93     9.53         9.80  10.00            9.68
14    9.53        10.00     9.88        10.00   9.94            9.49
7     9.87         9.43     9.47         9.53   9.60            9.49
38    9.44         9.33     9.50         9.69   9.56            9.39
134   9.26         9.34     9.37         9.39   9.44            9.39
..     ...          ...      ...          ...    ...             ...
43    8.52         7.81     7.14         8.90   8.29            7.99
26    7.79         8.00     8.14         8.00   8.29            7.90
120   7.67         7.25     7.92         7.50   7.75            7.71
33    7.06         7.44     8.35         7.33   7.82            7.65
108   7.19         7.06     7.44         6.94   6.94            7.35

[143 rows x 12 columns]
```

```
[30]: # Create a pivot table showing average rating per reviewer using Pandas
      print(pd.pivot_table(df, values='overall_rating', index='docid', aggfunc='mean'))
```

```
                         overall_rating
docid
2009_acura_mdx                     8.41
2009_acura_rdx                     8.07
2009_acura_rl                      9.23
2009_acura_tl                      9.16
2009_acura_tsx                     9.18
...                                 ...
2009_volkswagen_passat             8.73
2009_volkswagen_rabbit             8.81
2009_volkswagen_routan             8.94
2009_volkswagen_tiguan             9.18
2009_volvo_c70                     9.00

[143 rows x 1 columns]
```

```
[ ]:
```