Part 1 Question 1: Analyze the ramen data

Tasks:

In [ ]:
```
import pandas as pd
```

1. Read the data from the CSV file into a DataFrame.

In [ ]:
```
df = pd.read_csv('datasets/ramen-ratings.csv')
df
```

Out[ ]:

	Brand	Variety	Style	Country	Stars
0	New Touch	T's Restaurant Tantanmen	Cup	Japan	3.75
1	Just Way	Noodles Spicy Hot Sesame Spicy Hot Sesame Guan...	Pack	Taiwan	1.00
2	Nissin	Cup Noodles Chicken Vegetable	Cup	USA	2.25
3	Wei Lih	GGE Ramen Snack Tomato Flavor	Pack	Taiwan	2.75
4	Ching's Secret	Singapore Curry	Pack	India	3.75
...	...	...	...	...	...
2572	Vifon	Hu Tiu Nam Vang ["Phnom Penh" style] Asian Sty...	Bowl	Vietnam	3.50
2573	Wai Wai	Oriental Style Instant Noodles	Pack	Thailand	1.00
2574	Wai Wai	Tom Yum Shrimp	Pack	Thailand	2.00
2575	Wai Wai	Tom Yum Chili Flavor	Pack	Thailand	2.00
2576	Westbrae	Miso Ramen	Pack	USA	0.50

2577 rows × 5 columns

2. Display the first five rows of data.

In [ ]:
```
df.head(5)
```

Out[ ]:

	Brand	Variety	Style	Country	Stars
0	New Touch	T's Restaurant Tantanmen	Cup	Japan	3.75
1	Just Way	Noodles Spicy Hot Sesame Spicy Hot Sesame Guan...	Pack	Taiwan	1.00
2	Nissin	Cup Noodles Chicken Vegetable	Cup	USA	2.25
3	Wei Lih	GGE Ramen Snack Tomato Flavor	Pack	Taiwan	2.75
4	Ching's Secret	Singapore Curry	Pack	India	3.75

3. Display the last five rows of data.

In [ ]:
```
df.tail(5)
```

Out[ ]:

	Brand	Variety	Style	Country	Stars
2572	Vifon	Hu Tiu Nam Vang ["Phnom Penh" style] Asian Sty...	Bowl	Vietnam	3.5
2573	Wai Wai	Oriental Style Instant Noodles	Pack	Thailand	1.0
2574	Wai Wai	Tom Yum Shrimp	Pack	Thailand	2.0
2575	Wai Wai	Tom Yum Chili Flavor	Pack	Thailand	2.0
2576	Westbrae	Miso Ramen	Pack	USA	0.5

4. Display statistical information for the numeric columns using the describe() method.

In [ ]:
```
df.describe()
```

Out[ ]:

	Stars
count	2577.000000
mean	3.654676
std	1.015331
min	0.000000
25%	3.250000
50%	3.750000
75%	4.250000
max	5.000000

5. Display the number of unique values for each column.

In [ ]:
```
df.nunique()
```

Out[ ]:
```
Brand       355
Variety    2410
Style         7
Country      38
Stars        42
dtype: int64
```

6. Display only rows where the country is Vietnam.

In [ ]:
```
df[df['Country'] == 'Vietnam']
```

Out[ ]:

	Brand	Variety	Style	Country	Stars
18	Binh Tay	Mi Hai Cua	Pack	Vietnam	4.00
52	Uni-President	Mushroom Flavor	Pack	Vietnam	0.00
143	Mum Ngon	Lau Tom Chua Cay	Pack	Vietnam	3.50
224	Vifon	Viet Cuisine Bun Rieu Cua Sour Crab Soup Insta...	Bowl	Vietnam	5.00
365	Acecook	Oh! Ricey Pork Flavour	Pack	Vietnam	4.00
...	...	...	...	...	...
2486	Binh Tay	Mi Chay Mushroom	Pack	Vietnam	2.75
2535	Ve Wong	Kung-Fu Chicken Flavor	Pack	Vietnam	2.75
2570	Ve Wong	Mushroom Pork	Pack	Vietnam	1.00
2571	Vifon	Nam Vang	Pack	Vietnam	2.50
2572	Vifon	Hu Tiu Nam Vang ["Phnom Penh" style] Asian Sty...	Bowl	Vietnam	3.50

108 rows × 5 columns

7. Display only the Brand and Style columns.

In [ ]:
```
df[['Brand', 'Style']]
```

Out[ ]:

	Brand	Style
0	New Touch	Cup
1	Just Way	Pack
2	Nissin	Cup
3	Wei Lih	Pack
4	Ching's Secret	Pack
...	...	...
2572	Vifon	Bowl
2573	Wai Wai	Pack
2574	Wai Wai	Pack
2575	Wai Wai	Pack
2576	Westbrae	Pack

2577 rows × 2 columns

8. Display only the Country column.

In [ ]:

```
df['Country']
```

Out[ ]:

```
0          Japan
1         Taiwan
2            USA
3         Taiwan
4          India
          ...
2572     Vietnam
2573    Thailand
2574    Thailand
2575    Thailand
2576         USA
Name: Country, Length: 2577, dtype: object
```

9. Display the data after it has been sorted by the Stars column from high values to low values.

In [ ]:

```
df.sort_values(by='Stars', ascending=False)
```

Out[ ]:

	Brand	Variety	Style	Country	Stars
1585	Prima Taste	Singapore Laksa La Mian	Pack	Singapore	5.0
446	Maruchan	Instant Lunch Chipotle Chicken Flavor Ramen No...	Cup	USA	5.0
484	Nongshim	Champong Noodle Soup Spicy Seafood Flavor	Pack	South Korea	5.0
483	Nissin	Straits Kitchen Laksa	Pack	Singapore	5.0
1613	Nissin	Raoh Backfat Rich Soy Sauce Flavor	Bowl	Japan	5.0
...	...	...	...	...	...
522	Koyo	Garlic Pepper Reduced Sodium Ramen	Pack	USA	0.0
561	Samyang Foods	Honey & Cheese Big Bowl	Bowl	South Korea	0.0
950	Azami	Kimchee Noodle Soup	Cup	Canada	0.0
2079	Hsin Tung Yang	Tiny Noodle With Oyster Flavor	Pack	Taiwan	0.0
52	Uni-President	Mushroom Flavor	Pack	Vietnam	0.0

2577 rows × 5 columns

10. In the Country column replace “USA” with “United States” Make sure this change is saved in the DataFrame and then display the first five rows to be sure the change was made correctly.

In [ ]:

```
df['Country'] = df['Country'].replace('USA', 'United States')
df.head()
```

Out[ ]:

	Brand	Variety	Style	Country	Stars
0	New Touch	T's Restaurant Tantanmen	Cup	Japan	3.75
1	Just Way	Noodles Spicy Hot Sesame Spicy Hot Sesame Guan...	Pack	Taiwan	1.00
2	Nissin	Cup Noodles Chicken Vegetable	Cup	United States	2.25
3	Wei Lih	GGE Ramen Snack Tomato Flavor	Pack	Taiwan	2.75
4	Ching's Secret	Singapore Curry	Pack	India	3.75

Questions:

1. How many countries are represented in the data?

In [ ]:
```
df['Country'].nunique()
```

Out[ ]:
```
37
```

There are 37 countries represented in the data

2. Which three countries have the highest average rating?

In [ ]:
```
df[['Country', 'Stars']].groupby('Country').mean().sort_values(by='Stars', a
```

Out[ ]:

Country	Stars
Brazil	4.350000
Sarawak	4.333333
Cambodia	4.200000

The 3 countries with the highest average rating are:

1. Brazil
2. Sarawak
3. Cambodia

3. Which three countries have the lowest average rating?

In [ ]:
```
df[['Country', 'Stars']].groupby('Country').mean().sort_values(by='Stars', a
```

Out[ ]:

Country	Stars
Netherlands	2.483333
Canada	2.243902
Nigeria	1.500000

The 3 countries with the lowest average rating are:

1. Netherlands
2. Canada
3. Nigeria

4. Which three countries have the most brands and how many brands does each of these countries have?

In [ ]:
```
df[['Country', 'Brand']].groupby('Country').count().sort_values(by='Brand',
```

Out[ ]:

	Brand
Country	
Japan	352
United States	324
South Korea	307

The 3 countries with the who have the most brands are:

1. Japan with **352** brands
2. United States with **324** brands
3. Cambodia with **307** brands

Part 1 Question 2: Analyze the avocado data

Tasks:

In [ ]:
```
import pandas as pd
```

1. Read the data from the CSV file into a DataFrame.

In [ ]:
```
df = pd.read_csv('datasets/avocado.csv')
df
```

Out[ ]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	To Ba
0	0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696
1	1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505
2	2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145
3	3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58	5811
4	4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78	6183
...	...	...	...	...	...	...	...	
18244	7	2018-02-04	1.63	17074.83	2046.96	1529.20	0.00	13498
18245	8	2018-01-28	1.71	13888.04	1191.70	3431.50	0.00	9264
18246	9	2018-01-21	1.87	13766.76	1191.92	2452.79	727.94	9394
18247	10	2018-01-14	1.93	16205.22	1527.63	2981.04	727.01	10969
18248	11	2018-01-07	1.62	17489.58	2894.77	2356.13	224.53	12014

18249 rows × 14 columns

2. Display type memory consumption and null count information using the info() method.

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18249 entries, 0 to 18248
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Unnamed: 0    18249 non-null  int64  
 1   Date          18249 non-null  object 
 2   AveragePrice  18249 non-null  float64
 3   Total Volume  18249 non-null  float64
 4   4046          18249 non-null  float64
 5   4225          18249 non-null  float64
 6   4770          18249 non-null  float64
 7   Total Bags    18249 non-null  float64
 8   Small Bags    18249 non-null  float64
 9   Large Bags    18249 non-null  float64
 10  XLarge Bags   18249 non-null  float64
 11  type          18249 non-null  object 
 12  year          18249 non-null  int64  
 13  region        18249 non-null  object 
dtypes: float64(9), int64(2), object(3)
memory usage: 1.9+ MB
```

3. Display the number of unique values in each column.

In [ ]:
```
df.nunique()
```

Out[ ]:
```
Unnamed: 0         53
Date              169
AveragePrice      259
Total Volume    18237
4046            17702
4225            18103
4770            12071
Total Bags      18097
Small Bags      17321
Large Bags      15082
XLarge Bags      5588
type                2
year                4
region             54
dtype: int64
```

4. Display all the rows of data that JupyterLab displays by default.

In [ ]:
```
df
```

Out[ ]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	To Ba
0	0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696
1	1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505
2	2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145
3	3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58	5811
4	4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78	6183
...	...	...	...	...	...	...	...	
18244	7	2018-02-04	1.63	17074.83	2046.96	1529.20	0.00	13498
18245	8	2018-01-28	1.71	13888.04	1191.70	3431.50	0.00	9264
18246	9	2018-01-21	1.87	13766.76	1191.92	2452.79	727.94	9394
18247	10	2018-01-14	1.93	16205.22	1527.63	2981.04	727.01	10969
18248	11	2018-01-07	1.62	17489.58	2894.77	2356.13	224.53	12014

18249 rows × 14 columns

5. Display the first and last five rows of data and the first and last four columns of data.

In [ ]:

```
pd.set_option('display.max_rows', 10)
pd.set_option('display.max_columns', 8)
df
```

Out[ ]:

	Unnamed: 0	Date	AveragePrice	Total Volume	...	XLarge Bags	type	year	
0	0	2015-12-27	1.33	64236.62	...	0.0	conventional	2015	
1	1	2015-12-20	1.35	54876.98	...	0.0	conventional	2015	
2	2	2015-12-13	0.93	118220.22	...	0.0	conventional	2015	
3	3	2015-12-06	1.08	78992.15	...	0.0	conventional	2015	
4	4	2015-11-29	1.28	51039.60	...	0.0	conventional	2015	
...	...	...	...	...	...	...	...	...	
18244	7	2018-02-04	1.63	17074.83	...	0.0	organic	2018	We
18245	8	2018-01-28	1.71	13888.04	...	0.0	organic	2018	We
18246	9	2018-01-21	1.87	13766.76	...	0.0	organic	2018	We
18247	10	2018-01-14	1.93	16205.22	...	0.0	organic	2018	We
18248	11	2018-01-07	1.62	17489.58	...	0.0	organic	2018	We

18249 rows × 14 columns

6. Choose any three columns access them with bracket notation and display the first five rows of this data.

In [ ]:

```
df[['Date', 'AveragePrice', 'Total Volume']].head(5)
```

Out[ ]:

	Date	AveragePrice	Total Volume
0	2015-12-27	1.33	64236.62
1	2015-12-20	1.35	54876.98
2	2015-12-13	0.93	118220.22
3	2015-12-06	1.08	78992.15
4	2015-11-29	1.28	51039.60

7. Select one column and access it with dot notation.

In [ ]:
```
df.Date
```

Out[ ]:
```
0        2015-12-27
1        2015-12-20
2        2015-12-13
3        2015-12-06
4        2015-11-29
            ...
18244    2018-02-04
18245    2018-01-28
18246    2018-01-21
18247    2018-01-14
18248    2018-01-07
Name: Date, Length: 18249, dtype: object
```

8. Multiply the Total Volume and AveragePrice columns and store the result in a new column called EstimatedRevenue. Then display the first five rows of this data to confirm that the column was added and has the correct values.

In [ ]:
```
df['EstimatedRevenue'] = df['AveragePrice'] * df['Total Volume']
df.head(5)
```

Out[ ]:

	Unnamed: 0	Date	AveragePrice	Total Volume	...	type	year	region	Estimate
0	0	2015-12-27	1.33	64236.62	...	conventional	2015	Albany	8
1	1	2015-12-20	1.35	54876.98	...	conventional	2015	Albany	7
2	2	2015-12-13	0.93	118220.22	...	conventional	2015	Albany	10
3	3	2015-12-06	1.08	78992.15	...	conventional	2015	Albany	8
4	4	2015-11-29	1.28	51039.60	...	conventional	2015	Albany	6

5 rows × 15 columns

9. Create a DataFrame that's grouped by region and type and that includes the average price for the grouped columns. Then reset the index and display the first five rows.

In [ ]:
```
df[['region', 'type', 'AveragePrice']].groupby(['region', 'type']).mean().re
```

Out[ ]:

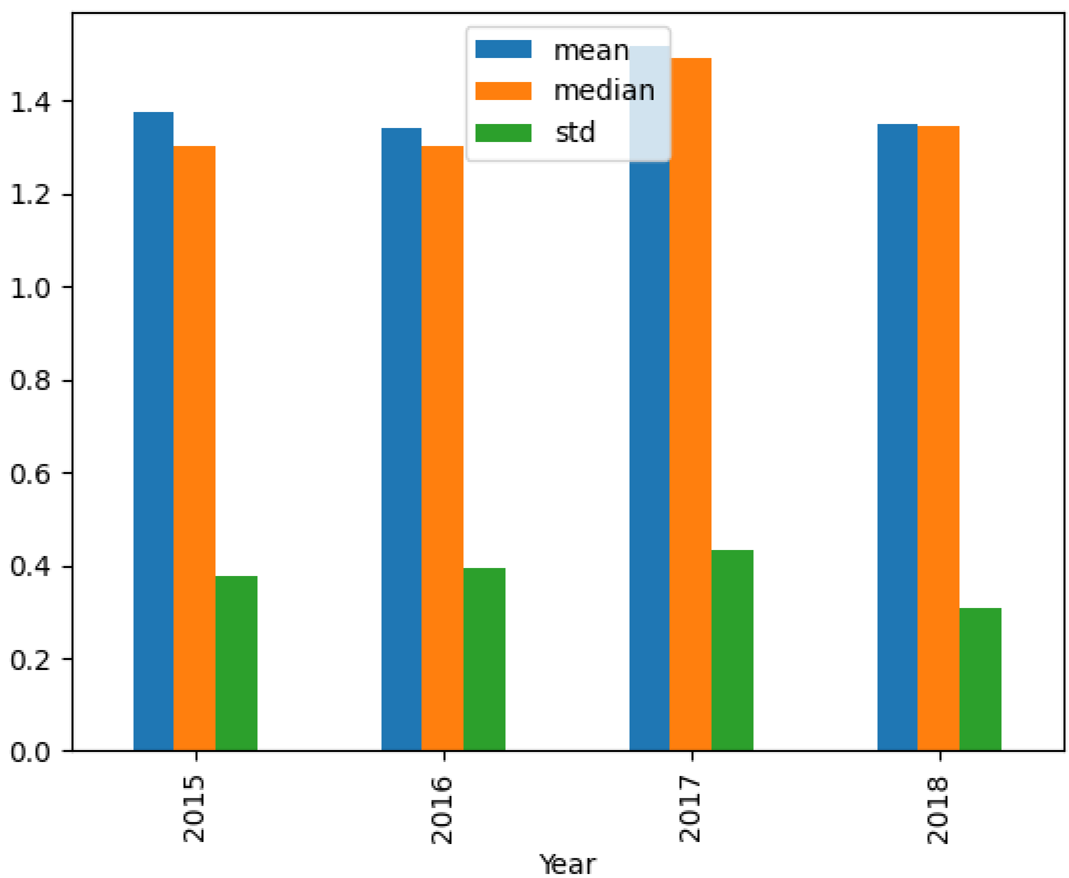
	region	type	AveragePrice
0	Albany	conventional	1.348757
1	Albany	organic	1.773314
2	Atlanta	conventional	1.068817
3	Atlanta	organic	1.607101
4	BaltimoreWashington	conventional	1.344201

10. Create a bar plot that shows the mean median and standard deviation of the Total Volume column by year.

In [ ]:

```
df['Year'] = pd.to_datetime(df['Date']).dt.year
plot_df = df[['Year', 'AveragePrice']].groupby('Year').agg(['mean', 'median'
plot_df.plot.bar(x='Year', y='AveragePrice')
```

Out[ ]: <Axes: xlabel='Year'>



Questions:

1. How many unique regions are there?

In [ ]:
```
df['region'].nunique()
```

Out[ ]: 54

There are 54 unique regions in the dataset

2. What is the average price for each type of avocado (organic and conventional)? Be sure to include just the type and AveragePrice columns in the results.

In [ ]:
```
df[['type', 'AveragePrice']].groupby('type').mean().reset_index()
```

Out[ ]:

	type	AveragePrice
0	conventional	1.158040
1	organic	1.653999

The average price for each type of avocado is as follows:

1. conventional - $1.16
2. organic - $1.65

3. Which region has the lowest average price for organic avocados? Hint: Create wide data from the grouped data that you created in task 8.

In [ ]:
```
wide_df = df[['region', 'type', 'AveragePrice']].groupby(['region', 'type'])
wide_df[wide_df['type'] == 'organic'].sort_values('AveragePrice', ascending=
```

Out[ ]:

	region	type	AveragePrice
37	Houston	organic	1.270769

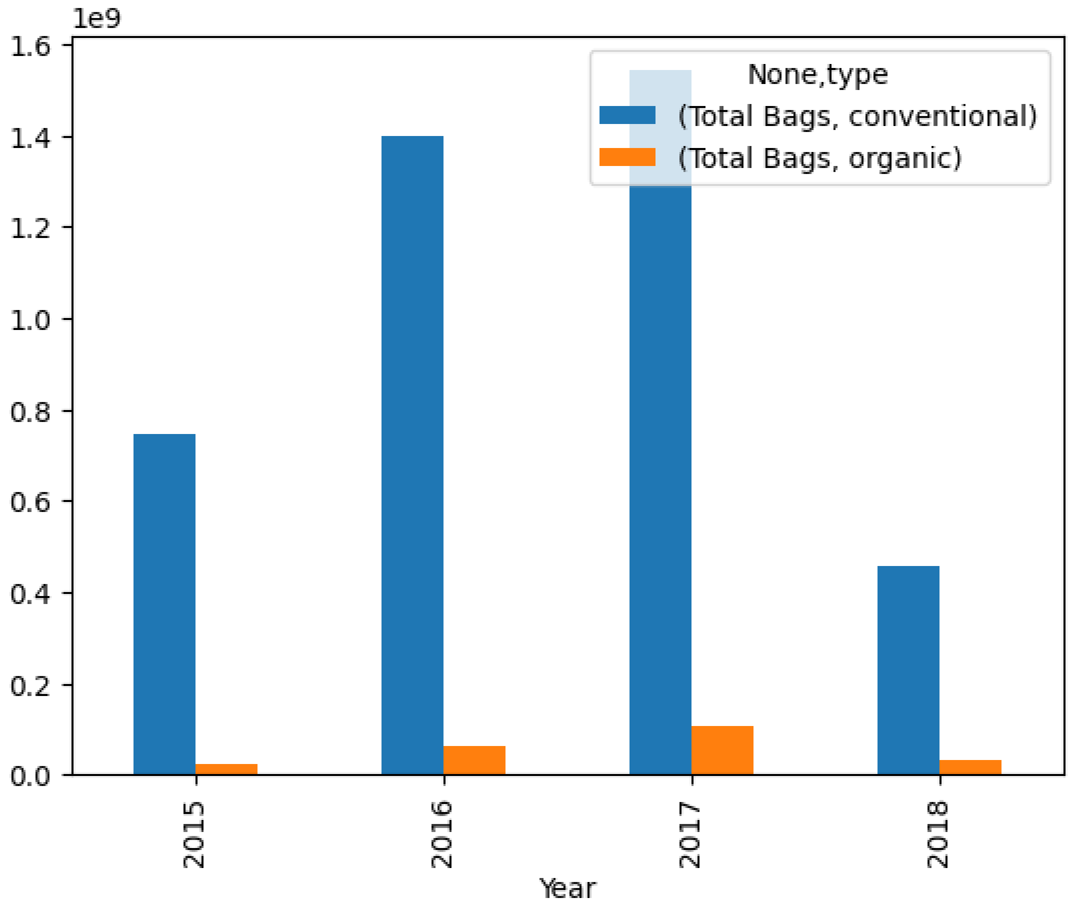
The region with the lowest average organic avocado price is Houstan at $1.27

4. Have the Total Bags sold per year of each type of avocado become more or less consistent over time?

In [ ]:
```
plot_df = df[['Year', 'type', 'Total Bags']].groupby(['Year', 'type']).sum()

plot_df.plot.bar()
```

Out[ ]: <Axes: xlabel='Year'>



The graph above represents the total bags sold per year of each type of avocado. From this graph it is clear that the total bags sold per year has become less consistent over time, this can be seen with the significant drop from 2017 to 2018.

Part 1 Question 3: Analyze the exam data

Tasks:

In [ ]:
```
import pandas as pd
```

1. Read the data from the CSV file into a DataFrame and display the first five rows.

In [ ]:
```
df = pd.read_csv('datasets/exams.csv')
df.head()
```

Out[ ]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writin scor
0	female	group B	bachelor's degree	standard	none	72	72	7
1	female	group C	some college	standard	completed	69	90	8
2	female	group B	master's degree	standard	none	90	95	9
3	male	group A	associate's degree	free/reduced	none	47	57	4
4	male	group C	some college	standard	none	76	78	7

2. Display the basic information for the DataFrame and its columns using the info() method.

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column                       Non-Null Count  Dtype 
---  ------                       --------------  ----- 
 0   gender                       1000 non-null   object
 1   race/ethnicity               1000 non-null   object
 2   parental level of education  1000 non-null   object
 3   lunch                        1000 non-null   object
 4   test preparation course      1000 non-null   object
 5   math score                   1000 non-null   int64 
 6   reading score                1000 non-null   int64 
 7   writing score                1000 non-null   int64 
dtypes: int64(3), object(5)
memory usage: 62.6+ KB
```

3. Display statistical information for the math score reading score and writing score columns using the describe() method.

In [ ]:
```
df[['math score', 'reading score', 'writing score']].describe()
```

Out[ ]:

	math score	reading score	writing score
count	1000.00000	1000.000000	1000.000000
mean	66.08900	69.169000	68.054000
std	15.16308	14.600192	15.195657
min	0.00000	17.000000	10.000000
25%	57.00000	59.000000	57.750000
50%	66.00000	70.000000	69.000000
75%	77.00000	79.000000	79.000000
max	100.00000	100.000000	100.000000

4. Group the data by the race/ethnicity column and display the mean scores.

In [ ]:
```
df[['race/ethnicity', 'math score', 'reading score', 'writing score']].group
```

Out[ ]:

	math score	reading score	writing score
race/ethnicity			
group A	61.629213	64.674157	62.674157
group B	63.452632	67.352632	65.600000
group C	64.463950	69.103448	67.827586
group D	67.362595	70.030534	70.145038
group E	73.821429	73.028571	71.407143

5. Display a single column as a DataFrame with bracket notation.

In [ ]:

```
df['gender'].to_frame(name='gender')
```

Out[ ]:

	gender
0	female
1	female
2	female
3	male
4	male
...	...
995	female
996	male
997	female
998	female
999	female

1000 rows × 1 columns

6. Display a single column as a Series with bracket notation.

In [ ]:

```
df['gender']
```

```
Out[ ]: 0      female
        1      female
        2      female
        3        male
        4        male
                ...
        995    female
        996      male
        997    female
        998    female
        999    female
        Name: gender, Length: 1000, dtype: object
```

7. Display a single column as a Series with dot notation.

```
In [ ]: df.gender
```

```
Out[ ]: 0      female
        1      female
        2      female
        3        male
        4        male
                ...
        995    female
        996      male
        997    female
        998    female
        999    female
        Name: gender, Length: 1000, dtype: object
```

8. Display only rows for females with a math score greater than or equal to 90.

```
In [ ]: df[(df['gender'] == 'female')&(df['math score'] > 90)]
```

Out[ ]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	wri sc
114	female	group E	bachelor's degree	standard	completed	99	100	
165	female	group C	bachelor's degree	standard	completed	96	100	
179	female	group D	some high school	standard	completed	97	100	
263	female	group E	high school	standard	none	99	93	
451	female	group E	some college	standard	none	100	92	
458	female	group E	bachelor's degree	standard	none	100	100	
501	female	group B	associate's degree	standard	completed	94	87	
503	female	group E	associate's degree	standard	completed	95	89	
521	female	group C	associate's degree	standard	none	91	86	
546	female	group A	some high school	standard	completed	92	100	
566	female	group E	bachelor's degree	free/reduced	completed	92	100	
594	female	group C	bachelor's degree	standard	completed	92	100	
685	female	group E	master's degree	standard	completed	94	99	
712	female	group D	some college	standard	none	98	100	
717	female	group C	associate's degree	standard	completed	96	96	
855	female	group B	bachelor's degree	standard	none	97	97	
886	female	group E	associate's degree	standard	completed	93	100	
903	female	group D	bachelor's degree	free/reduced	completed	93	100	
957	female	group D	master's degree	standard	none	92	100	
962	female	group E	associate's	standard	none	100	100	

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	wri s
			degree					
979	female	group C	associate's degree	standard	none	91	95	

Questions:

1. Does taking a test preparation course improve average scores?

In [ ]:
```
df[['test preparation course', 'math score', 'reading score', 'writing score
```

Out[ ]:

	math score	reading score	writing score
test preparation course			
completed	69.695531	73.893855	74.418994
none	64.077882	66.534268	64.504673

It can be seen in the table above that the average score for all 3 categories was higher when the test preparation course was taken, so we can say that taking the test improves average scores.

2. Which gender is better on average at math?

In [ ]:
```
df[['gender', 'math score']].groupby('gender').mean()
```

Out[ ]:

	math score
gender	
female	63.633205
male	68.728216

From the table above we can see that males on average are better at math than females

3. Which gender is better on average at all three subjects? Hint: Start by adding a column to the DataFrame with the total score

In [ ]:
```
df['average score'] = df[['math score', 'reading score', 'writing score']].s
df[['gender', 'average score']].groupby('gender').mean()
```

Out[ ]:

	average score
gender	
female	69.569498
male	65.837483

From the table above we can see that on average in all 3 subjects, females are better than males.

Assignment 1

Brady Mitchelmore - 202112249

Part 2 - Stats and Attribute Comparison:

Part 2 Question 1: The process of knowledge discovery:

The knowledge discovery process is a crucial part of data mining. It involves several steps, starting with data preparation. This initial stage involves **cleaning**, **integrating**, **transforming**, and **selecting** the data to ensure it is usable and relevant. Misleading or irrelevant data can lead to inaccurate findings, so these steps are vital.

The next step is data mining itself. This involves applying mathematical and statistical methods to the prepared data to uncover interesting **patterns and relationships** within it.

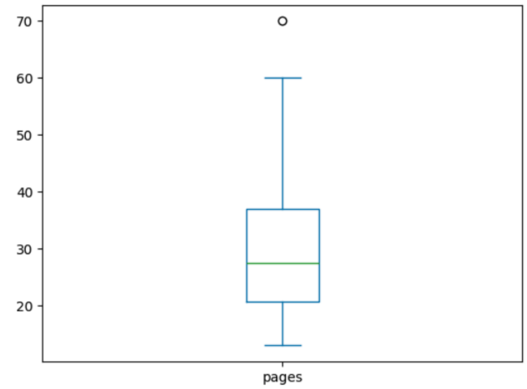
Once the data mining stage is complete, the patterns or models discovered are evaluated. This evaluation can involve a variety of techniques, but the goal is to assess the significance, usefulness, and validity of the findings.

The final stage of the knowledge discovery process is knowledge presentation. This involves presenting the findings in a clear, understandable format. This could be a written report, a visual representation of the data, or a combination of both. The aim is to communicate the findings effectively to those who need to use them.

Part 2 Question 2: Statistics of Data:

Question 2.1:

- The mean of the data is **31.79**, the median is **27.50**
- The mode of the data is **25.00**, since there is only one mode, the data is unimodal
- The midrange of the data is **41.50**
- The first quartile (Q1) is **20.75**, the third quartile is **37.00**
- Five-number summary:
    - **Minimum: 13.00**
    - **Q1: 20.75**
    - **Median: 27.50**
    - **Q3: 37.00**
    - **Maximum: 70.00**
- Boxplot:



Question 2.2:

- If the median is significantly different than the mean, we can infer that the dataset is **skewed**.
  - If the median is less than the mean, then the dataset is **positively** skewed.
  - If the median is greater than the mean, then the data is **negatively** skewed.

Question 2.3:

- A dataset with a variance of 0 implies that there is no spread between data points, this means that all items in the dataset would have to be equal to each other.

Part 2 Question 3: Attribute Types:

- **Car Model:** nominal
- **Sunroof:** binary symmetric
- **Is A Transformer:** binary non-symmetric
- **Condition:** ordinal
- **Engine Oil Temp In Celsius:** ratio scaled
- **Weight:** continuous
- **Owners:** discrete
- **Mileage:** ratio scaled

Part 2 Question 4: Comparing attributes:

Q4.1:

	parental education numeric	math score	reading score	writing score
parental education numeric	1.000000	0.159432	0.190908	0.236715
math score	0.159432	1.000000	0.817580	0.802642
reading score	0.190908	0.817580	1.000000	0.954598
writing score	0.236715	0.802642	0.954598	1.000000

- From the correlation matrix above, we can see that the parental education levels have a weak positive correlation with higher scores in all subjects. Another interesting observation here is that students who score high in one subject usually score higher in the other subjects as well.

Q4.2:

- The parental education levels have a weak positive correlation with higher scores for the math, reading, and writing scores

Q4.3:

- The writing score has the strongest correlation with the parents education levels with a correlation of **0.2367**

Part 2 Question 5: χ^2-Square Hypothesis Testing:

1. Δ_0 : Passenger survival dependent on passenger status
2. Contingency Table: $e_{ij} = \frac{count(a_i) \times count(b_j)}{n}$

status	Lived	Died	Total
Crew	212 (85.24)	673 (270.61)	885
1st Class	202 (29.83)	123 (18.16)	325
2nd Class	118 (15.28)	167 (21.62)	285
3rd Class	178 (57.10)	528 (169.36)	706
Total	710	1491	2201

3. $\chi^2 = \sum_i^n \sum_j^m \frac{(o_{ij} - e_{ij})^2}{e_{ij}}$

$$\chi^2 = \frac{(212-85.24)^2}{85.24} + \frac{(202-29.83)^2}{29.83} + \frac{(118-15.28)^2}{15.28} + \frac{(178-57.10)^2}{57.10} + \frac{(673-270.61)^2}{270.61} + \frac{(123-18.16)^2}{18.16} + \frac{(167-21.62)^2}{21.62} + \frac{(528-169.36)^2}{169.36} = 5069.4$$

We have a 4×2 table which gives us $(4-1) \times (2-1) = 3$ degrees of freedom.

The value from the χ^2 lookup table at significance 0.001 and df 3 is **34.528**

4. We computed the χ^2 test statistic to be **5069.4**, and the value from the lookup table is **34.528.** Since our calculated value is significantly larger than the lookup value, we can reject the null hypothesis and concluded that a passengers survival indeed (strongly) depends on the passengers status.

Part 3 - Distance Matrices and Data Normalization:

Part 3 Question 1: Distance Matrices:

Manhattan Distance Matrix:

- $d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \cdots + |x_{il} - x_{jl}|$

  (Anna, Bob): $|3-8| + |6-4| + |5-3| = 9$

  (Anna, Chuck): $|3-1| + |6-9| + |5-8| = 8$

  (Chuck, Bob): $|8-1| + |4-9| + |3-8| = 17$

	Anna	Bob	Chuck
Anna	0		
Bob	9	0	
Chuck	8	17	0

Euclidean Distance Matrix:

- $d(i,j) = \sqrt{|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \cdots + |x_{il} - x_{jl}|^2}$

  (Anna, Bob): $\sqrt{|3-8|^2 + |6-4|^2 + |5-3|^2} = 5.74$

  (Anna, Chuck): $\sqrt{|3-1|^2 + |6-9|^2 + |5-8|^2} = 4.69$

  (Chuck, Bob): $\sqrt{|8-1|^2 + |4-9|^2 + |3-8|^2} = 9.95$

	Anna	Bob	Chuck
Anna	0		
Bob	5.74	0	
Chuck	4.69	9.95	0

Part 3 Question 2: Calculating distance between data with mixed attribute types:

The data types for each attribute are as follows:

- **Name**: nominal
- **Age**: ratio scaled
- **Olympic Medalist**: binary non symmetric
- **Sex**: binary symmetric
- **Occupation**: nominal
- **Education Level:** ordinal

Since my student ID is **202112249** we will be comparing rows Dave (**4)** and Irene (**9)**

The distance between the 2 rows can be calculated as follows:

$$d(i,j) = \frac{\sum_{f=1}^{p} \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^{p} \delta_{ij}^{(f)}}$$

First lets get the $\delta_{ij}^{(f)}$ and $d_{ij}^{(f)}$ values.

For the ordinal Education attribute we have

$M_{Education} = 3$, where Bachelor's = 1, Master's = 2, and Doctorate = 3.

Then we have $r_{4Education} = 3$ and $r_{9Education} = 1$, which gives us

$z_{4Education} = 2/2 = 1$ and $z_{9Education} = 0/2 = 0$

$$\delta_{49}^{(Name)} = 1 \ , \ d_{49}^{(Name)} = 1$$
$$\delta_{49}^{(Sex)} = 1 \ , \ d_{49}^{(Sex)} = 1$$
$$\delta_{49}^{(Age)} = 1 \ , \ d_{49}^{(Age)} = \frac{|40 - 31|}{50 - 22} = 0.32$$
$$\delta_{49}^{(Occup.)} = 1 \ , \ d_{49}^{(Occup.)} = 1$$
$$\delta_{49}^{(Olymp.)} = 0 \ , \ d_{49}^{(Olymp.)} = 0$$
$$\delta_{49}^{(Education)} = 1 \ , \ d_{49}^{(Education)} = \frac{|1 - 0|}{3 - 1} = 0.50$$

Now that we have all needed values, we can calculate $d(i,j)$:

$$\begin{aligned} d(4,9) &= \frac{(1 \times 1) + (1 \times 1) + (1 \times 0.32) + (1 \times 1) + (1 \times 0) + (1 \times 0.50)}{(1) + (1) + (1) + (1) + (0) + (1)} \\ &= 0.76 \end{aligned}$$

Part 3 Question 3: Data Normalization:

1. **Min-max Normalization:**

   Age:

   - $min = 22$
   - $max = 35$

Index	Age
1	$\frac{25-22}{35-22}(1-0) + 0 = 0.23$
2	$\frac{30-22}{35-22}(1-0) + 0 = 0.62$

   Salary:

   - $min = 40,000$
   - $max = 80,000$

Index	Salary
1	$\frac{55,000-40,000}{80,000-40,000}(1-0) + 0 = 0.38$
2	$\frac{40,000-40,000}{80,000-40,000}(1-0) + 0 = 0.00$

2. **Z-score Normalization:**

   $\mu = \frac{35+28+40+30+25}{5} = 31.60$

$$\begin{aligned} \sigma^2 &= \frac{1}{N}\sum_{i=1}^{n}(x_i - \mu)^2 \\ &= \frac{1}{5}[(35-31.6)^2 + (28-31.6)^2 + (40-31.6)^2 + (30-31.6)^2 + (25-31.6)^2] \\ &= 28.24 \end{aligned}$$

$$\sigma = \sqrt{\sigma^2} = \sqrt{28.24} = 5.31$$

Index	Temperature
1	$\frac{35-31.6}{5.31} = 0.64$
2	$\frac{28-31.6}{5.31} = -0.68$

3. **Normalization by Decimal Scaling:**

   The salary values in the table have 5 decimal places which gives $j = 5$, this means we have to normalize by $10^5 = 100,000$

Index	Salary
1	$\frac{55,000}{100,000} = 0.55$
2	$\frac{40,000}{100,000} = 0.40$

4. **Summary:**

- When we know the min and max values and want to preserve the original distribution of the data, we should use **min-max normalization**
- When dealing with attributes having unknown future min and max, or when we have outliers, we should use **z-score normalization**
- When the range is not known, and is less affected by outliers, we should use **normalization by decimal scaling**