UDACITY

< Return to Classroom

# Investigate a Dataset

REVIEW

HISTORY

## Meets Specifications

## Congratulations 🎓

You have met all of the requirements for this project!

---

- You take a complex dataset and reveal some interesting, and unusual, patterns *(for example: "The bar chart proves that movie are watched more in summer and winter times. It makes sense because mostly people have school vacations and holidays during these time").*

Excellent work!

---

For data analysts, one of the most difficult parts of the job is explaining the implications of their studies to non-statisticians. With your choice of analysis and visualizations, and your data wrangling to generate them, you have highlighted that you have the skill to do this with ease. It is a skill that translates to any *data exploration*. You should look forward with confidence to applying what you have learned here, to any of the interesting data analyses that you will face in your career.

*(I have made some suggestions below. These suggestions don't detract from your work, they are just included so that you can add the finishing touches to your excellent report).*

Congrats again and best wishes for your next project!!

If you are interested in developing your skills in data analysis, this free text is a valuable resource (the link is to the authors github version of the text) *(**Note:** The first 4 chapters complement the contents of this course (they begin with concepts that you have learned, and then delve a little deeper to areas that could not be covered in this course). Later chapters involve more advanced areas of analysis - they will give you a good introduction to some of the directions that data analysis/science can take)*

## Code Functionality

- All code is functional and produces no errors when run.
- The code given is sufficient to reproduce the results described.

# This Section Meets Specifications

✓ CRITERIA: ALL CODE IS FUNCTIONAL (I.E. NO ERRORS ARE THROWN BY THE CODE). WARNINGS ARE OKAY, AS LONG AS THEY ARE NOT A RESULT OF POOR CODING PRACTICES.

The code in your notebook evaluates as expected (without errors). Nice work!


ADDITIONAL NOTES

- To help develop your skills using python, I highly recommend working through the examples in this free online text
- For guides for most aspects of the *Python Programming Language* this site is a great resource
- Finally, a guide to developing good coding habits in python

- The project uses NumPy arrays and Pandas Series and DataFrames where appropriate rather than Python lists and dictionaries.
- Where possible, vectorized operations and built-in functions are used instead of loops.

# This Section Meets Specifications

CRITERIA: THE PROJECT USES NUMPY ARRAYS AND PANDAS SERIES AND DATAFRAMES WHERE APPROPRIATE RATHER THAN PYTHON LISTS AND DICTIONARIES.

Where appropriate, your project uses and Pandas Series and DataFrames, rather than Python lists and dictionaries. Great work!

CRITERIA: WHERE POSSIBLE, VECTORIZED OPERATIONS AND BUILT-IN FUNCTIONS ARE USED INSTEAD OF LOOPS.

---

Pandas' *built-in functions*, which use vectorized operations, instead of loops, are used where possible. Great job!

---

IMPORTANT *(THESE TIPS WILL HELP YOU TO ANSWER YOUR QUESTIONS WITH LESS, AND MORE EFFICIENT, CODE)*

---

EXPLORE VECTORIZED OPERATIONS METHODS

- As you get more experience using *pandas*, you will find that, whatever task you want to achieve, there will be a *vectorized operation* that will allow you to achieve it.
- Use vectorized operations to explore `genres`, `cast`, `director`, or `production_companies`.

**The most informative variables in this dataset are string variables with substrings. This tip outlines how to incorporate those into your analysis.**

***Background:*** *As you know, the string variables included in the dataset are all composite observations (containing multiple values delimited by pipe characters (i.e.* `|` *). In the case of* `genres` *those entries are included in* ***alphabetical*** *order, for all of the others, the entries are included in order of* ***importance****. So, when you ask a question about the directors, or cast, etc, that ranks them (say by revenue or popularity), you have to make a decision about how to handle this unusual data.*

*One method is to split those strings.*

As you get more experience using *pandas*, you will find that, whatever task you want to achieve, there will be a *vectorized operation* that will allow you to achieve it.

When possible, it is always more computationally efficient to use *pandas* or *numpy* built-in operations over explicit for loops. The short reason is that they use vectorized methods, which attack a computational problem based on vectors by computing large chunks simultaneously.

For example, when analyzing the `genres`, you the can use in-built vectorized operation `.split()` along with the `.explode()` method. *See the* ***final*** *example* ***(Method Three)*** *in this post*

```
# split genres string, at |, explode the list to rows
genres_df = df.assign(genres=df['genres'].str.split('|')).explode('genres')
genres_df.head(10)
```

```
genres_df.head(10)
```

Same Movie

One row for each genre

| | id | popularity | budget | revenue | original_title | runtime | genres | vote_count | vote_average | release_year | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 135397 | 32.99 | 150000000 | 1513528810 | Jurassic World | 124 | Action | 5562 | 6.50 | 2015 | 1: |
| 0 | 135397 | 32.99 | 150000000 | 1513528810 | Jurassic World | 124 | Adventure | 5562 | 6.50 | 2015 | 1: |
| 0 | 135397 | 32.99 | 150000000 | 1513528810 | Jurassic World | 124 | Science Fiction | 5562 | 6.50 | 2015 | 1: |
| 0 | 135397 | 32.99 | 150000000 | 1513528810 | Jurassic World | 124 | Thriller | 5562 | 6.50 | 2015 | 1: |
| 1 | 76341 | 28.42 | 150000000 | 378436354 | Mad Max: Fury Road | 120 | Action | 6185 | 7.10 | 2015 | 1: |
| 1 | 76341 | 28.42 | 150000000 | 378436354 | Mad Max: Fury Road | 120 | Adventure | 6185 | 7.10 | 2015 | 1: |
| 1 | 76341 | 28.42 | 150000000 | 378436354 | Mad Max: Fury Road | 120 | Science Fiction | 6185 | 7.10 | 2015 | 1: |
| 1 | 76341 | 28.42 | 150000000 | 378436354 | Mad Max: Fury Road | 120 | Thriller | 6185 | 7.10 | 2015 | 1: |
| 2 | 262500 | 13.11 | 110000000 | 295238201 | Insurgent | 119 | Adventure | 2480 | 6.30 | 2015 | 1( |
| 2 | 262500 | 13.11 | 110000000 | 295238201 | Insurgent | 119 | Science Fiction | 2480 | 6.30 | 2015 | 1( |

*Click On Images To Enlarge Them*

*(If you are using Udacity's **workspace**, you will have to **update pandas** for this approach to work. If you are not familiar with how to do this, you can view this post and/or ask a mentor on **Knowledge** )*

As you can see it:

1. Splits the `genres` string *(into a list)*
2. It then ***"explodes"*** the list created in the first step into multiple rows
    - Each row, *for each individual movies*, will be **identical** *except* for the `genres` entry.

So, you will end up with:

1. Multiple rows for each movie, but
2. Each row represents only one genres.

**How is this helpful?**

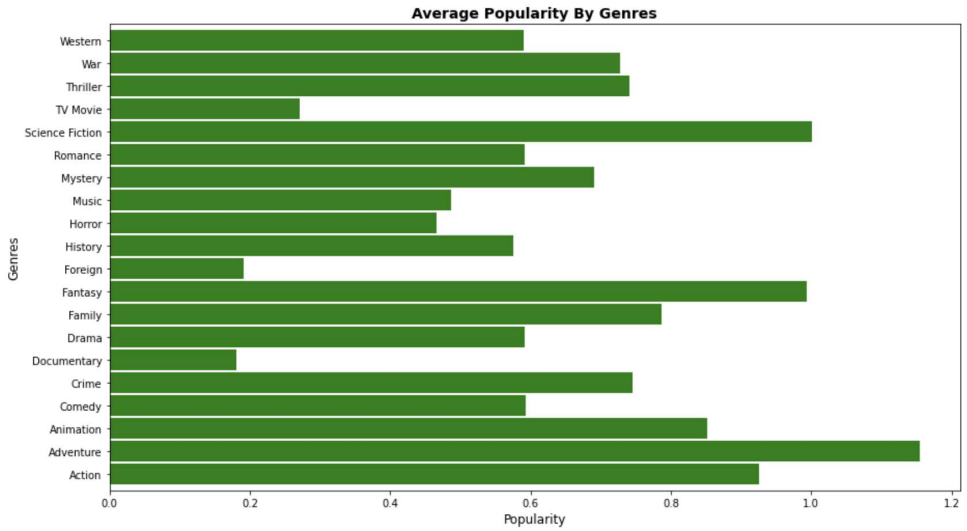
You can extend your analysis by using `.groupby()`. For example, to calculate the average popularity values, say:

```
# group exploded dataframe by genres, get average popularity
genres_df.groupby('genres').popularity.mean()
```

In [85]:
```
# 1. groupby genres, get average popularity (or any of the continuous variables) and plot
genres_df.groupby('genres').popularity.mean()
```

```
genres
Action              0.926136
Adventure           1.154259
Animation           0.852182
Comedy              0.592607
Crime               0.744821
Documentary         0.181432
Drama               0.591496
Family              0.786668
Fantasy             0.992840
Foreign             0.191496
History             0.575936
Horror              0.465357
Music               0.487321
Mystery             0.690012
Romance             0.592082
Science Fiction     1.001218
TV Movie            0.270896
Thriller            0.741513
War                 0.727683
Western             0.590615
Name: popularity, dtype: float64
```

In [86]:
```
# 2. groupby genres, get average popularity (or any of the continuous variables) and plot
genres_df.groupby('genres').popularity.mean().plot.barh(color=['g'],width=0.9,figsize=[14.70, 8.27],rot=0)

# titles/labels
addTitlesLabels('popularity','genres','Average Popularity by genres')
```




---

**Important: Given the unusual structure of the dataframe created, it can only be used for the analysis of `genres` *(which is why it is given a different name from the original dataframe, which you want to preserve for other analysis)*. If you use this dataframe to analyze features of the data that don't relate to `genres` then you will get biased results (because there are now multiple rows per movie).**

---

*(Obviously, approaches like this are something that you will pick up over time - hopefully you find this useful/interesting).*

---


ADDITIONAL NOTES

---

The popularity of *Pandas* in data analysis is a result of two features: It has a simple but flexible data representation; and It provides a toolbox of in-built methods that allows you to easily analyze your data (which are efficient and easy to use).

- Understanding Vectorization in NumPy and Pandas
- How to Speed Up Pandas Data Operations Using Vectorized Operations

- **The code makes use of at least 1 function to avoid repetitive code.**
- **The code contains good comments and meaningful variable names, making it easy to read.**

# This Section Meets Specifications

✓ THE CODE MAKES USE OF AT LEAST 1 FUNCTION TO AVOID REPETITIVE CODE.

✓ THE CODE CONTAINS GOOD COMMENTS AND MEANINGFUL VARIABLE NAMES, MAKING IT EASY TO READ.

- Your code is easy to follow, thanks to the code comments added and variable names used. Very nice work!
- Excellent work using a wrangling function to avoid repetitive coding. Kudos!
    - Always remember to add a docstring (link to examples) to explain the purpose of your function.

In [81]:
```
#Create new funtion to separate each person in each movie.
def get_data(column_name):
    '''
    Split the column values by '|' and stack them into a single Series
    docstring: IMPORTANT explain function here (not completed here)
    inputs: ...
    output: ...
    '''
    all_data = movies[column_name].str.split('|', expand=True).stack()

    # Count the occurrences of each value in descending order
    count = all_data.value_counts(ascending=False)

    return count
```

In [82]:
```
help(get_data)
```

```
Help on function get_data in module __main__:

get_data(column_name)
    Split the column values by '|' and stack them into a single Series
    docstring: IMPORTANT explain function here (not completed here)
    inputs: ...
    output: ...
```


ADDITIONAL NOTES

- In data analysis, in a work environment, commenting code, so that your colleagues understand the intent of the code that you write, is a basic necessity.
    - It only takes a few minutes to clearly comment code like this but the benefits are substantial. Overtime you will built up a library of scripts/notebook that you will refer back to. With clearly commented code, you can scan those resources quickly.
    - Commenting code is a good habit to develop, because it communicates to colleagues, or to yourself at some future time, the intent of the code that you have written (in a succinct way that avoids you having to examine the code line by line). It is difficult to overstate the importance of clear code commenting, in a work environment, in data analysis.
    - Here is a summary of good commenting etiquette
    - Here is a more detailed discussion, including a guide to commenting and a response to the idea that you do not need to comment code

Again, nice work!!

- If you want to explore functions in python:
    - An *Udacity* guide to functions, with examples
    - Another introduction to functions, with examples
    - This free online text/tutorial provides a *fun* and ***practical*** approach to help you to develop this area of your coding skills
    - Here is a simple overview of functions

## Quality of Analysis

**The project clearly states one or more questions, then addresses those questions in the rest of the analysis.**

# This Section Meets Specifications

✓ CRITERIA: THE PROJECT CLEARLY STATES ONE OR MORE QUESTIONS, THEN ADDRESSES THOSE QUESTIONS IN THE REST OF THE ANALYSIS.

---

You list **six** questions:

1. What the best movies??
2. Which movies have the highest profit, and least budget?
3. How does popularity affect the profit?
4. Which years do movies made the maximum profits?
5. What are the top casts, directors?
6. Which months have higher profits??

at the start of your analysis and develop your analysis around those questions throughout your script. Very nice work!

---

IMPORTANT

---

- Using *Markdown* syntax leads to report that is much easier to read. Here is a *"cheatsheet"* for basic markdown syntax (which shows the result of using that syntax)
- Here is a guide to adding commentary to a *Jupyter Notebook*

*For example, there should be a **space** after each markdown character:*

MARKDOWN

```
3.Study the stats to find meaningful information. <br>

Questions: <br>
1. What the best movies??
2. Which movies have the highest profit, and least budget?
3.How does popularity affect the profit?
4.Which years do movies made the maximum profits?
5.What are the top casts, directors?
6.Which months have higher profits??
```

```
Questions: <br>
1.  What the best movies??
2.  Which movies have the highest profit, and least budget?
3. How does popularity affect the profit?
4. Which years do movies made the maximum profits?
5. What are the top casts, directors?
6. Which months have higher profits??
```

```
In [40]: #import libraries to this project
```

EVALUATED

Questions:

1. What the best movies??
2. Which movies have the highest profit, and least budget? 3.How does popularity affect the profit? 4.Which years do movies made the maximum profits? 5.What are the top casts, directors? 6.Which months have higher profits??

Questions:

1. What the best movies??
2. Which movies have the highest profit, and least budget?
3. How does popularity affect the profit?
4. Which years do movies made the maximum profits?
5. What are the top casts, directors?
6. Which months have higher profits??

*which is easier to read?*

---


ADDITIONAL NOTES

---

Research questions are an important element of the research process.

By defining exactly what you, as a researcher, are interested in answering, these questions influence most of the rest of the steps that you take to conduct the research.

Importantly, they also inform the reader, giving them a *road map* for the direction that your report will take.

- Research question: the importance of your research question
- Developing research questions

Please take the time to read them; I hope you learn something new from these resources."

## Data Wrangling Phase

**The project documents any changes that were made to clean the data, such as merging multiple files, handling missing values, etc.**

# This Section Meets Specifications

CRITERIA: THE PROJECT DOCUMENTS ANY CHANGES THAT WERE MADE TO CLEAN THE DATA, SUCH AS MERGING MULTIPLE FILES, HANDLING MISSING VALUES, ETC.

---

You have wrangled, and documenting the wrangling of your dataset. Your code

1. Identified missing/erroneous and duplicated data
2. Identified data that requires wrangling to be useful in your analysis and/or would bias your analysis

Then you resolved both issue to result in a cleaned dataset ready for analysis. Excellent work!

---

IMPORTANT

---

MISSING/ERRONEOUS DATA

While wrangling the data, also look out for unusual values. For example, in your final dataframe, at least 50% of movies had revenue and budget values that were zero dollars.

## 3 Q2: Which movies have the highest profit, and least budget?

[64]:
```
movies2.describe()
```

t[64]:

| | popularity | budget | revenue | runtime | release_year | budget_adj | revenue_adj | profit | profit_adj |
|---|---|---|---|---|---|---|---|---|---|
| count | 10730.000000 | 1.073000e+04 | 1.073000e+04 | 10730.000000 | 10730.000000 | 1.073000e+04 | 1.073000e+04 | 1.073000e+04 | 1.073000e+04 |
| mean | 0.280242 | 1.480223e+07 | 4.032356e+07 | 102.469804 | 2001.258807 | 1.776416e+07 | 5.201099e+07 | 2.552133e+07 | 3.424682e+07 |
| std | 0.959246 | 3.106566e+07 | 1.176573e+08 | 30.495126 | 12.820470 | 3.446771e+07 | 1.454311e+08 | 9.715381e+07 | 1.259456e+08 |
| min | 0.000000 | 0.000000e+00 | 0.000000e+00 | 0.000000 | 1960.000000 | 0.000000e+00 | 0.000000e+00 | -4.139124e+08 | -4.139124e+08 |
| 25% | 0.000000 | 0.000000e+00 | 0.000000e+00 | 90.000000 | 1995.000000 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 |
| 50% | 0.000000 | 0.000000e+00 | 0.000000e+00 | 99.000000 | 2006.000000 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 |
| 75% | 0.000000 | 1.600000e+07 | 2.500000e+07 | 112.000000 | 2011.000000 | 2.110357e+07 | 3.470565e+07 | 9.985547e+06 | 1.384643e+07 |
| max | 32.000000 | 4.250000e+08 | 2.781506e+09 | 900.000000 | 2015.000000 | 4.250000e+08 | 2.827124e+09 | 2.544506e+09 | 2.750137e+09 |

[65]:
```
# create a copy, to avoid interferring with your code
df1=movies2.copy()
# columns with zero values (which must be missing values)
zero_cols=[ 'budget', 'revenue', 'runtime','budget_adj', 'revenue_adj']

# replace zeros with np.nan (pandas will ignore nan)
# .apply(), by default, works on rows. Use axis=1 for columns
df1[zero_cols]=df1[zero_cols].apply(lambda cols: cols.replace(0,np.nan),axis=1)

df1.describe()
```

t[65]:

| | popularity | budget | revenue | runtime | release_year | budget_adj | revenue_adj | profit | profit_adj |
|---|---|---|---|---|---|---|---|---|---|
| count | 10730.000000 | 5.152000e+03 | 4.842000e+03 | 10702.000000 | 10730.000000 | 5.152000e+03 | 4.842000e+03 | 1.073000e+04 | 1.073000e+04 |
| mean | 0.280242 | 3.082840e+07 | 8.935806e+07 | 102.737899 | 2001.258807 | 3.699718e+07 | 1.152577e+08 | 2.552133e+07 | 3.424682e+07 |
| std | 0.959246 | 3.893577e+07 | 1.621663e+08 | 30.080557 | 12.820470 | 4.198598e+07 | 1.989561e+08 | 9.715381e+07 | 1.259456e+08 |
| min | 0.000000 | 1.000000e+00 | 2.000000e+00 | 3.000000 | 1960.000000 | 9.210911e-01 | 2.000000e+00 | -4.139124e+08 | -4.139124e+08 |
| 25% | 0.000000 | 6.000000e+06 | 7.791448e+06 | 90.000000 | 1995.000000 | 8.140009e+06 | 1.048753e+07 | 0.000000e+00 | 0.000000e+00 |
| 50% | 0.000000 | 1.750000e+07 | 3.191220e+07 | 99.000000 | 2006.000000 | 2.287867e+07 | 4.403217e+07 | 0.000000e+00 | 0.000000e+00 |
| 75% | 0.000000 | 4.000000e+07 | 1.000000e+08 | 112.000000 | 2011.000000 | 5.024535e+07 | 1.318074e+08 | 9.985547e+06 | 1.384643e+07 |
| max | 32.000000 | 4.250000e+08 | 2.781506e+09 | 900.000000 | 2015.000000 | 4.250000e+08 | 2.827124e+09 | 2.544506e+09 | 2.750137e+09 |

[66]:
```
# Define new function to find the max/min of movies
def min max data(column name):
```

*Click On Images To Enlarge Them*

It may be conceivable for revenue to be zero (although it is highly unlikely) but for budgets that is not possible. You typically would make an explicit choice about how to deal with this anomaly *(given that the most reasonable explanation is that they represent missing data then, for any analysis involving those variables, those observations should be changed to* ***np.nan*** *(which pandas will ignore and not include in any analysis))*.

---

INFLATION ADJUSTED VS CURRENT DOLLAR VALUES

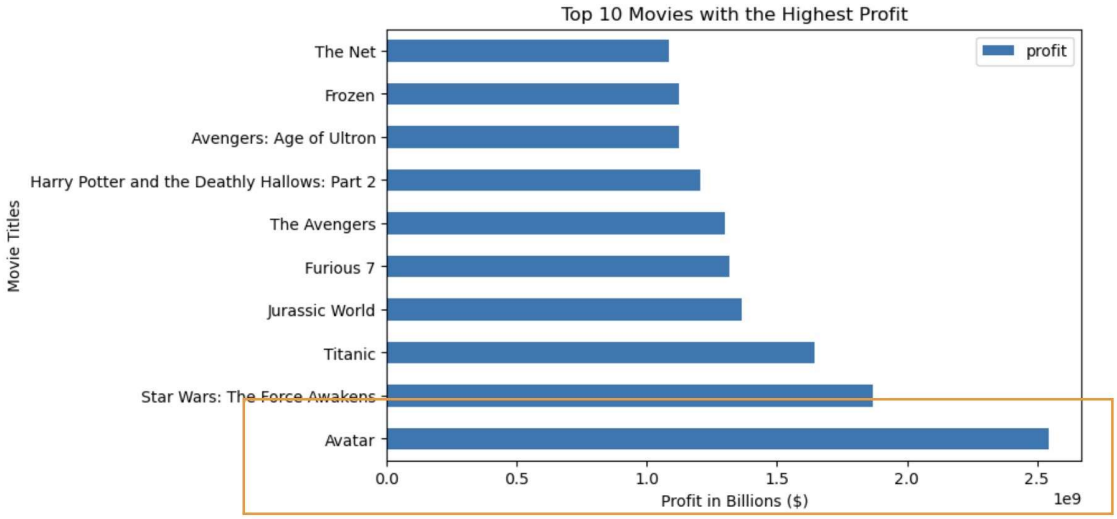
Excellent work presenting an analysis of the relationship between *revenue* and *budget*.

However, the *revenue* and *budget* values used are the dollar values for each year. These values are not directly comparable over time *(a single dollar in 1960 would buy you a lot more than it would today)*.

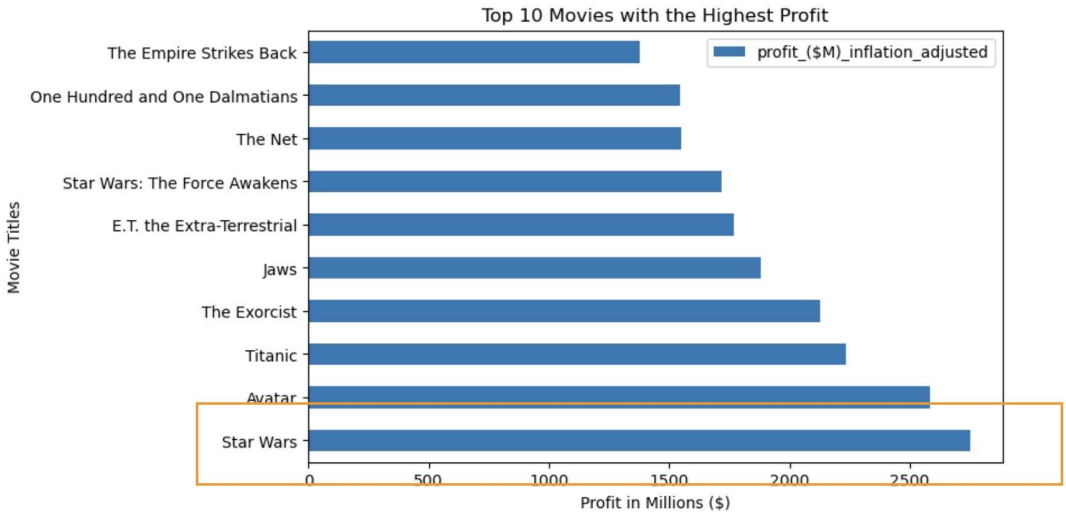
For this reason, an *adjusted* version of *revenue* and *budget* are included (the variables with `_adj` in their names). These variables account for changes in the *cost of living* (also known as *purchasing power parity adjustments*) - so that, for those variables, a dollar in 1960 is comparable to a dollar value today.

It would be interesting to compare the analysis using both versions of the data for *revenue* and *budget*.

[5]:
```
# Create a bar graph for the top 10 most profitable movies
# First sort the raw data by the profit.
top_10_profitable_movies = movies2.nlargest(10, 'profit')
# Create title and profit lists which will be used as X-axis and Y-axis values in bar graph.
ax = top_10_profitable_movies.plot(kind='barh', x='Title', y='profit', figsize=(8, 5))
plt.title('Top 10 Movies with the Highest Profit')
plt.ylabel('Movie Titles')
plt.xlabel('Profit in Billions ($)')
plt.show()
```




[6]:
```
# Create a bar graph for the top 10 most profitable movies
# First sort the raw data by the profit.
top_10_profitable_movies = movies2.nlargest(10, 'profit_(\$M)_inflation_adjusted')
# Create title and profit lists which will be used as X-axis and Y-axis values in bar graph.
ax = top_10_profitable_movies.plot(kind='barh', x='Title', y='profit_(\$M)_inflation_adjusted', figsize=(8, 5))
plt.title('Top 10 Movies with the Highest Profit')
plt.ylabel('Movie Titles')
plt.xlabel('Profit in Millions ($)')
plt.show()
```




---

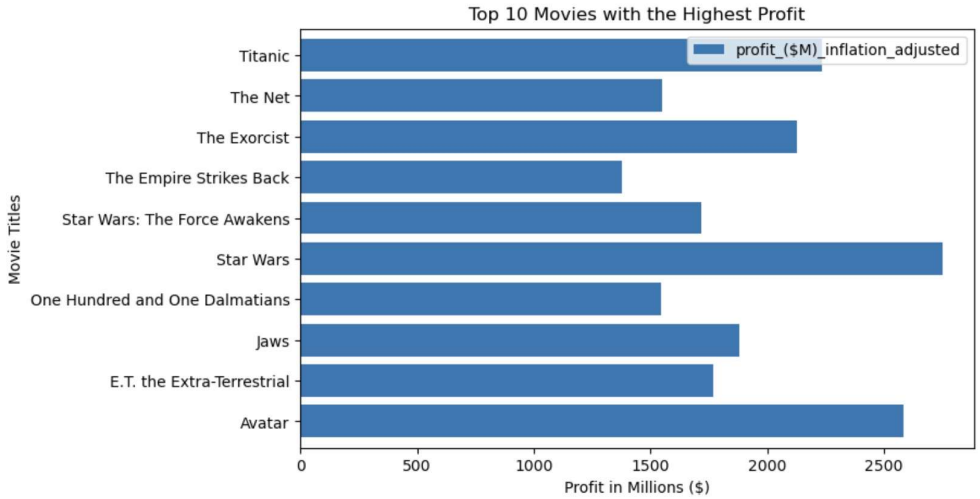
DATA UNITS

If the units of the data result in large numbers in visualizations, the units should be changed - to make the visualizations easier to interpret.

For example, change the financial information to Millions of dollars, instead of dollars:

```
0]: # convert financial data to Millions
movies2['revenue_(\$M)']=movies2['revenue'].div(10**6)
movies2['budget_(\$M)']=movies2['budget'].div(10**6)
movies2['revenue_(\$M)_inflation_adjusted']=movies2['revenue_adj'].div(10**6)
movies2['budget_(\$M)_inflation_adjusted']=movies2['budget_adj'].div(10**6)

# calculate profit
movies2['profit_(\$M)']=movies2['revenue_(\$M)']-movies2['budget_(\$M)']
movies2['profit_(\$M)_inflation_adjusted']=movies2['revenue_(\$M)_inflation_adjusted']-movies2['budget_(\$M)_inflation_adjusted']
```

```
3]: # Create a bar graph for the top 10 most profitable movies
# First sort the raw data by the profit.
top_10_profitable_movies = movies2.nlargest(10, 'profit_(\$M)_inflation_adjusted').sort_values(by='Title')
# Create title and profit lists which will be used as X-axis and Y-axis values in bar graph.
ax = top_10_profitable_movies.plot(kind='barh', x='Title', y='profit_(\$M)_inflation_adjusted', figsize=(8, 5), width=0.8)
plt.title('Top 10 Movies with the Highest Profit')
plt.ylabel('Movie Titles')
plt.xlabel('Profit in Millions ($)')
plt.show()
```




*Click On Images To Enlarge Them*

---


ADDITIONAL NOTES

---

Any insights that you generate from your analysis are only as valid as the data that you use is.

This is why data cleaning/wrangling is a fundamental feature of any data analysis.

Data wrangling seeks to ensure that your results are valid by ensuring that data is in a reliable state before it's analyzed

- Data Wrangling in Python
- Data Wrangling With Pandas

## Exploration Phase

- The project investigates the stated question(s) from multiple angles.
- The project explores at least three variables in relation to the primary question. This can be an exploratory relationship between three variables of interest, or looking at how two independent variables relate to a single dependent variable of interest.
- The project performs both single-variable (1d) and multiple-variable (2d) explorations.

# This Section Meets Specifications

✓ CRITERIA: THE PROJECT INVESTIGATES THE STATED QUESTION(S) FROM MULTIPLE ANGLES.

✓ CRITERIA: THE PROJECT EXPLORES AT LEAST THREE VARIABLES IN RELATION TO THE PRIMARY QUESTION. THIS CAN BE AN EXPLORATORY RELATIONSHIP BETWEEN THREE VARIABLES OF INTEREST, OR LOOKING AT HOW TWO INDEPENDENT VARIABLES RELATE TO A SINGLE DEPENDENT VARIABLE OF INTEREST.

✓ CRITERIA: THE PROJECT PERFORMS BOTH SINGLE-VARIABLE (1D) AND MULTIPLE-VARIABLE (2D) EXPLORATIONS.

---

You have investigated your questions from multiple angles, using 1d and 2d explorations.

Appropriate techniques were used to explore at least three variables that helped answer your research questions. Excellent work!

---


ADDITIONAL NOTES

---

Investigating a dataset, i.e. performing an exploratory data analysis, is an approach to analysing data to summarize their main characteristics, using visual and statistical methods.

Data explorations are typically hierarchical, where the properties of the individual variables are examined first, followed by an analysis of the relationships between the variables.

Data explorations are less structured than model building. The idea of a *data exploration* is that, before you begin, you note down some thoughts that you have about the data that you are going to explore - some immediate questions that pop into your mind. Then you use that as a base for creating visualizations - to answer those questions. Often, those visualizations will lead to other questions. So that the data exploration becomes like a *""stream of consciousness""* of questions and answers. This creates a broad picture of important trends and relationships.

- Data Exploration and Analysis Using Python
- Exploratory Analysis Using Univariate, Bivariate, and Multivariate Analysis Techniques

- The project's visualizations are varied and show multiple comparisons and trends.

- At least two kinds of plots should be created as part of the explorations.
- Relevant statistics are computed throughout the analysis when an inference is made about the data.

# This Section Meets Specifications

✓ CRITERIA: THE PROJECT'S VISUALIZATIONS ARE VARIED AND SHOW MULTIPLE COMPARISONS AND TRENDS.

✓ CRITERIA: AT LEAST TWO KINDS OF PLOTS SHOULD BE CREATED AS PART OF THE EXPLORATIONS.

✓ CRITERIA: RELEVANT STATISTICS ARE COMPUTED THROUGHOUT THE ANALYSIS WHEN AN INFERENCE IS MADE ABOUT THE DATA.

---

Your selection of visualizations highlight interesting patterns, and you perform a range of statistical analyses. Both of which help to answer your research questions. Excellent work!

---

 TIPS

---

PLOT DIMENSIONS

The plot dimensions change from plot to plot in your *data exploration*.

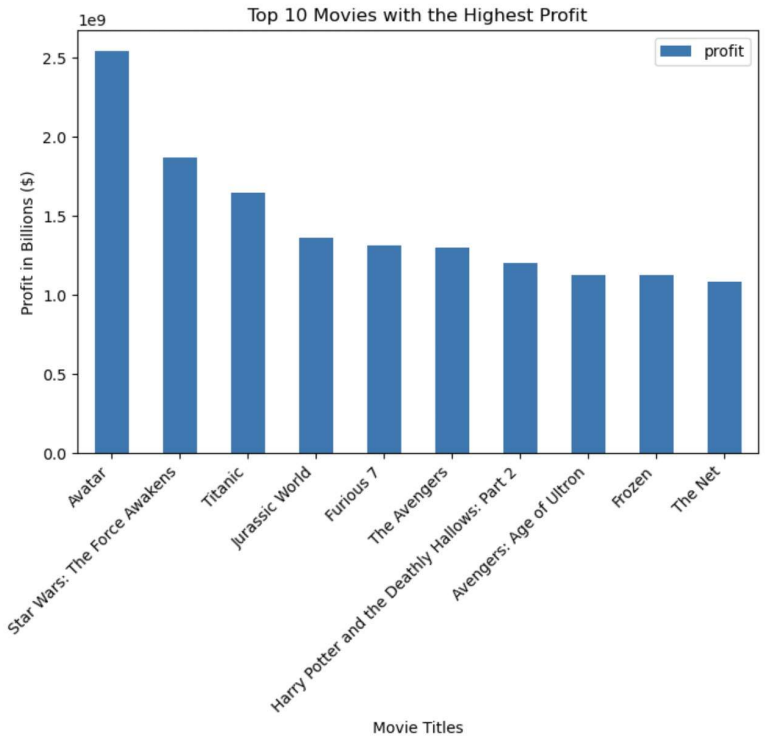
Using a fixed plot dimension, that is large enough to show all of the relevant information in the plot, for all of your visualizations helps (so that the reader doesn't have to re-focus on each visualization (and the increased width means that you don't have to angle some of your tick labels)).

---

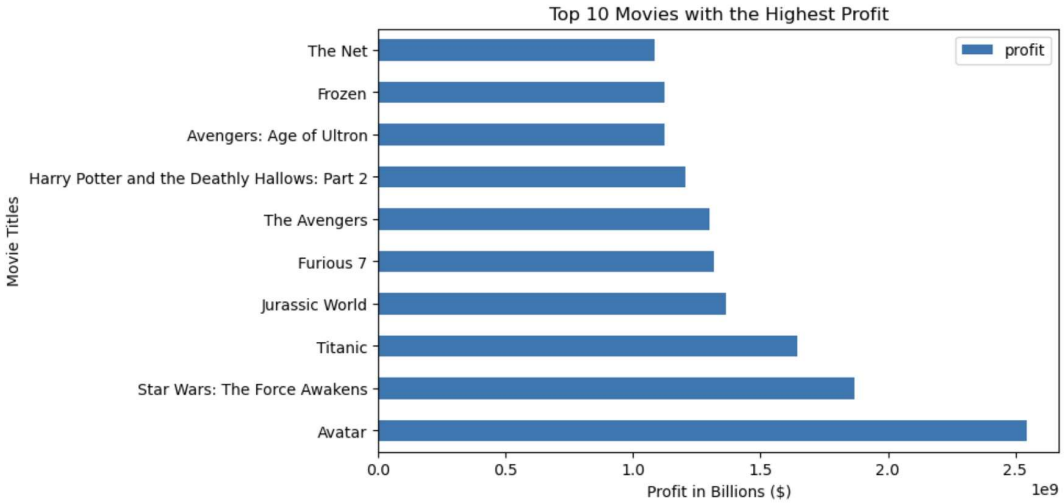
TICK LABELS

Avoid angling tick labels. It makes visualizations difficult to interpret *(Imagine the viewer/reader angling their heads at the same time as trying to take in the rest of the information in the visualizations)*. There are two ways to resolve this:

1. Change the plot dimensions (in many cases, the tick labels in your visualizations did not need to be angled after an appropriate plot size was set),
2. Change the orientation of the visualization.




57]:
```
# Create a bar graph for the top 10 most profitable movies
# First sort the raw data by the profit.
top_10_profitable_movies = movies2.nlargest(10, 'profit')
# Create title and profit lists which will be used as X-axis and Y-axis values in bar graph.
ax = top_10_profitable_movies.plot(kind='barh', x='Title', y='profit', figsize=(8, 5))
plt.title('Top 10 Movies with the Highest Profit')
plt.ylabel('Movie Titles')
plt.xlabel('Profit in Billions ($)')
plt.show()
```

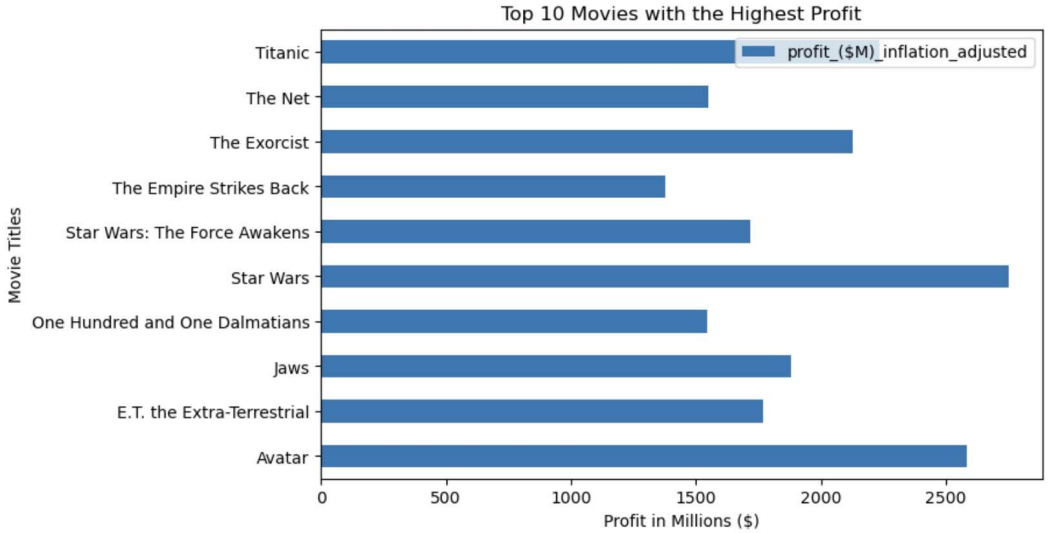



---

ORDINAL VARIABLES

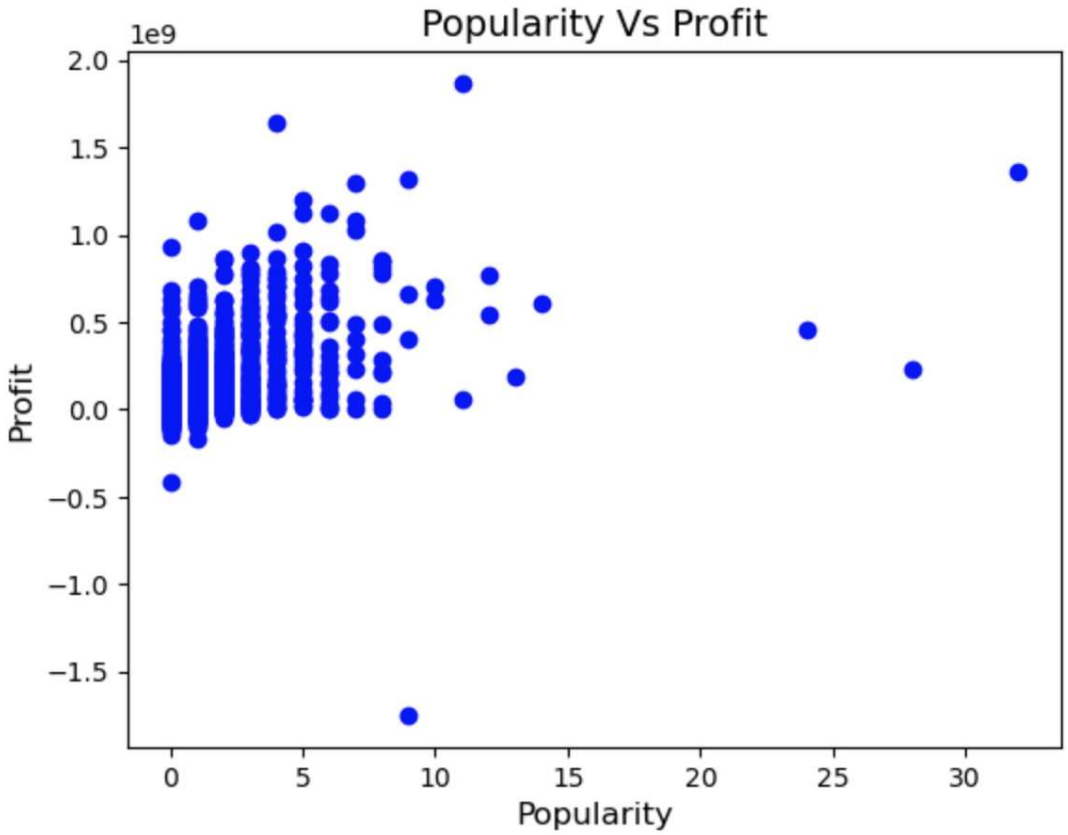
Also, variables with string categories should be plotted in alphabetical order *(that is, it is easier to determine the difference in the height of bars (which is the only benefit of ordering by value) than it is to search an unordered list of strings for a word)*

```
9]: # Create a bar graph for the top 10 most profitable movies
    # First sort the raw data by the profit.
    top_10_profitable_movies = movies2.nlargest(10, 'profit_(\$M)_inflation_adjusted').sort_values(by='Title')
    # Create title and profit lists which will be used as X-axis and Y-axis values in bar graph.
    ax = top_10_profitable_movies.plot(kind='barh', x='Title', y='profit_(\$M)_inflation_adjusted', figsize=(8, 5))
    plt.title('Top 10 Movies with the Highest Profit')
    plt.ylabel('Movie Titles')
    plt.xlabel('Profit in Millions ($)')
    plt.show()
```




JITTER

Typically it is not a good idea to use scatterplots with discrete data.




This is because the points on the plot will be ***overplotted*** (link) (i.e. points in the visualizations plotting on top of each other). That is, because there are only a finite number of values that *X* can take, all of the points will lie

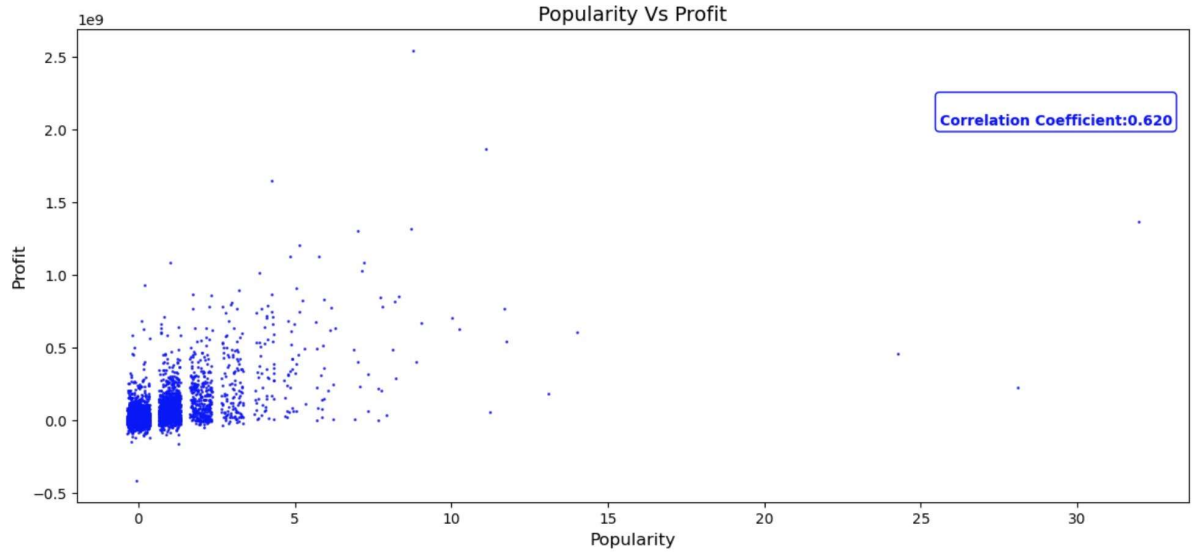
along those lines. ***Which means, that it is not possible to interpret the information in this plot.***

If you do decide to use scatterplot, you should use `jitter`. _

- **Standard scatterplot:** Seaborn's_ `.stripplot()` allows you to use `jitter`.
- **Linear regression:** `.regplot()` or `.lmplot()` have *jitter* options *(where you can specify either `x` or `y` or both).*

```
# Set the figure size
plt.figure(figsize=(14, 6))

import seaborn as sb

# Create a scatter plot to study the relationship between popularity and profit
sb.regplot(data=movies2, x='popularity', y='profit', color='blue',x_jitter=0.35, scatter_kws={'s':1},fit_reg=False )

# Calculate the correlation coefficient
correlation_coefficient = movies2['popularity'].corr(movies2['profit'])

plt.annotate(f'\nCorrelation Coefficient:{correlation_coefficient:0.3f}',
             (movies2['popularity'].max()*0.8,movies2['profit'].max()*0.8 ),
              color='blue', weight='bold',
              bbox=dict(facecolor='none', edgecolor='blue', boxstyle='round'))
# Setup the title and labels of the scatter plot
plt.title(f"Popularity Vs Profit", fontsize=14)
plt.xlabel("Popularity", fontsize=12)
plt.ylabel("Profit", fontsize=12)


# Show the scatter plot
plt.show()
```

**Annotate the plot, instead of adding a print statement**




*Click On Images To Enlarge Them*

*(where the `jitter` value is the maximum value that the points can be jittered around the original position, and the `dodge` option separates the categories. That is, for that plot, add the option `jitter = 0.35` (**Seaborn** allocates a space of `1` for each discrete value, jittering by `0.35` randomly moves the points around their original location **(both sides)**, so that is a space of `0.7` the remaining `0.3` allows enough room between categories so that you have a space between each).).*

- Here is a discussion of using jitter

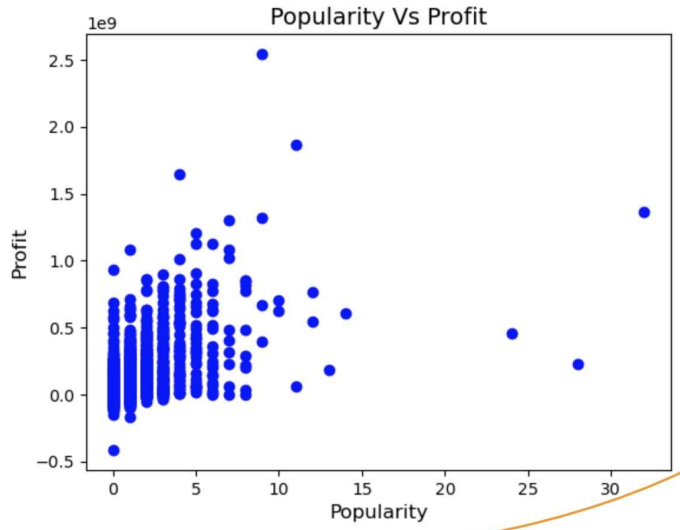
---

Also, the `plt.figure()` statement has to be placed **before** the plot statement:

```
30]: # Create a scatter plot to study the relationship between popularity and profit
     plt.scatter(movies2['popularity'], movies2['profit'], color='blue')

     # Setup the title and labels of the scatter plot
     plt.title("Popularity Vs Profit", fontsize=14)
     plt.xlabel("Popularity", fontsize=12)
     plt.ylabel("Profit", fontsize=12)

     # Set the figure size
     plt.figure(figsize=(5, 3))

     # Calculate the correlation coefficient
     correlation_coefficient = movies2['popularity'].corr(movies2['profit'])

     # Print the correlation coefficient
     print("Correlation Coefficient:", correlation_coefficient)

     # Show the scatter plot
     plt.show()
```

This statement creates the figure, it has to be before the plot statement (if not:
i. It does not change the plot size
ii. It only prints a statement below the plot)

```
Correlation Coefficient: 0.6200145991888912
```




```
<Figure size 500x300 with 0 Axes>
```

**4.0.0.1 From the chart above, we can noticed that there is a positive relationship between popularity and profit. The lower correlation coefficient value is less desirable and is probabily because of movies with high budget and low profit and vv.**


ADDITIONAL NOTES

Visual and statistical methods play two roles in data explorations:

1. They help uncover the properties of the data, and discover any relationships between variables.
2. They also are invaluable in *communicating* those result to colleagues and readers.

*Python*, and its modules, like *pandas* and *Seaborn*, are powerful tools that allow you to achieve these goals. *Obviously, the wide range of possible approaches are something that you will pick up over time - hopefully you found your introduction to these tools useful, interesting and productive.*

- The Python Graph Gallery

    - This site contains a gallery of most types of visualizations, *each example contains code that you can use as a template for your visualizations.*
- Plotting with categorical data

## Conclusions Phase

- **The Conclusions have reflected on the steps taken during the data exploration.**
- **The Conclusions have summarized the main findings in relation to the question(s) provided at the beginning of the analysis accurately.**
- **The project has pointed out where additional research can be done or where additional information could be useful.**
- **The conclusion should have at least 1 limitation explained clearly.**
- **The analysis does not state or imply that one change causes another based solely on a correlation.**

# This Section Meets Specifications

✓ CRITERIA: THE CONCLUSIONS HAVE REFLECTED ON THE STEPS TAKEN DURING THE DATA EXPLORATION.

✓ CRITERIA: THE CONCLUSIONS HAVE SUMMARIZED THE MAIN FINDINGS IN RELATION TO THE QUESTION(S) PROVIDED AT THE BEGINNING OF THE ANALYSIS ACCURATELY.

✓ CRITERIA: THE PROJECT HAS POINTED OUT WHERE ADDITIONAL RESEARCH CAN BE DONE OR WHERE ADDITIONAL INFORMATION COULD BE USEFUL.

✓ CRITERIA: THE CONCLUSION SHOULD HAVE AT LEAST 1 LIMITATION EXPLAINED CLEARLY.

✓ CRITERIA: THE ANALYSIS DOES NOT STATE OR IMPLY THAT ONE CHANGE CAUSES ANOTHER BASED SOLELY ON A CORRELATION.

---

Your report includes a conclusion section, summarizing the results of your analysis.

Throughout your analysis, you highlight the limitations of making direct causal statements (and the limitations due to the nature of features of the dataset). *You also include a discussion of limitations in your conclusion.* Excellent work!

---


ADDITIONAL NOTES

---

You will write your *Conclusion* (and *Introduction*) after you have completed your analysis.

The conclusion is typically the first part of a report that is read - even though it occupies a space at the end. Reading a conclusion first is the most efficient, and effective, way to read a report. It helps the reader organize their thoughts around a *'big picture'* perspective of the subject.

This is why it is important to make your *'last words count'* (because they will be the first words read).

- Conclusions
- Write Discussion and Conclusion

## Communication

- **The code should have ideally the following sections: Introduction; Questions; Data Wrangling; Exploratory Data Analysis; Conclusions, Limitation.**
- **Reasoning is provided for each analysis decision, plot, and statistical summary.**
- **Interpretation of plots and application of statistical tests should be correct and without error.**
- **Comments are used within the code cells.**
- **Documented the flow of analysis in the mark-down cells.**

# This Section Meets Specifications

✓ CRITERIA: THE CODE SHOULD HAVE IDEALLY THE FOLLOWING SECTIONS: INTRODUCTION; QUESTIONS; DATA WRANGLING; EXPLORATORY DATA ANALYSIS; CONCLUSIONS, LIMITATION.

✓ CRITERIA: REASONING IS PROVIDED FOR EACH ANALYSIS DECISION, PLOT, AND STATISTICAL SUMMARY.

✓ CRITERIA: INTERPRETATION OF PLOTS AND APPLICATION OF STATISTICAL TESTS SHOULD BE CORRECT AND WITHOUT ERROR.

✓ CRITERIA: COMMENTS ARE USED WITHIN THE CODE CELLS.

✓ CRITERIA: DOCUMENTED THE FLOW OF ANALYSIS IN THE MARK-DOWN CELLS.

---

Your commentary is clear and concise.

You have shown some very interesting results! Excellent job!

However, the discussion of those results is quite limited.

The commentary on your results is very important. It helps to clarify what you feel your visualizations and statistics are showing the reader. As this overview highlights

---

TIPS

- The commentary included should inform the reader of your thinking during your data exploration. A key part of that process is the *Introduction*, i.e. what interests you about this dataset. Then, each stage of the *exploration* should be outlined. That is:

    - Why (the visualization/statistic was created), and
    - What (patterns in the data the visualization/statistic revealed)

    Currently, the commentary includes one or two facts about the visualization.

    **The commentary doesn't have to be technical (or long) but it has to explain to the reader the role of the visualization in your exploration.**

- **FYI**: Typically, when working on data exploration projects, in a work environment, you have more background knowledge of the data - which informs your commentary (i.e. allows you to tell a story about the role of the variables that you are analyzing.)

---


ADDITIONAL NOTES

---

The commentary on your results is very important. It helps to clarify what you feel your visualizations and statistics are showing the reader. As this overview highlights:

> 'However, tables and figures do not simply speak for themselves but you have to communicate to your readers what e.g. numbers and quantities mean; you also want your readers to accept your conclusions, which should follow logically from your results. If you do not comment on the results, your reader might interpret them differently, and your conclusions might appear rather strange or surprising.'

- How to comment on statistics
- Writing The Results

**Visualizations made in the project depict the data in an appropriate manner (i.e., has appropriate labels, scale, legends, and plot type) that allows plots to be readily interpreted.**

# This Section Meets Specifications

✓ CRITERIA: VISUALIZATIONS MADE IN THE PROJECT DEPICT THE DATA IN AN APPROPRIATE MANNER (I.E., HAS APPROPRIATE LABELS, SCALE, LEGENDS, AND PLOT TYPE) THAT ALLOWS PLOTS TO BE READILY INTERPRETED.

---

All plots are clearly titled, and axes are labeled making your visualizations easy to interpret, excellent work!


ADDITIONAL NOTES

Meaningful titles and labels help your readers to interpret, and understand, the visualizations that you are using to communicate your results.

Titles and labels also help readers recall the message. The title is the most important, where specific, informative, titles reinforce the visualization's main message.

- Importance of Titles and Labels
- How to add labels and titles to a plot in matplotlib

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

START