https://colab.research.google.com/drive/1OaR3jG3LWm1CgjqGoUTQIVeVFzHj19_D?usp=sharing

https://colab.research.google.com/drive/1bJ4v34qB2QdnHxIFiMSM9cxBe-DRtOEN?usp=sharing

In [4]:
```
#Melisa Moses Dabre 60009220200
#Minal Joshi  60009220180
#Neha Ravishankar 60009220181




import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_excel("/content/OLA_trips_dataset (1).xlsx")

#display the top 5 observations of the dataset
df.head()
```

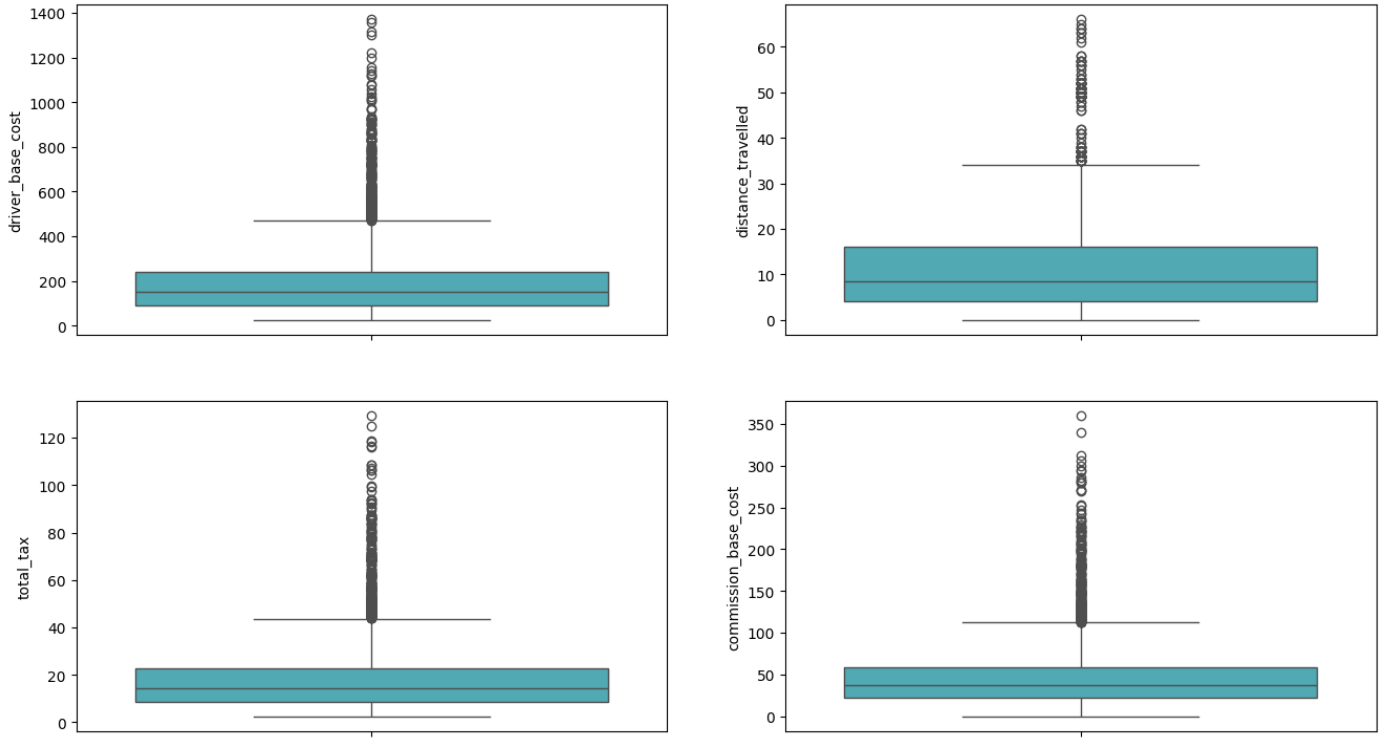
Out[4]:

	booking id	booking_date_time	gender	month	day_of_week	time_of_day	distance_travelled	time_taken
0	1890061540	43249.919444	Male	May	Tue	0.919444	17	58.0
1	1542148932	43153.925000	Female	February	Thu	0.925000	18	43.0
2	1672692603	43194.882639	Female	April	Wed	0.882639	2	5.0
3	1925600201	43258.932639	Female	June	Thu	0.932639	15	49.0
4	1530845664	43150.479861	Male	February	Mon	0.479861	46	0.0

In [4]:
```

```

In [5]:
```
#boxplot to see how the categorical feature “RATING” is distributed with all other four

fig, axes = plt.subplots(2, 2, figsize=(16,9))
sns.boxplot( y='driver_base_cost', data=df, orient='v' , ax=axes[0, 0], palette='YlGnBu'
sns.boxplot( y='distance_travelled', data=df, orient='v' , ax=axes[0, 1], palette='YlGnB
sns.boxplot( y='total_tax', data=df, orient='v' , ax=axes[1, 0], palette='YlGnBu')
sns.boxplot( y='commission_base_cost', data=df, orient='v' , ax=axes[1, 1], palette='YlG
plt.show()
```

```
<ipython-input-5-044507b13909>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.
Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot( y='driver_base_cost', data=df, orient='v' , ax=axes[0, 0], palette='YlGnB
u')
<ipython-input-5-044507b13909>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.
Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot( y='distance_travelled', data=df, orient='v' , ax=axes[0, 1], palette='YlG
nBu')
<ipython-input-5-044507b13909>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.
Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot( y='total_tax', data=df, orient='v' , ax=axes[1, 0], palette='YlGnBu')
<ipython-input-5-044507b13909>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.
Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot( y='commission_base_cost', data=df, orient='v' , ax=axes[1, 1], palette='Y
lGnBu')
```

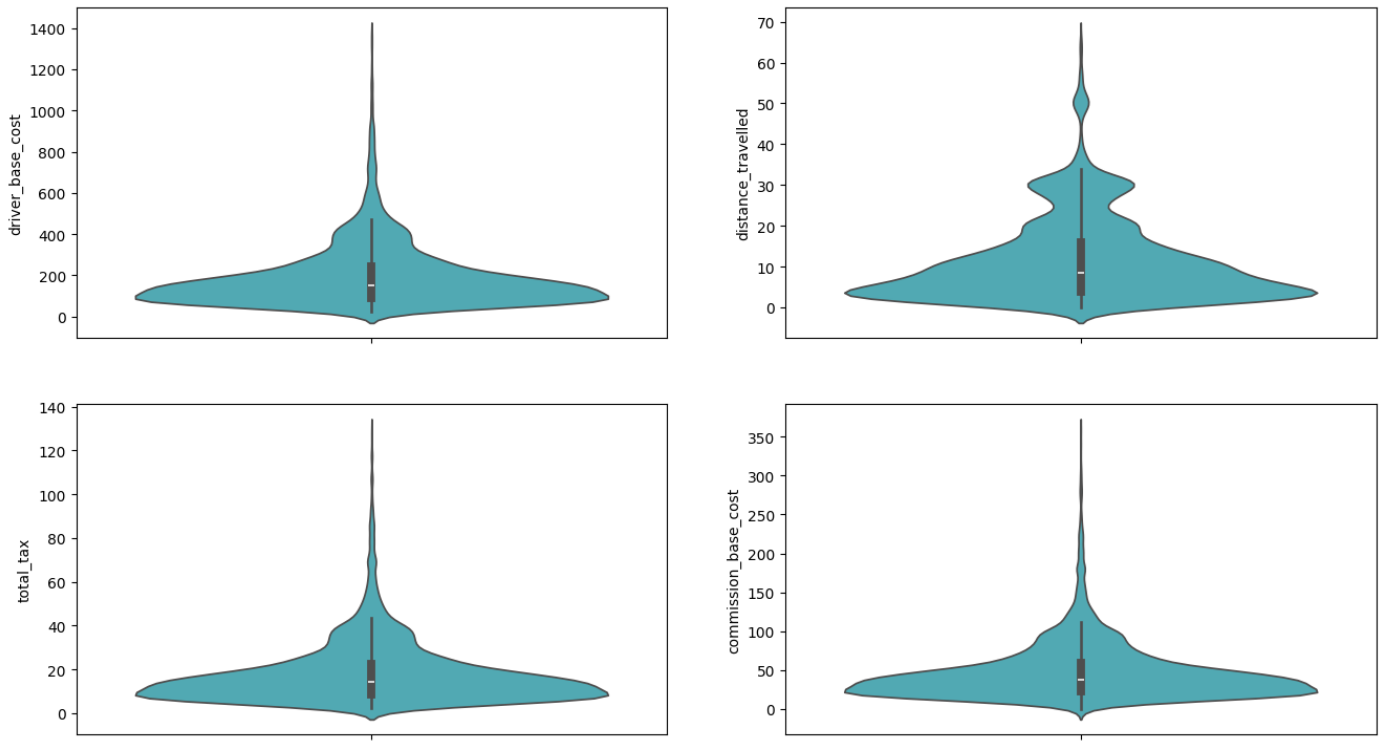


In [6]:

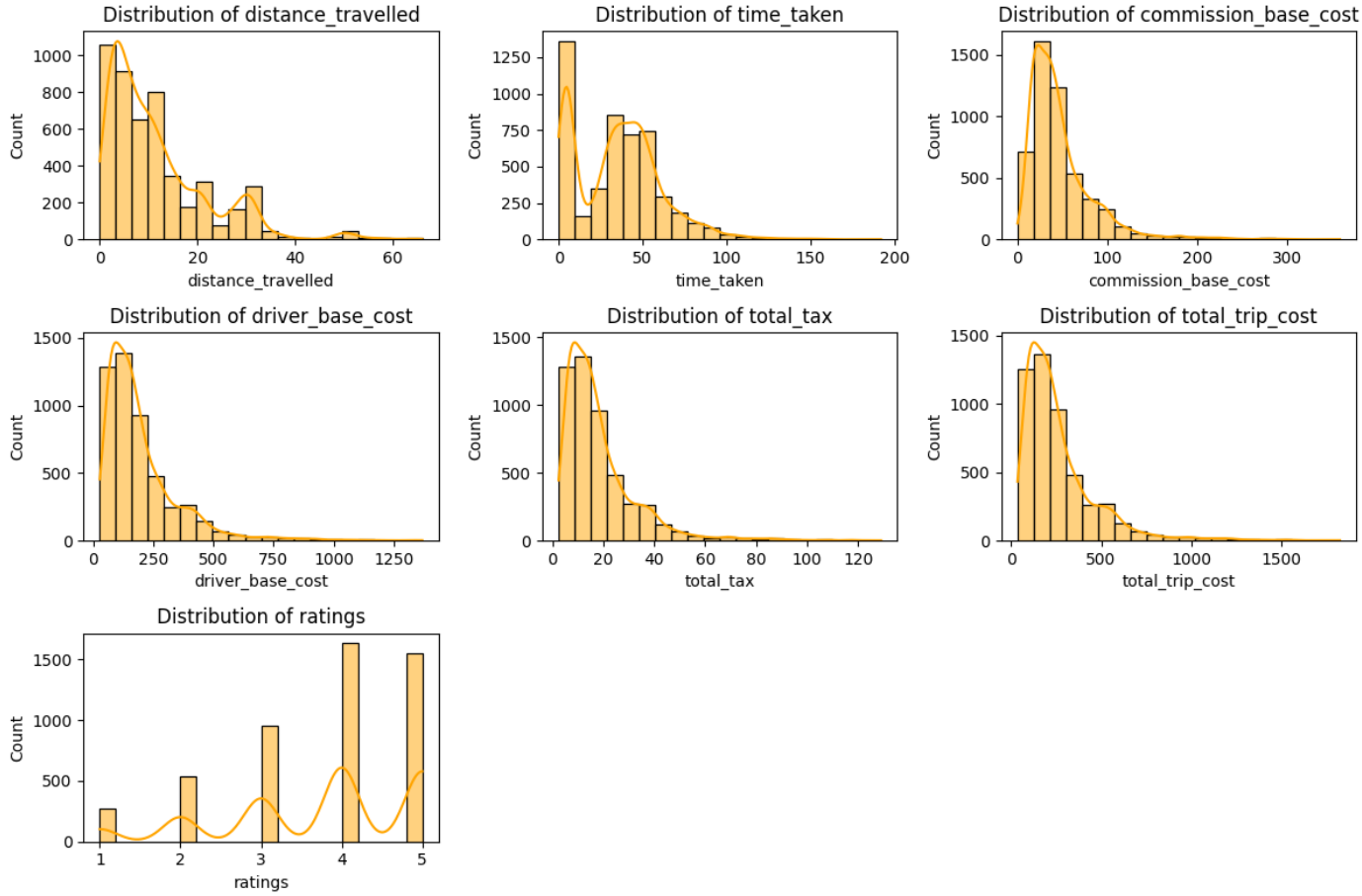
```
import matplotlib.pyplot as plt
import seaborn as sns

# Create subplots
fig, axes = plt.subplots(2, 2, figsize=(16,9))

# Plot violin plots
sns.violinplot(y='driver_base_cost', data=df, ax=axes[0, 0], palette='YlGnBu')
sns.violinplot(y='distance_travelled', data=df, ax=axes[0, 1], palette='YlGnBu')
sns.violinplot(y='total_tax', data=df, ax=axes[1, 0], palette='YlGnBu')
sns.violinplot(y='commission_base_cost', data=df, ax=axes[1, 1], palette='YlGnBu')
```

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

```
<ipython-input-6-5e18ee8dc5f9>:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.
Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.violinplot(y='driver_base_cost', data=df, ax=axes[0, 0], palette='YlGnBu')
<ipython-input-6-5e18ee8dc5f9>:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.
Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.violinplot(y='distance_travelled', data=df, ax=axes[0, 1], palette='YlGnBu')
<ipython-input-6-5e18ee8dc5f9>:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.
Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.violinplot(y='total_tax', data=df, ax=axes[1, 0], palette='YlGnBu')
<ipython-input-6-5e18ee8dc5f9>:11: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.
Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.violinplot(y='commission_base_cost', data=df, ax=axes[1, 1], palette='YlGnBu')
```



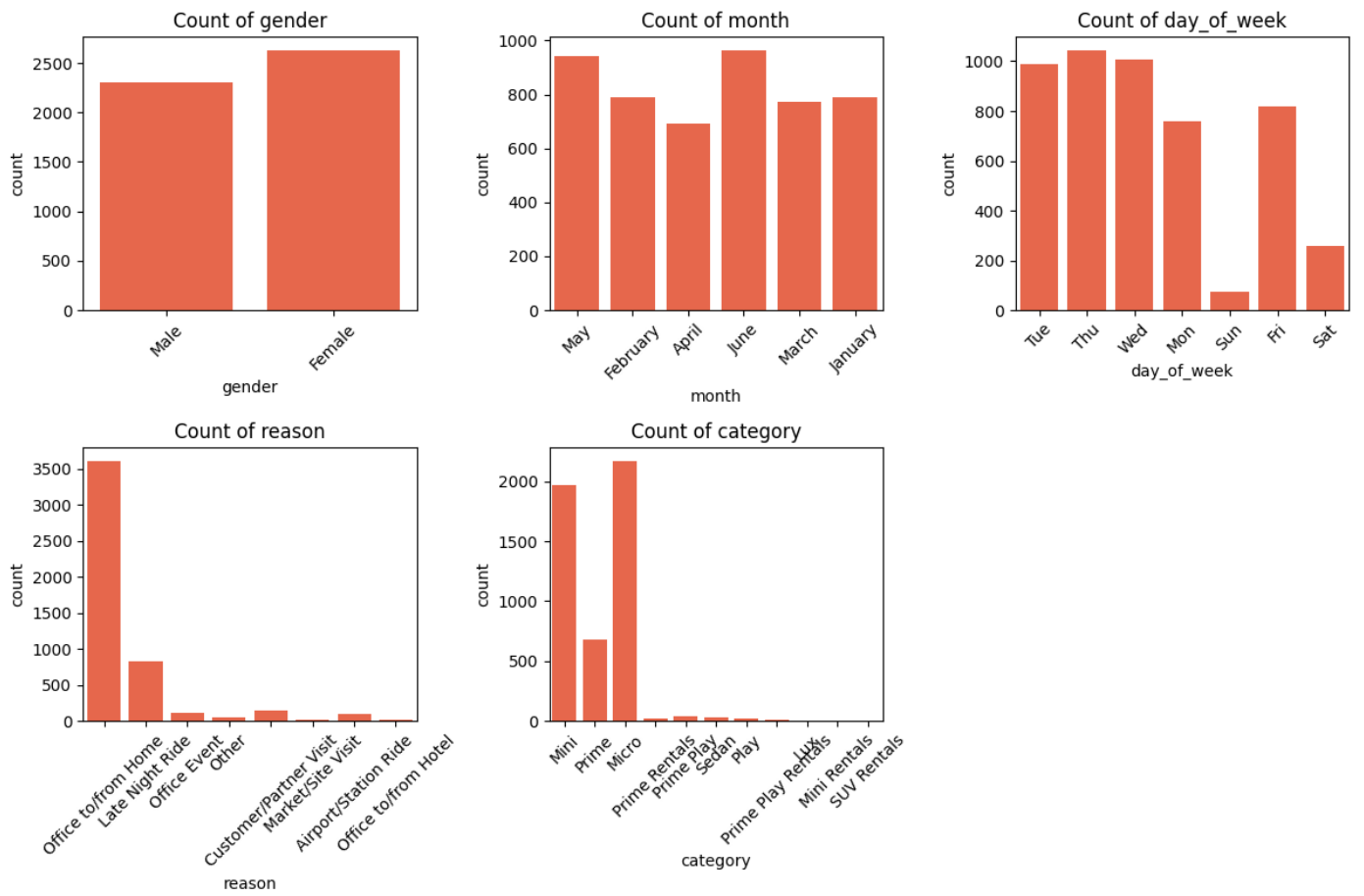
In [7]:
```
# Histograms for Numerical Variables with Warm Color Palette
plt.figure(figsize=(12, 8))
numerical_vars = ['distance_travelled', 'time_taken', 'commission_base_cost', 'driver_ba
for i, var in enumerate(numerical_vars, 1):
    plt.subplot(3, 3, i)
    sns.histplot(data=df, x=var, bins=20, kde=True, color='orange')
    plt.title(f'Distribution of {var}')
plt.tight_layout()
plt.show()
```



In [8]:

```
# Countplots for Categorical Variables with Proper Red Color Palette
plt.figure(figsize=(12, 8))
categorical_vars = ['gender', 'month', 'day_of_week', 'reason', 'category']
for i, var in enumerate(categorical_vars, 1):
    plt.subplot(2, 3, i)
    sns.countplot(data=df, x=var, palette=['#FF5733'])  # Using a custom shade of red
    plt.title(f'Count of {var}')
    plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

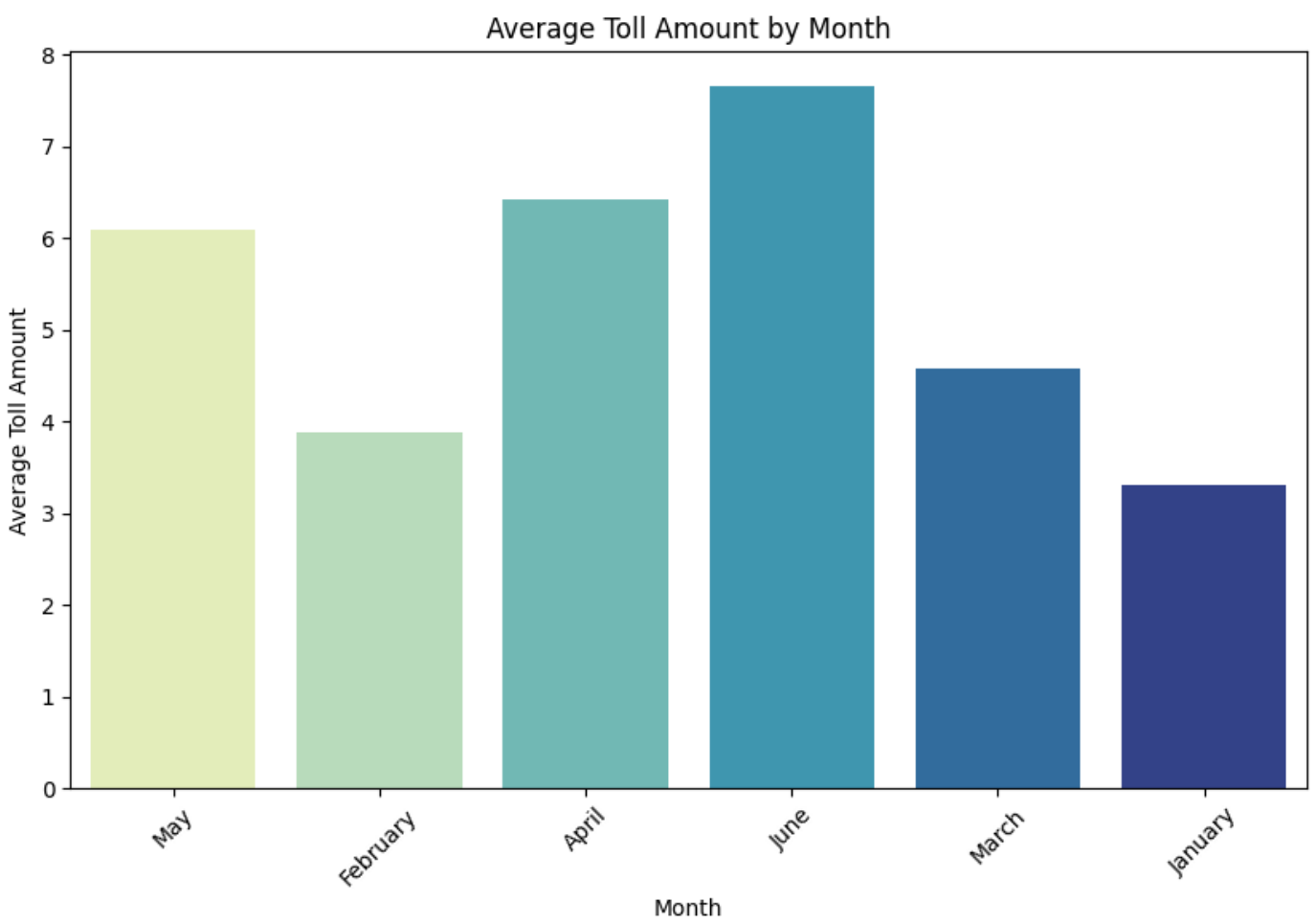
```
<ipython-input-8-f246e7a11a85>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.
Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(data=df, x=var, palette=['#FF5733'])  # Using a custom shade of red
<ipython-input-8-f246e7a11a85>:6: UserWarning:
The palette list has fewer values (1) than needed (2) and will cycle, which may produce
an uninterpretable plot.
  sns.countplot(data=df, x=var, palette=['#FF5733'])  # Using a custom shade of red
<ipython-input-8-f246e7a11a85>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.
Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(data=df, x=var, palette=['#FF5733'])  # Using a custom shade of red
<ipython-input-8-f246e7a11a85>:6: UserWarning:
The palette list has fewer values (1) than needed (6) and will cycle, which may produce
an uninterpretable plot.
  sns.countplot(data=df, x=var, palette=['#FF5733'])  # Using a custom shade of red
<ipython-input-8-f246e7a11a85>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.
Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(data=df, x=var, palette=['#FF5733'])  # Using a custom shade of red
<ipython-input-8-f246e7a11a85>:6: UserWarning:
The palette list has fewer values (1) than needed (7) and will cycle, which may produce
an uninterpretable plot.
  sns.countplot(data=df, x=var, palette=['#FF5733'])  # Using a custom shade of red
<ipython-input-8-f246e7a11a85>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.
Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(data=df, x=var, palette=['#FF5733'])  # Using a custom shade of red
<ipython-input-8-f246e7a11a85>:6: UserWarning:
The palette list has fewer values (1) than needed (8) and will cycle, which may produce
an uninterpretable plot.
  sns.countplot(data=df, x=var, palette=['#FF5733'])  # Using a custom shade of red
<ipython-input-8-f246e7a11a85>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.
Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(data=df, x=var, palette=['#FF5733'])  # Using a custom shade of red
<ipython-input-8-f246e7a11a85>:6: UserWarning:
The palette list has fewer values (1) than needed (11) and will cycle, which may produce
an uninterpretable plot.
  sns.countplot(data=df, x=var, palette=['#FF5733'])  # Using a custom shade of red
```



In [9]:

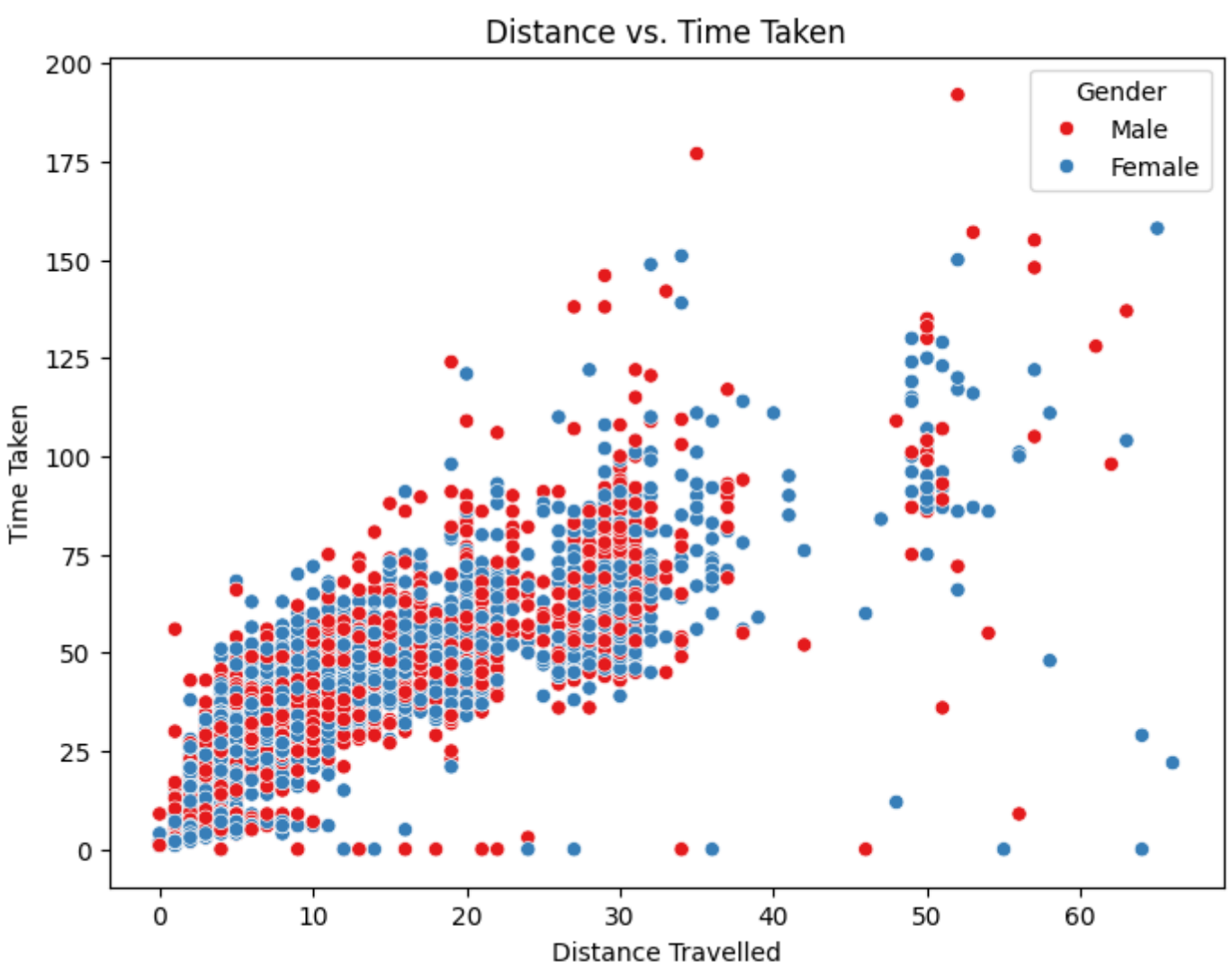
```
# Bar Plot for Toll vs. Month with Different Color Palette
plt.figure(figsize=(10, 6))
sns.barplot(data=df, x='month', y='toll', ci=None, palette='YlGnBu')  # Using Yellow-Gre
plt.title('Average Toll Amount by Month')
plt.xlabel('Month')
plt.ylabel('Average Toll Amount')
plt.xticks(rotation=45)
plt.show()
```

```
<ipython-input-9-ab7a8758f7bc>:3: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(data=df, x='month', y='toll', ci=None, palette='YlGnBu')  # Using Yellow-G
reen-Blue palette
<ipython-input-9-ab7a8758f7bc>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.
Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(data=df, x='month', y='toll', ci=None, palette='YlGnBu')  # Using Yellow-G
reen-Blue palette
```



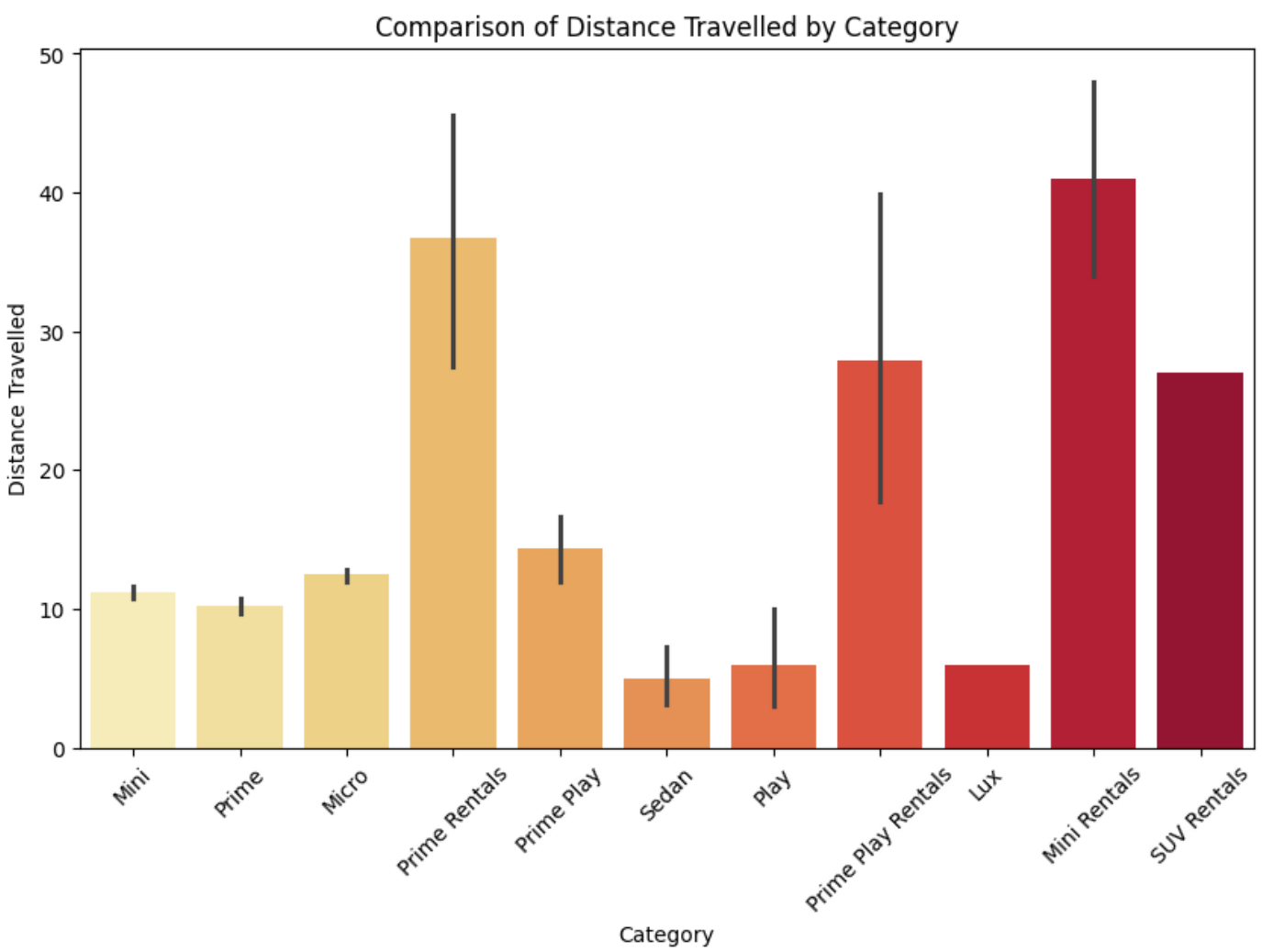
In [10]:

```
# Scatter Plot for Distance vs. Time
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x='distance_travelled', y='time_taken', hue='gender', palette='
plt.title('Distance vs. Time Taken')
plt.xlabel('Distance Travelled')
plt.ylabel('Time Taken')
plt.legend(title='Gender')
plt.show()
```



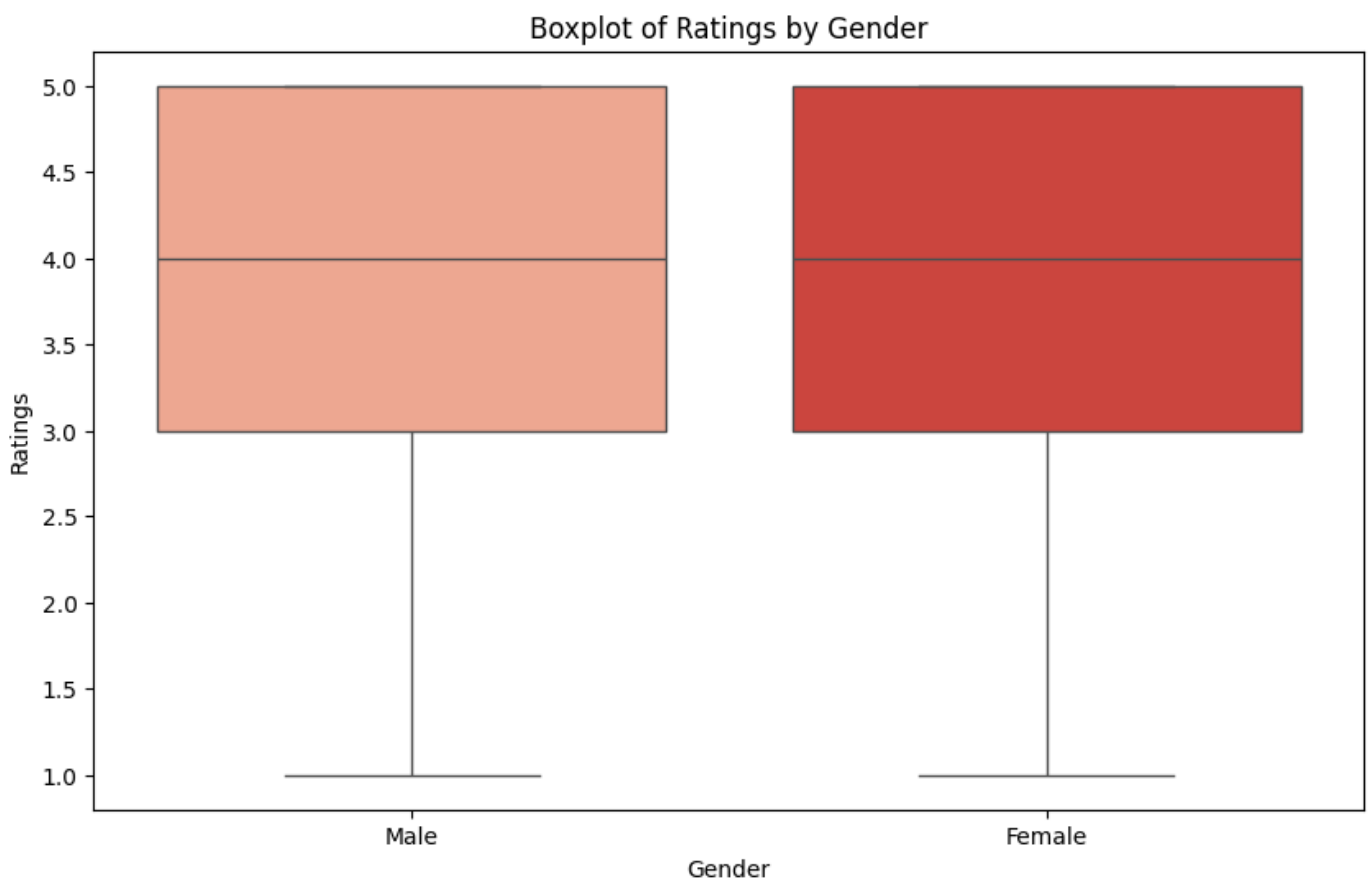
In [11]:
```
# Comparison of Distance Travelled by Category with Warm Colors
plt.figure(figsize=(10, 6))
sns.barplot(data=df, x='category', y='distance_travelled', palette='YlOrRd')  # Using Ye
plt.title('Comparison of Distance Travelled by Category')
plt.xlabel('Category')
plt.ylabel('Distance Travelled')
plt.xticks(rotation=45)
plt.show()
```

```
<ipython-input-11-2833253d2a12>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.
Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(data=df, x='category', y='distance_travelled', palette='YlOrRd')  # Using
Yellow-Orange-Red palette
```



In [12]:
```
# Boxplot for Ratings by Gender
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='gender', y='ratings', palette='Reds')  # Boxplot
plt.title('Boxplot of Ratings by Gender')
plt.xlabel('Gender')
plt.ylabel('Ratings')
plt.show()
```

```
<ipython-input-12-fc2590b0a7d7>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.
Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(data=df, x='gender', y='ratings', palette='Reds')  # Boxplot
```



In [15]:
```
!jupyter nbconvert --to html Dev_PROJECT_EDA.ipynb
```

```
[NbConvertApp] WARNING | pattern 'Dev_PROJECT_EDA.ipynb' matched no files
This application is used to convert notebook files (*.ipynb)
        to various other formats.

        WARNING: THE COMMANDLINE INTERFACE MAY CHANGE IN FUTURE RELEASES.

Options
=======
The options below are convenience aliases to configurable class-options,
as listed in the "Equivalent to" description-line of the aliases.
To see all configurable class-options for some <cmd>, use:
    <cmd> --help-all

--debug
    set log level to logging.DEBUG (maximize logging output)
    Equivalent to: [--Application.log_level=10]
--show-config
    Show the application's configuration (human-readable format)
    Equivalent to: [--Application.show_config=True]
--show-config-json
    Show the application's configuration (json format)
    Equivalent to: [--Application.show_config_json=True]
--generate-config
    generate default config file
    Equivalent to: [--JupyterApp.generate_config=True]
-y
    Answer yes to any questions instead of prompting.
    Equivalent to: [--JupyterApp.answer_yes=True]
--execute
    Execute the notebook prior to export.
    Equivalent to: [--ExecutePreprocessor.enabled=True]
--allow-errors
    Continue notebook execution even if one of the cells throws an error and include the
error message in the cell output (the default behaviour is to abort conversion). This fl
ag is only relevant if '--execute' was specified, too.
    Equivalent to: [--ExecutePreprocessor.allow_errors=True]
--stdin
    read a single notebook file from stdin. Write the resulting notebook with default ba
sename 'notebook.*'
    Equivalent to: [--NbConvertApp.from_stdin=True]
--stdout
    Write notebook output to stdout instead of files.
    Equivalent to: [--NbConvertApp.writer_class=StdoutWriter]
--inplace
    Run nbconvert in place, overwriting the existing notebook (only
            relevant when converting to notebook format)
    Equivalent to: [--NbConvertApp.use_output_suffix=False --NbConvertApp.export_format=
notebook --FilesWriter.build_directory=]
--clear-output
    Clear output of current file and save in place,
            overwriting the existing notebook.
    Equivalent to: [--NbConvertApp.use_output_suffix=False --NbConvertApp.export_format=
notebook --FilesWriter.build_directory= --ClearOutputPreprocessor.enabled=True]
--no-prompt
    Exclude input and output prompts from converted document.
    Equivalent to: [--TemplateExporter.exclude_input_prompt=True --TemplateExporter.excl
ude_output_prompt=True]
--no-input
    Exclude input cells and output prompts from converted document.
            This mode is ideal for generating code-free reports.
    Equivalent to: [--TemplateExporter.exclude_output_prompt=True --TemplateExporter.exc
lude_input=True --TemplateExporter.exclude_input_prompt=True]
--allow-chromium-download
    Whether to allow downloading chromium if no suitable version is found on the system.
```

```
    Equivalent to: [--WebPDFExporter.allow_chromium_download=True]
--disable-chromium-sandbox
    Disable chromium security sandbox when converting to PDF..
    Equivalent to: [--WebPDFExporter.disable_sandbox=True]
--show-input
    Shows code input. This flag is only useful for dejavu users.
    Equivalent to: [--TemplateExporter.exclude_input=False]
--embed-images
    Embed the images as base64 dataurls in the output. This flag is only useful for the
HTML/WebPDF/Slides exports.
    Equivalent to: [--HTMLExporter.embed_images=True]
--sanitize-html
    Whether the HTML in Markdown cells and cell outputs should be sanitized..
    Equivalent to: [--HTMLExporter.sanitize_html=True]
--log-level=<Enum>
    Set the log level by value or name.
    Choices: any of [0, 10, 20, 30, 40, 50, 'DEBUG', 'INFO', 'WARN', 'ERROR', 'CRITICA
L']
    Default: 30
    Equivalent to: [--Application.log_level]
--config=<Unicode>
    Full path of a config file.
    Default: ''
    Equivalent to: [--JupyterApp.config_file]
--to=<Unicode>
    The export format to be used, either one of the built-in formats
            ['asciidoc', 'custom', 'html', 'latex', 'markdown', 'notebook', 'pdf', 'pyth
on', 'rst', 'script', 'slides', 'webpdf']
            or a dotted object name that represents the import path for an
            ``Exporter`` class
    Default: ''
    Equivalent to: [--NbConvertApp.export_format]
--template=<Unicode>
    Name of the template to use
    Default: ''
    Equivalent to: [--TemplateExporter.template_name]
--template-file=<Unicode>
    Name of the template file to use
    Default: None
    Equivalent to: [--TemplateExporter.template_file]
--theme=<Unicode>
    Template specific theme(e.g. the name of a JupyterLab CSS theme distributed
    as prebuilt extension for the lab template)
    Default: 'light'
    Equivalent to: [--HTMLExporter.theme]
--sanitize_html=<Bool>
    Whether the HTML in Markdown cells and cell outputs should be sanitized.This
    should be set to True by nbviewer or similar tools.
    Default: False
    Equivalent to: [--HTMLExporter.sanitize_html]
--writer=<DottedObjectName>
    Writer class used to write the
                                        results of the conversion
    Default: 'FilesWriter'
    Equivalent to: [--NbConvertApp.writer_class]
--post=<DottedOrNone>
    PostProcessor class used to write the
                                        results of the conversion
    Default: ''
    Equivalent to: [--NbConvertApp.postprocessor_class]
--output=<Unicode>
    overwrite base name use for output files.
                can only be used when converting one notebook at a time.
    Default: ''
```

```
    Equivalent to: [--NbConvertApp.output_base]
--output-dir=<Unicode>
    Directory to write output(s) to. Defaults
                                to output to the directory of each notebook. To recove
r
                                previous default behaviour (outputting to the current
                                working directory) use . as the flag value.
    Default: ''
    Equivalent to: [--FilesWriter.build_directory]
--reveal-prefix=<Unicode>
    The URL prefix for reveal.js (version 3.x).
            This defaults to the reveal CDN, but can be any url pointing to a copy
            of reveal.js.
            For speaker notes to work, this must be a relative path to a local
            copy of reveal.js: e.g., "reveal.js".
            If a relative path is given, it must be a subdirectory of the
            current directory (from which the server is run).
            See the usage documentation
            (https://nbconvert.readthedocs.io/en/latest/usage.html#reveal-js-html-slides
how)
            for more details.
    Default: ''
    Equivalent to: [--SlidesExporter.reveal_url_prefix]
--nbformat=<Enum>
    The nbformat version to write.
            Use this to downgrade notebooks.
    Choices: any of [1, 2, 3, 4]
    Default: 4
    Equivalent to: [--NotebookExporter.nbformat_version]

Examples
--------

    The simplest way to use nbconvert is

            > jupyter nbconvert mynotebook.ipynb --to html

            Options include ['asciidoc', 'custom', 'html', 'latex', 'markdown', 'noteboo
k', 'pdf', 'python', 'rst', 'script', 'slides', 'webpdf'].

            > jupyter nbconvert --to latex mynotebook.ipynb

            Both HTML and LaTeX support multiple output templates. LaTeX includes
            'base', 'article' and 'report'.  HTML includes 'basic', 'lab' and
            'classic'. You can specify the flavor of the format used.

            > jupyter nbconvert --to html --template lab mynotebook.ipynb

            You can also pipe the output to stdout, rather than a file

            > jupyter nbconvert mynotebook.ipynb --stdout

            PDF is generated via latex

            > jupyter nbconvert mynotebook.ipynb --to pdf

            You can get (and serve) a Reveal.js-powered slideshow

            > jupyter nbconvert myslides.ipynb --to slides --post serve

            Multiple notebooks can be given at the command line in a couple of
            different ways:

            > jupyter nbconvert notebook*.ipynb
```

```
> jupyter nbconvert notebook1.ipynb notebook2.ipynb

or you can specify the notebooks list in a config file, containing::

    c.NbConvertApp.notebooks = ["my_notebook.ipynb"]

> jupyter nbconvert --config mycfg.py

To see all available configurables, use `--help-all`.
```

In [13]:

In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline



#display the top 5 observations of the dataset
```

In [3]:
```
df = pd.read_excel("/content/OLA_trips_dataset (1).xlsx")
```

In [4]:
```
df.head()
```

Out[4]:

	booking id	booking_date_time	gender	month	day_of_week	time_of_day	distance_travelled
0	1890061540	43249.919444	Male	May	Tue	0.919444	17
1	1542148932	43153.925000	Female	February	Thu	0.925000	18
2	1672692603	43194.882639	Female	April	Wed	0.882639	2
3	1925600201	43258.932639	Female	June	Thu	0.932639	15
4	1530845664	43150.479861	Male	February	Mon	0.479861	46

In [5]:
```
#several unique values in each column
df.nunique()
df.tail()
```

Out[5]:

	booking id	booking_date_time	gender	month	day_of_week	time_of_day	distance_tr
4945	1901877370	43252.909722	Female	June	Fri	0.909722	
4946	1867091987	43243.933333	Female	May	Wed	0.933333	
4947	1747322670	43214.971528	Male	April	Tue	0.971528	
4948	1635338680	43183.008333	Male	March	Sat	0.008333	
4949	OSN_1039565727	43270.986806	Female	June	Tue	0.986806	

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

Out[6]:

	booking_date_time	time_of_day	distance_travelled	time_taken	toll	commission_
count	4950.000000	4950.000000	4950.000000	4950.000000	4950.000000	49
mean	43195.816098	0.686199	11.713333	35.126137	5.408081	
std	53.621694	0.373218	10.338660	25.592958	15.915681	
min	43101.043056	0.000000	0.000000	0.000000	0.000000	
25%	43147.922222	0.273090	4.000000	7.000000	0.000000	
50%	43195.875000	0.900000	8.500000	35.000000	0.000000	
75%	43245.182465	0.936806	16.000000	50.000000	0.000000	
max	43281.198611	0.999306	66.000000	192.000000	140.000000	3

In [7]:
```
#identify null values in the data
df.isnull()
```

Out[7]:

	booking id	booking_date_time	gender	month	day_of_week	time_of_day	distance_travelled
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...
4945	False	False	False	False	False	False	False
4946	False	False	False	False	False	False	False
4947	False	False	False	False	False	False	False
4948	False	False	False	False	False	False	False
4949	False	False	False	False	False	False	False

4950 rows × 16 columns

In [8]:
```
#used to get the number of missing records in each column
df.isnull().sum()
```

Out[8]:
```
booking id              0
booking_date_time       0
gender                 18
month                   0
day_of_week             0
time_of_day             0
distance_travelled      0
time_taken              0
reason                 93
toll                    0
category                0
commission_base_cost    0
```

```
driver_base_cost         0
total_tax                0
total_trip_cost         15
ratings                  0
dtype: int64
```

In [9]:
```
#calculate the percentage of missing values in each column
(df.isnull().sum()/(len(df)))*100
```

Out[9]:
```
booking id              0.000000
booking_date_time       0.000000
gender                  0.363636
month                   0.000000
day_of_week             0.000000
time_of_day             0.000000
distance_travelled      0.000000
time_taken              0.000000
reason                  1.878788
toll                    0.000000
category                0.000000
commission_base_cost    0.000000
driver_base_cost        0.000000
total_tax               0.000000
total_trip_cost         0.303030
ratings                 0.000000
dtype: float64
```

In [10]:
```
df[df.duplicated()]
```

Out[10]:

booking id	booking_date_time	gender	month	day_of_week	time_of_day	distance_travelled	time_

In [10]:

In [10]:

In [11]:
```
#used to get the number of missing records in each column
df.isnull().sum()
```

Out[11]:
```
booking id              0
booking_date_time       0
gender                 18
month                   0
day_of_week             0
time_of_day             0
distance_travelled      0
time_taken              0
reason                 93
toll                    0
category                0
commission_base_cost    0
driver_base_cost        0
total_tax               0
total_trip_cost        15
ratings                 0
dtype: int64
```

In [12]:
```
import numpy as np
```

```
# Assuming df is your pandas DataFrame
mean = np.mean(df["total_trip_cost"])
```

In [13]:
```
df["total_trip_cost"] = df["total_trip_cost"].fillna(mean)
```

In [14]:
```
#used to get the number of missing records in each column
df.isnull().sum()
```

Out[14]:
```
booking id               0
booking_date_time        0
gender                  18
month                    0
day_of_week              0
time_of_day              0
distance_travelled       0
time_taken               0
reason                  93
toll                     0
category                 0
commission_base_cost     0
driver_base_cost         0
total_tax                0
total_trip_cost          0
ratings                  0
dtype: int64
```

In [15]:
```
df = df.dropna(subset=["reason"])
```

In [16]:
```
#used to get the number of missing records in each column
df.isnull().sum()
```

Out[16]:
```
booking id               0
booking_date_time        0
gender                  17
month                    0
day_of_week              0
time_of_day              0
distance_travelled       0
time_taken               0
reason                   0
toll                     0
category                 0
commission_base_cost     0
driver_base_cost         0
total_tax                0
total_trip_cost          0
ratings                  0
dtype: int64
```

In [17]:
```
df = df.dropna(subset=["time_of_day"])
```

In [18]:
```
#used to get the number of missing records in each column
df.isnull().sum()
```

Out[18]:
```
booking id               0
booking_date_time        0
gender                  17
month                    0
day_of_week              0
time_of_day              0
distance_travelled       0
time_taken               0
```

```
reason                  0
toll                    0
category                0
commission_base_cost    0
driver_base_cost        0
total_tax               0
total_trip_cost         0
ratings                 0
dtype: int64
```

In [19]:
```
df['gender'].value_counts()
```

Out[19]:
```
Female    2580
Male      2260
Name: gender, dtype: int64
```

In [20]:
```
#filling in the most probable value
df.fillna({'gender':'Female'},inplace=True)
```

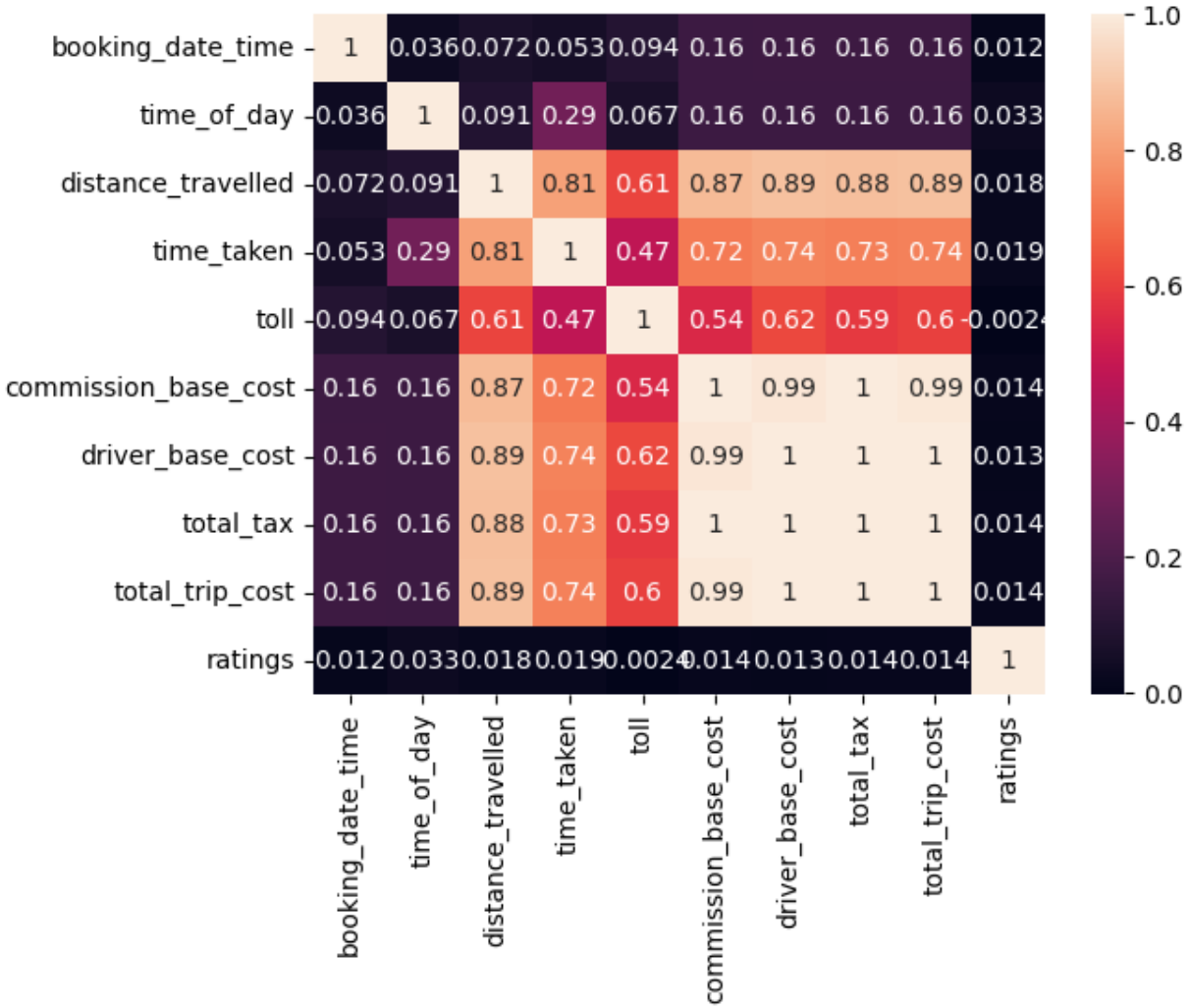
In [21]:
```
#used to get the number of missing records in each column
df.isnull().sum()
```

Out[21]:
```
booking id              0
booking_date_time       0
gender                  0
month                   0
day_of_week             0
time_of_day             0
distance_travelled      0
time_taken              0
reason                  0
toll                    0
category                0
commission_base_cost    0
driver_base_cost        0
total_tax               0
total_trip_cost         0
ratings                 0
dtype: int64
```

In [22]:
```
sns.heatmap(df.corr(),annot=True)
```

```
<ipython-input-22-8df7bcac526d>:1: FutureWarning: The default value of numeric
_only in DataFrame.corr is deprecated. In a future version, it will default to
False. Select only valid columns or specify the value of numeric_only to silen
ce this warning.
  sns.heatmap(df.corr(),annot=True)
```

Out[22]:
```
<Axes: >
```



In [23]:

```
output = []
for col in df.columns:
    unique = df[col].nunique()
    colType = str(df[col].dtype)
    categories=df[col].unique()

    output.append([col, unique, colType,categories])

output = pd.DataFrame(output)
output.columns = ['colName','unique','dtype','categories']
output
```

Out[23]:

	colName	unique	dtype	categories
0	booking id	4857	object	[1890061540, 1542148932, 1672692603, 192560020...
1	booking_date_time	4498	float64	[43249.919444444444, 43153.925, 43194.88263888...
2	gender	2	object	[Male, Female]
3	month	6	object	[May, February, April, June, March, January]
4	day_of_week	7	object	[Tue, Thu, Wed, Mon, Sun, Fri, Sat]
5	time_of_day	799	float64	[0.9194444444444444, 0.925, 0.8826388888888889...
6	distance_travelled	61	int64	[17, 18, 2, 15, 46, 30, 4, 5, 3, 62, 21, 6, 24...
7	time_taken	295	float64	[58.0, 43.0, 5.0, 49.0, 0.0, 91.0, 4.0, 18.0, ...
8	reason	8	object	[Office to/from Home, Late Night Ride, Office ...

	colName	unique	dtype	categories
9	toll	7	int64	[0, 35, 60, 70, 120, 105, 140]
10	category	11	object	[Mini, Prime, Micro, Prime Rentals, Prime Play...
11	commission_base_cost	2896	float64	[57.73, 52.04, 19.7, 51.24, 195.92, 132.71, 19...
12	driver_base_cost	3357	float64	[230.91, 208.16, 78.81, 239.96, 783.68, 565.86...
13	total_tax	2161	float64	[21.94, 19.76, 7.49, 21.22, 74.45, 52.18, 7.39...
14	total_trip_cost	1806	float64	[311.0, 279.96, 106.0, 312.0, 1054.05, 751.0, ...
15	ratings	5	int64	[3, 5, 4, 1, 2]

In [24]:
```
corr=df.corr()
```

```
<ipython-input-24-0014364bc22a>:1: FutureWarning: The default value of numeric
_only in DataFrame.corr is deprecated. In a future version, it will default to
False. Select only valid columns or specify the value of numeric_only to silen
ce this warning.
  corr=df.corr()
```

In [25]:
```
print(np.max(df['total_trip_cost']))
print(np.min(df['total_trip_cost']))
```

```
1828.12
34.0
```

In [26]:
```
df['total_trip_cost bins'] = pd.cut(x=df['total_trip_cost'], bins=[100,300,500
```

In [27]:
```
df['total_trip_cost bins'].value_counts()
```

Out[27]:
```
(100, 300]      2742
(300, 500]       804
(500, 700]       360
(700, 900]        83
(900, 1100]       46
(1100, 1300]      32
(1300, 1500]       8
(1500, 1700]       7
(1700, 1900]       4
Name: total_trip_cost bins, dtype: int64
```

In [28]:
```
pd.crosstab(df['ratings'], df['gender'])
```

Out[28]:

gender	Female	Male
ratings		
1	149	125
2	278	248
3	515	417
4	853	747
5	802	723

In [29]:
```
pd.crosstab(df['ratings'], df['month'])
```

Out[29]:

month	April	February	January	June	March	May

month ratings	April	February	January	June	March	May
ratings						
1	34	48	53	59	34	46
2	64	79	93	113	84	93
3	126	168	142	181	152	163
4	217	247	268	314	251	303
5	240	243	230	271	234	307

In [31]:
```
!jupyter nbconvert --to html DEV_PROJECT_1.ipynb
```

```
[NbConvertApp] WARNING | pattern 'DEV_PROJECT_1.ipynb' matched no files
This application is used to convert notebook files (*.ipynb)
        to various other formats.

        WARNING: THE COMMANDLINE INTERFACE MAY CHANGE IN FUTURE RELEASES.

Options
=======
The options below are convenience aliases to configurable class-options,
as listed in the "Equivalent to" description-line of the aliases.
To see all configurable class-options for some <cmd>, use:
    <cmd> --help-all

--debug
    set log level to logging.DEBUG (maximize logging output)
    Equivalent to: [--Application.log_level=10]
--show-config
    Show the application's configuration (human-readable format)
    Equivalent to: [--Application.show_config=True]
--show-config-json
    Show the application's configuration (json format)
    Equivalent to: [--Application.show_config_json=True]
--generate-config
    generate default config file
    Equivalent to: [--JupyterApp.generate_config=True]
-y
    Answer yes to any questions instead of prompting.
    Equivalent to: [--JupyterApp.answer_yes=True]
--execute
    Execute the notebook prior to export.
    Equivalent to: [--ExecutePreprocessor.enabled=True]
--allow-errors
    Continue notebook execution even if one of the cells throws an error and i
nclude the error message in the cell output (the default behaviour is to abort
conversion). This flag is only relevant if '--execute' was specified, too.
    Equivalent to: [--ExecutePreprocessor.allow_errors=True]
--stdin
    read a single notebook file from stdin. Write the resulting notebook with
default basename 'notebook.*'
    Equivalent to: [--NbConvertApp.from_stdin=True]
--stdout
    Write notebook output to stdout instead of files.
    Equivalent to: [--NbConvertApp.writer_class=StdoutWriter]
--inplace
    Run nbconvert in place, overwriting the existing notebook (only
            relevant when converting to notebook format)
    Equivalent to: [--NbConvertApp.use_output_suffix=False --NbConvertApp.expo
```

```
rt_format=notebook --FilesWriter.build_directory=]
--clear-output
    Clear output of current file and save in place,
            overwriting the existing notebook.
    Equivalent to: [--NbConvertApp.use_output_suffix=False --NbConvertApp.expo
rt_format=notebook --FilesWriter.build_directory= --ClearOutputPreprocessor.en
abled=True]
--no-prompt
    Exclude input and output prompts from converted document.
    Equivalent to: [--TemplateExporter.exclude_input_prompt=True --TemplateExp
orter.exclude_output_prompt=True]
--no-input
    Exclude input cells and output prompts from converted document.
            This mode is ideal for generating code-free reports.
    Equivalent to: [--TemplateExporter.exclude_output_prompt=True --TemplateEx
porter.exclude_input=True --TemplateExporter.exclude_input_prompt=True]
--allow-chromium-download
    Whether to allow downloading chromium if no suitable version is found on t
he system.
    Equivalent to: [--WebPDFExporter.allow_chromium_download=True]
--disable-chromium-sandbox
    Disable chromium security sandbox when converting to PDF..
    Equivalent to: [--WebPDFExporter.disable_sandbox=True]
--show-input
    Shows code input. This flag is only useful for dejavu users.
    Equivalent to: [--TemplateExporter.exclude_input=False]
--embed-images
    Embed the images as base64 dataurls in the output. This flag is only usefu
l for the HTML/WebPDF/Slides exports.
    Equivalent to: [--HTMLExporter.embed_images=True]
--sanitize-html
    Whether the HTML in Markdown cells and cell outputs should be sanitized..
    Equivalent to: [--HTMLExporter.sanitize_html=True]
--log-level=<Enum>
    Set the log level by value or name.
    Choices: any of [0, 10, 20, 30, 40, 50, 'DEBUG', 'INFO', 'WARN', 'ERROR',
'CRITICAL']
    Default: 30
    Equivalent to: [--Application.log_level]
--config=<Unicode>
    Full path of a config file.
    Default: ''
    Equivalent to: [--JupyterApp.config_file]
--to=<Unicode>
    The export format to be used, either one of the built-in formats
            ['asciidoc', 'custom', 'html', 'latex', 'markdown', 'notebook', 'p
df', 'python', 'rst', 'script', 'slides', 'webpdf']
            or a dotted object name that represents the import path for an
            ``Exporter`` class
    Default: ''
    Equivalent to: [--NbConvertApp.export_format]
--template=<Unicode>
    Name of the template to use
    Default: ''
    Equivalent to: [--TemplateExporter.template_name]
--template-file=<Unicode>
    Name of the template file to use
    Default: None
    Equivalent to: [--TemplateExporter.template_file]
--theme=<Unicode>
    Template specific theme(e.g. the name of a JupyterLab CSS theme distribute
d
```

```
    as prebuilt extension for the lab template)
    Default: 'light'
    Equivalent to: [--HTMLExporter.theme]
--sanitize_html=<Bool>
    Whether the HTML in Markdown cells and cell outputs should be sanitized.Th
is
    should be set to True by nbviewer or similar tools.
    Default: False
    Equivalent to: [--HTMLExporter.sanitize_html]
--writer=<DottedObjectName>
    Writer class used to write the
                                        results of the conversion
    Default: 'FilesWriter'
    Equivalent to: [--NbConvertApp.writer_class]
--post=<DottedOrNone>
    PostProcessor class used to write the
                                        results of the conversion
    Default: ''
    Equivalent to: [--NbConvertApp.postprocessor_class]
--output=<Unicode>
    overwrite base name use for output files.
                can only be used when converting one notebook at a time.
    Default: ''
    Equivalent to: [--NbConvertApp.output_base]
--output-dir=<Unicode>
    Directory to write output(s) to. Defaults
                                    to output to the directory of each notebook.
To recover
                                    previous default behaviour (outputting to th
e current
                                    working directory) use . as the flag value.
    Default: ''
    Equivalent to: [--FilesWriter.build_directory]
--reveal-prefix=<Unicode>
    The URL prefix for reveal.js (version 3.x).
            This defaults to the reveal CDN, but can be any url pointing to a
copy
            of reveal.js.
            For speaker notes to work, this must be a relative path to a local
            copy of reveal.js: e.g., "reveal.js".
            If a relative path is given, it must be a subdirectory of the
            current directory (from which the server is run).
            See the usage documentation
            (https://nbconvert.readthedocs.io/en/latest/usage.html#reveal-js-h
tml-slideshow)
            for more details.
    Default: ''
    Equivalent to: [--SlidesExporter.reveal_url_prefix]
--nbformat=<Enum>
    The nbformat version to write.
            Use this to downgrade notebooks.
    Choices: any of [1, 2, 3, 4]
    Default: 4
    Equivalent to: [--NotebookExporter.nbformat_version]

Examples
--------

    The simplest way to use nbconvert is

            > jupyter nbconvert mynotebook.ipynb --to html
```

```
            Options include ['asciidoc', 'custom', 'html', 'latex', 'markdow
        n', 'notebook', 'pdf', 'python', 'rst', 'script', 'slides', 'webpdf'].

            > jupyter nbconvert --to latex mynotebook.ipynb

            Both HTML and LaTeX support multiple output templates. LaTeX inclu
        des
            'base', 'article' and 'report'.  HTML includes 'basic', 'lab' and
            'classic'. You can specify the flavor of the format used.

            > jupyter nbconvert --to html --template lab mynotebook.ipynb

            You can also pipe the output to stdout, rather than a file

            > jupyter nbconvert mynotebook.ipynb --stdout

            PDF is generated via latex

            > jupyter nbconvert mynotebook.ipynb --to pdf

            You can get (and serve) a Reveal.js-powered slideshow

            > jupyter nbconvert myslides.ipynb --to slides --post serve

            Multiple notebooks can be given at the command line in a couple of
            different ways:

            > jupyter nbconvert notebook*.ipynb
            > jupyter nbconvert notebook1.ipynb notebook2.ipynb

            or you can specify the notebooks list in a config file, containin
        g::

                c.NbConvertApp.notebooks = ["my_notebook.ipynb"]

            > jupyter nbconvert --config mycfg.py

        To see all available configurables, use `--help-all`.
```

In [30]:

In [30]:

In [30]:

In [30]: