

```
In [14]: df = pd.read_csv('Online_Learning_Behavior_Dataset_Worldwide')  
df . head()
```

FileNotFoundError Traceback (most recent call last)

Cell In[14], line 1

```
----> 1 df = pd.read_csv(  
      2 df . head()
```

File D:\anaconda\envs\project2\Lib\site-packages\pandas\io\parsers\readers.py:873, in read_csv(filepath_or_buffer, sep, delimiter, header, names, index_col, usecols, dtype, engine, converters, true_values, false_values, skipinitialspace, skiprows, skipfooter, nrows, na_values, keep_default_na, na_filter, skip_blank_lines, parse_dates, date_format, dayfirst, cache_dates, iterator, chunksize, compression, thousands, decimal, lineterminator, quotechar, quoting, doublequote, escapechar, comment, encoding, encoding_errors, dialect, on_bad_lines, low_memory, memory_map, float_precision, storage_options, dtype_backend)

```
    861 kwds_defaults = _refine_defaults_read(  
    862     dialect,  
    863     delimiter,  
(...)    869     dtype_backend=dtype_backend,  
    870 )  
    871 kwds.update(kwds_defaults)  
--> 873 return _read(filepath_or_buffer, kwds)
```

File D:\anaconda\envs\project2\Lib\site-packages\pandas\io\parsers\readers.py:300, in _read(filepath_or_buffer, kwds)

```
    297 _validate_names(kwds.get("names", None))  
    299 # Create the parser.  
--> 300 parser = TextFileReader(filepath_or_buffer, **kwds)  
    302 if chunksize or iterator:  
    303     return parser
```

File D:\anaconda\envs\project2\Lib\site-packages\pandas\io\parsers\readers.py:1645, in TextFileReader.__init__(self, f, engine, **kwds)

```
    1642 self.options["has_index_names"] = kwds["has_index_names"]  
    1644 self.handles: IOHandles | None = None  
-> 1645 self._engine = self._make_engine(f, self.engine)
```

File D:\anaconda\envs\project2\Lib\site-packages\pandas\io\parsers\readers.py:1904, in TextFileReader._make_engine(self, f, engine)

```
    1902 if "b" not in mode:  
    1903     mode += "b"  
-> 1904 self.handles = get_handle(  
    1905     f,  
    1906     mode,  
    1907     encoding=self.options.get(  
    1908     compression=self.options.get(  
    1909     memory_map=self.options.get(  
    1910     is_text=is_text,  
    1911     errors=self.options.get(  
    1912     storage_options=self.options.get(  
    1913 )  
    1914 assert self.handles is not None  
    1915 f = self.handles.handle
```

File D:\anaconda\envs\project2\Lib\site-packages\pandas\io\common.py:926, in get_handle(path_or_buf, mode, encoding, compression, memory_map, is_text, errors, storage_options)

```
    921 elif isinstance(handle, str):  
    922     # Check whether the filename is to be opened in binary mode.
```

```

923     # Binary mode does not support 'encoding' and 'newline'.
924     if ioargs.encoding and "b" not in ioargs.mode:
925         # Encoding
--> 926         handle = open(
927             handle,
928             ioargs.mode,
929             encoding=ioargs.encoding,
930             errors=errors,
931             newline= ,
932         )
933     else:
934         # Binary mode
935         handle = open(handle, ioargs.mode)

```

FileNotFoundError: [Errno 2] No such file or directory: 'Online_Learning_Behavior_Dataset_Worldwide'

```

In [54]: import pandas as pd
df = pd.read_csv('Online_Learning_Behavior_Dataset_Worldwide.csv')
df.head()

```

Out[54]:

	Student_ID	Country	Age	Gender	Education_Level	Field_of_Study	Platform_Used	Device_Used	Learning_Mode	Enrollment_Date	Daily_Learning_Hour
0	1	South Africa	21	Other	High School	Medicine	Khan Academy	Mobile	Self-Paced	2023-11-24	4.5
1	2	South Africa	49	Female	Undergraduate	Arts	Udacity	Tablet	Self-Paced	2023-02-17	3.2
2	3	UK	20	Female	High School	Engineering	edX	Tablet	Self-Paced	2023-03-23	2.3
3	4	France	38	Other	Undergraduate	Data Science	Coursera	Mobile	Hybrid	2023-08-24	5.8
4	5	France	44	Other	Postgraduate	Business	edX	Tablet	Self-Paced	2023-04-27	5.9

```

In [110... df.isnull().sum()

```

```
Out[110...] Student_ID      0
            Country      0
            Age          0
            Gender      0
            Education_Level  0
            Field_of_Study  0
            Platform_Used  0
            Device_Used   0
            Learning_Mode  0
            Enrollment_Date 0
            Daily_Learning_Hours 0
            Quizzes_Attempted 0
            Assignments_Submitted 0
            Course_Completion_Rate(%) 0
            Satisfaction_Score(1-5) 0
            dtype: int64
```

```
In [111]: df.loc[4]
```

```
Out[111:] Student_ID      5
            Country      France
            Age          44
            Gender      Other
            Education_Level  Postgraduate
            Field_of_Study  Business
            Platform_Used  edX
            Device_Used   Tablet
            Learning_Mode  Self-Paced
            Enrollment_Date 2023-04-27
            Daily_Learning_Hours 5.92
            Quizzes_Attempted 7
            Assignments_Submitted 1
            Course_Completion_Rate(%) 81.85
            Satisfaction_Score(1-5) 4
            Name: 4, dtype: object
```

```
In [57]: df[(df["Age"] < 30 ) & (df["Education_Level"] == ["Postgraduate"])] ]
```

```

ValueError                                Traceback (most recent call last)
Cell In[57], line 1
----> 1 df[(df["Age"] < 30 ) & (df[          ] == [          ] ) ]

File D:\anaconda\envs\project2\Lib\site-packages\pandas\core\ops\common.py:85, in _unpack_zerodim_and_defer.<locals>.new_method(self, other)
    82     other = sanitize_array(other, None)
    83     other = ensure_wrapped_if_datetimelike(other)
--> 85 return method(self, other)

File D:\anaconda\envs\project2\Lib\site-packages\pandas\core\arraylike.py:42, in OpsMixin.__eq__(self, other)
    40 @unpack_zerodim_and_defer("__eq__")
    41 def __eq__(self, other):
--> 42     return self._cmp_method(other, operator.eq)

File D:\anaconda\envs\project2\Lib\site-packages\pandas\core\series.py:6735, in Series._cmp_method(self, other, op)
   6732 lvalues = self._values
   6733 rvalues = extract_array(other, extract_numpy=True, extract_range=True)
-> 6735 res_values = ops.comparison_op(lvalues, rvalues, op)
   6737 return self._construct_result(res_values, name=res_name, other=other)

File D:\anaconda\envs\project2\Lib\site-packages\pandas\core\ops\array_ops.py:332, in comparison_op(left, right, op)
   327 if isinstance(rvalues, (np.ndarray, ABCExtensionArray)) and not rvalues_is_zerodim:
   328     # TODO: make this treatment consistent across ops and classes.
   329     # We are not catching all listlikes here (e.g. frozenset, tuple)
   330     # The ambiguous case is object-dtype. See GH#27803
   331     if len(lvalues) != len(rvalues):
--> 332         raise ValueError(
   333             "Lengths must match to compare", lvalues.shape, rvalues.shape
   334         )
   336 if should_extension_dispatch(lvalues, rvalues) or (
   337     (isinstance(rvalues, (Timedelta, BaseOffset, Timestamp)) or right is NaT)
   338     and lvalues.dtype != object
   339 ):
   340     # Call the method on lvalues
   341     res_values = op(lvalues, rvalues)

ValueError: ('Lengths must match to compare', (1000,), (1,))

```

```
In [58]: df[(df["Age"] < 30 ) & (df["Education_Level"] == "Postgraduate") ]
```

Out[58]:

	Student_ID	Country	Age	Gender	Education_Level	Field_of_Study	Platform_Used	Device_Used	Learning_Mode	Enrollment_Date	Daily_Learning_Hours
	7	8	France	23	Male	Postgraduate	Data Science	Udacity	Desktop	Hybrid	2023-12-02
	29	30	Pakistan	16	Male	Postgraduate	Arts	YouTube Learning	Laptop	Instructor-Led	2023-07-28
	57	58	USA	18	Other	Postgraduate	Data Science	Udemy	Desktop	Self-Paced	2023-07-04
	66	67	Brazil	21	Male	Postgraduate	Arts	YouTube Learning	Desktop	Instructor-Led	2023-09-09
	70	71	Brazil	24	Female	Postgraduate	Medicine	YouTube Learning	Tablet	Instructor-Led	2023-07-28
	...	...	...	...	...	...	...	...	...	...	...
	975	976	Germany	26	Female	Postgraduate	Data Science	Khan Academy	Tablet	Self-Paced	2023-05-23
	978	979	China	23	Female	Postgraduate	Engineering	Udacity	Desktop	Hybrid	2023-04-22
	988	989	Pakistan	24	Female	Postgraduate	Arts	YouTube Learning	Desktop	Self-Paced	2023-04-19
	994	995	UAE	16	Other	Postgraduate	Business	edX	Tablet	Instructor-Led	2023-11-03
	998	999	Pakistan	22	Male	Postgraduate	Engineering	Udacity	Laptop	Hybrid	2023-07-25

104 rows × 15 columns



```
In [59]: import numpy as np
import pandas as pd from matplotlib
import seaborn pyplot as plt as sns
```

Cell In[59], line 2
import pandas as pd from matplotlib
 ^
SyntaxError: invalid syntax

```
In [60]: import numpy as np
import pandas as pd from matplotlib import
import seaborn pyplot as plt as sns
```

Cell In[60], line 2
import pandas as pd from matplotlib import
 ^
SyntaxError: invalid syntax

```
In [61]: import pandas as pd
from matplotlib import pyplot as plt
```

```
import seaborn pyplot as plt as sns
```

```
Cell In[61], line 3
import seaborn pyplot as plt as sns
      ^
SyntaxError: invalid syntax
```

```
In [62]: import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
```

```
In [63]: df.describe()
```

Out[63]:

	Student_ID	Age	Daily_Learning_Hours	Quizzes_Attempted	Assignments_Submitted	Course_Completion_Rate(%)	Satisfaction_Score(1-5)
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	500.500000	32.734000	3.188840	10.12500	5.06200	60.01732	2.903000
std	288.819436	10.456509	1.581955	6.15022	3.17872	23.25666	1.413361
min	1.000000	15.000000	0.510000	0.00000	0.00000	20.00000	1.000000
25%	250.750000	23.000000	1.810000	5.00000	2.00000	39.82750	2.000000
50%	500.500000	33.000000	3.115000	10.00000	5.00000	59.90000	3.000000
75%	750.250000	42.000000	4.560000	16.00000	8.00000	80.88000	4.000000
max	1000.000000	50.000000	6.000000	20.00000	10.00000	99.90000	5.000000

```
In [64]: gender_counts = df['Gender'].value_counts()
print ("Number of Males:", gender_counts[1])
print ("Number of Females:", gender_counts[2])
print ("Number of others:", gender_counts[3])
```

```
-----
KeyError                                Traceback (most recent call last)
File D:\anaconda\envs\project2\Lib\site-packages\pandas\core\indexes\base.py:3641, in Index.get_loc(self, key)
    3640 try:
-> 3641     return self._engine.get_loc(casted_key)
    3642 except KeyError as err:
```

```
File pandas/_libs/index.pyx:168, in pandas._libs.index.IndexEngine.get_loc()
```

```
File pandas/_libs/index.pyx:176, in pandas._libs.index.IndexEngine.get_loc()
```

```
File pandas/_libs/index.pyx:583, in pandas._libs.index.StringObjectEngine._check_type()
```

KeyError: 1

The above exception was the direct cause of the following exception:

```
KeyError                                Traceback (most recent call last)
Cell In[64], line 2
      1 gender_counts = df['Gender'].value_counts()
----> 2 print ("Number of Males:", gender_counts[1])
      3 print ("Number of Females:", gender_counts[2])
      4 print ("Number of others:", gender_counts[3])

File D:\anaconda\envs\project2\Lib\site-packages\pandas\core\series.py:959, in Series.__getitem__(self, key)
    954     key = unpack_1tuple(key)
    956 elif key_is_scalar:
    957     # Note: GH#50617 in 3.0 we changed int key to always be treated as
    958     # a label, matching DataFrame behavior.
-> 959     return self._get_value(key)
    961 # Convert generator to list before going through hashable part
    962 # (We will iterate through the generator there to check for slices)
    963 if is_iterator(key):

File D:\anaconda\envs\project2\Lib\site-packages\pandas\core\series.py:1046, in Series._get_value(self, label, takeable)
    1043     return self._values[label]
    1045 # Similar to Index.get_value, but we do not fall back to positional
-> 1046 loc = self.index.get_loc(label)
    1048 if is_integer(loc):
    1049     return self._values[loc]

File D:\anaconda\envs\project2\Lib\site-packages\pandas\core\indexes\base.py:3648, in Index.get_loc(self, key)
    3643     if isinstance(casted_key, slice) or (
    3644         isinstance(casted_key, abc.Iterable)
    3645         and any(isinstance(x, slice) for x in casted_key)
    3646     ):
    3647         raise InvalidIndexError(key) from err
-> 3648     raise KeyError(key) from err
    3649 except TypeError:
    3650     # If we have a listlike key, _check_indexing_error will raise
    3651     # InvalidIndexError. Otherwise we fall through and re-raise
```

```
3652 # the TypeError.  
3653 self._check_indexing_error(key)
```

KeyError: 1

```
In [67]: gender_counts = df['Gender'].value_counts()  
  
print("Number of Males:", gender_counts['Male'])  
print("Number of Females:", gender_counts['Female'])  
print("Number of Others:", gender_counts['Other'])
```

Number of Males: 317
Number of Females: 346
Number of Others: 337

```
In [66]: plt.figure(figsize=(8,5))
```

```
Cell In[66], line 1  
plt.figure(figsize=(8,5)  
^
```

_IncompleteInputError: incomplete input

```
In [65]: plt.figure(figsize=(8,5))  
gender_counts.plot(Kind='bar',color=['blue','yellow','green'])  
plt.title('Number of each Gender in the online learning')  
plt.xlabel('Gender')  
plt.ylabel('count')  
plt.grid(axis='y')  
plt.show()
```

```
Cell In[65], line 7  
plt.show()  
^
```

IndentationError: unexpected indent

```
In [16]: plt.figure(figsize=(8,5))  
gender_counts.plot(Kind='bar',color=['blue','yellow','green'])  
plt.xticks(rotation=0)  
plt.title('Number of each Gender in the online learning')  
plt.xlabel('Gender')  
plt.ylabel('count')  
plt.grid(axis='y')  
plt.show()
```

NameError Traceback (most recent call last)

Cell In[16], line 2

```
1 plt.figure(figsize=(8,5))
----> 2 gender_counts.plot(kind='bar',color=['blue','yellow','green'])
3 plt.xticks(rotation=0)
4 plt.title('Number of each Gender in the online learning')
```

NameError: name 'gender_counts' is not defined

<Figure size 800x500 with 0 Axes>

```
In [68]: plt.figure(figsize=(8,5))
gender_counts = df['Gender'].value_counts()
gender_counts.plot(kind='bar',color=['blue','yellow','green'])
plt.xticks(rotation=0)
plt.title('Number of each Gender in the online learning')
plt.xlabel('Gender')
plt.ylabel('count')
plt.grid(axis='y')
plt.show()
```

AttributeError Traceback (most recent call last)

Cell In[68], line 3

```
1 plt.figure(figsize=(8,5))
2 gender_counts = df['Gender'].value_counts()
----> 3 gender_counts.plot(Kind=      ,color=[      ,      ,      ])
4 plt.xticks(rotation=0)
5 plt.title('Number of each Gender in the online learning')
```

File D:\anaconda\envs\project2\Lib\site-packages\pandas\plotting_core.py:1185, in PlotAccessor.__call__(self, *args, **kwargs)

```
1182         label_name = label_kw or data.columns
1183         data.columns = label_name
-> 1185 return plot_backend.plot(data, kind=kind, **kwargs)
```

File D:\anaconda\envs\project2\Lib\site-packages\pandas\plotting_matplotlib__init__.py:71, in plot(data, kind, **kwargs)

```
69         kwargs["ax"] = getattr(ax, "left_ax", ax)
70 plot_obj = PLOT_CLASSES[kind](data, **kwargs)
--> 71 plot_obj.generate()
72 plt.draw_if_interactive()
73 return plot_obj.result
```

File D:\anaconda\envs\project2\Lib\site-packages\pandas\plotting_matplotlib\core.py:518, in MPLPlot.generate(self)

```
516 self._compute_plot_data()
517 fig = self.fig
--> 518 self._make_plot(fig)
519 self._add_table()
520 self._make_legend()
```

File D:\anaconda\envs\project2\Lib\site-packages\pandas\plotting_matplotlib\core.py:1596, in LinePlot._make_plot(self, fig)

```
1593 label = self._mark_right_label(label, index=i)
1594 kwds["label"] = label
-> 1596 newlines = plotf(
1597     ax,
1598     x,
1599     y,
1600     style=style,
1601     column_num=i,
1602     stacking_id=stacking_id,
1603     is_errorbar=is_errorbar,
1604     **kwds,
1605 )
1606 self._append_legend_handles_labels(newlines[0], label)
1608 if self._is_ts_plot():
1609     # reset of xlim should be used for ts data
1610     # TODO: GH28021, should find a way to change view limit on axis
```

File D:\anaconda\envs\project2\Lib\site-packages\pandas\plotting_matplotlib\core.py:1632, in LinePlot._plot(cls, ax, x, y, style, column_num, stacking_id, **kwds)

```
1630     cls._initialize_stacker(ax, stacking_id, len(y))
1631 y_values = cls._get_stacked_values(ax, stacking_id, y, kwds["label"])
-> 1632 lines = MPLPlot._plot(ax, x, y_values, style=style, **kwds)
```

```
1633 cls._update_stacker(ax, stacking_id, y)
1634 return lines
```

```
File D:\anaconda\envs\project2\Lib\site-packages\pandas\plotting\_matplotlib\converter.py:90, in register_pandas_matplotlib_converters.<locals>
>.wrapper(*args, **kwargs)
    87 @functools.wraps(func)
    88 def wrapper(*args, **kwargs):
    89     with pandas_converters():
--> 90         return func(*args, **kwargs)
```

```
File D:\anaconda\envs\project2\Lib\site-packages\pandas\plotting\_matplotlib\core.py:997, in MPLPlot._plot(cls, ax, x, y, style, is_errorbar,
**kwds)
    994 else:
    995     # prevent style kwarg from going to errorbar, where it is unsupported
    996     args = (x, y, style) if style is not None else (x, y)
--> 997     return ax.plot(*args, **kwds)
```

```
File D:\anaconda\envs\project2\Lib\site-packages\matplotlib\axes\_axes.py:1777, in Axes.plot(self, scalex, scaley, data, *args, **kwargs)
1534 """
1535 Plot y versus x as lines and/or markers.
1536 (...) 1774 (``'green'``) or hex strings (``'#008000'``).
1775 """
1776 kwargs = cbook.normalize_kwargs(kwargs, mlines.Line2D)
-> 1777 lines = [*self._get_lines(self, *args, data=data, **kwargs)]
1778 for line in lines:
1779     self.add_line(line)
```

```
File D:\anaconda\envs\project2\Lib\site-packages\matplotlib\axes\_base.py:297, in _process_plot_var_args.__call__(self, axes, data, return_kwa
rgs, *args, **kwargs)
    295     this += args[0],
    296     args = args[1:]
--> 297 yield from self._plot_args(
    298     axes, this, kwargs, ambiguous_fmt_datakey=ambiguous_fmt_datakey,
    299     return_kwargs=return_kwargs
    300 )
```

```
File D:\anaconda\envs\project2\Lib\site-packages\matplotlib\axes\_base.py:546, in _process_plot_var_args._plot_args(self, axes, tup, kwargs, r
eturn_kwargs, ambiguous_fmt_datakey)
    544     return list(result)
    545 else:
--> 546     return [l[0] for l in result]
```

```
File D:\anaconda\envs\project2\Lib\site-packages\matplotlib\axes\_base.py:539, in <genexpr>(.0)
    534 else:
    535     raise ValueError(
    536         f"label must be scalar or have the same length as the input "
    537         f"data, but found {len(label)} for {n_datasets} datasets.")
--> 539 result = (make_artist(axes, x[:, j % ncx], y[:, j % ncy], kw,
    540                     {**kwargs, : label}))
    541     for j, label in enumerate(labels))
```

```
543 if return_kwargs:
544     return list(result)
```

File D:\anaconda\envs\project2\Lib\site-packages\matplotlib\axes_base.py:338, in `_process_plot_var_args._make_line(self, axes, x, y, kw, kwargs)`

```
336 kw = {**kw, **kwargs} # Don't modify the original kw.
337 self._setdefaults(self._getdefaults(kw), kw)
-> 338 seg = mlines.Line2D(x, y, **kw)
339 return seg, kw
```

File D:\anaconda\envs\project2\Lib\site-packages\matplotlib\lines.py:421, in `Line2D.__init__(self, xdata, ydata, linewidth, linestyle, color, gapcolor, marker, markersize, markeredgewidth, markeredgewidth, markeredgewidth, markeredgewidth, markerfacecolor, markerfacecoloralt, fillstyle, antialiased, dash_capstyle, solid_capstyle, dash_joinstyle, solid_joinstyle, pickradius, drawstyle, markevery, **kwargs)`

```
417 self.set_markedgewidth(markedgewidth)
419 # update kwargs before updating data to give the caller a
420 # chance to init axes (and hence unit support)
-> 421 self._internal_update(kwargs)
422 self.pickradius = pickradius
423 self.ind_offset = 0
```

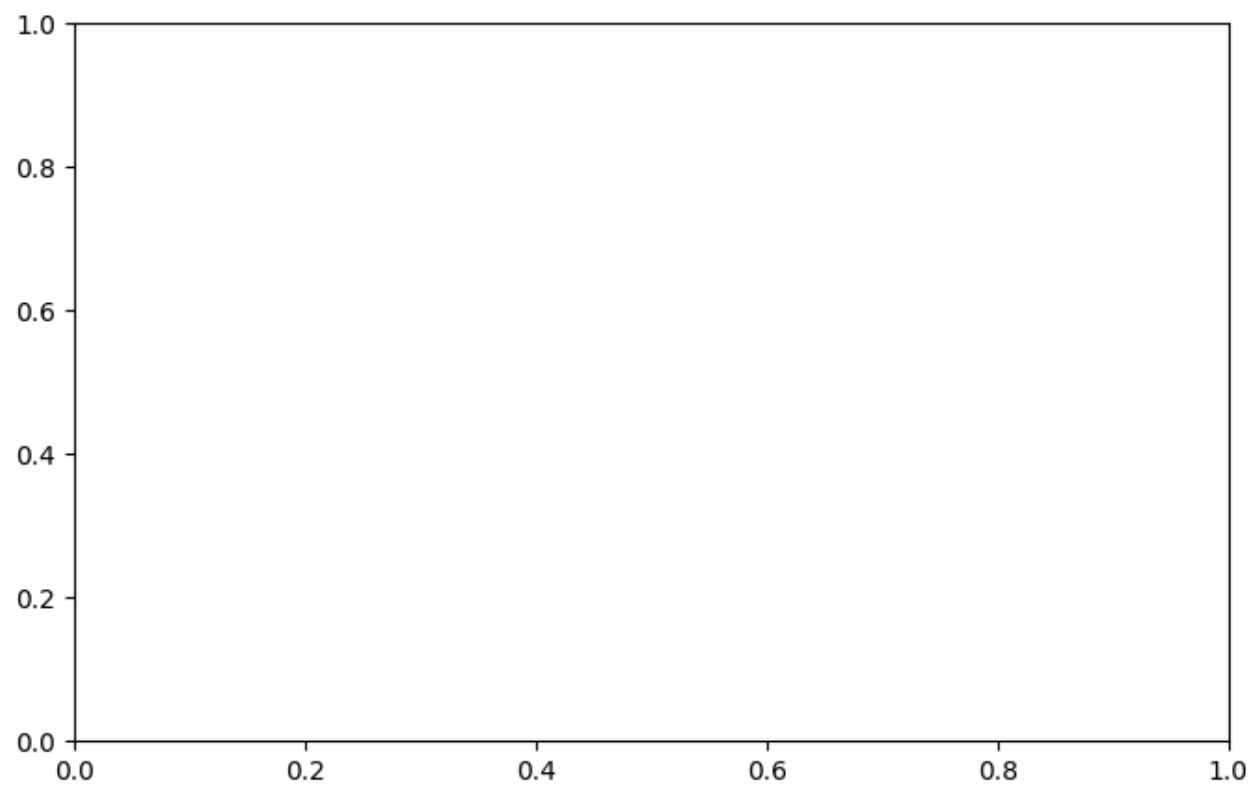
File D:\anaconda\envs\project2\Lib\site-packages\matplotlib\artist.py:1233, in `Artist._internal_update(self, kwargs)`

```
1226 def _internal_update(self, kwargs):
1227     """
1228     Update artist properties without prenormalizing them, but generating
1229     errors as if calling `set`.
1230
1231     The lack of prenormalization is to maintain backcompatibility.
1232     """
-> 1233     return self._update_props(
1234         kwargs, {cls.__name__}
1235         {prop_name!r} )
```

File D:\anaconda\envs\project2\Lib\site-packages\matplotlib\artist.py:1206, in `Artist._update_props(self, props, errfmt)`

```
1204     func = getattr(self, f"set_{k}", None)
1205     if not callable(func):
-> 1206         raise AttributeError(
1207             errfmt.format(cls=type(self), prop_name=k),
1208             name=k)
1209     ret.append(func(v))
1210 if ret:
```

AttributeError: Line2D.set() got an unexpected keyword argument 'Kind'



```
In [112... gender_counts = df['Gender'].value_counts()

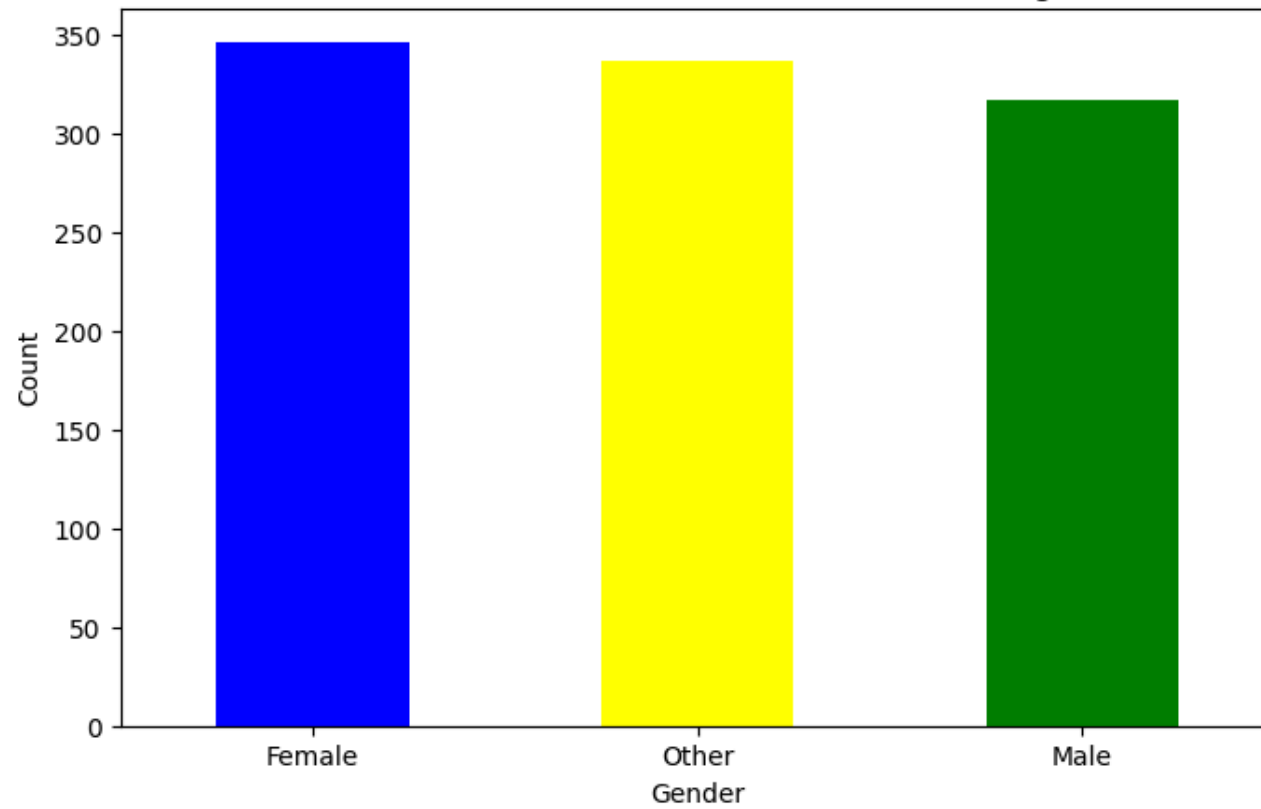
plt.figure(figsize=(8,5))
gender_counts.plot(kind='bar', color=['blue','yellow','green'])

plt.title("Number of each Gender in the online learning")
plt.xlabel("Gender")
plt.ylabel("Count")

plt.xticks(rotation=0)

plt.show()
```

Number of each Gender in the online learning



```
In [69]: df = pd.read_csv("Online_Learning_Behavior_Dataset_Worldwide.csv")
```

```
correlation_matrix = df.corr(numeric_only=True)  
print(correlation_matrix)
```

	Student_ID	Age	Daily_Learning_Hours	\
Student_ID	1.000000	-0.013334	-0.011861	
Age	-0.013334	1.000000	0.001679	
Daily_Learning_Hours	-0.011861	0.001679	1.000000	
Quizzes_Attempted	0.003707	0.000860	-0.040299	
Assignments_Submitted	0.001533	-0.008478	-0.038444	
Course_Completion_Rate(%)	-0.042603	-0.012029	0.002566	
Satisfaction_Score(1-5)	-0.025158	-0.004999	-0.012443	

	Quizzes_Attempted	Assignments_Submitted	\
Student_ID	0.003707	0.001533	
Age	0.000860	-0.008478	
Daily_Learning_Hours	-0.040299	-0.038444	
Quizzes_Attempted	1.000000	0.055619	
Assignments_Submitted	0.055619	1.000000	
Course_Completion_Rate(%)	-0.053488	-0.002943	
Satisfaction_Score(1-5)	0.019246	-0.023392	

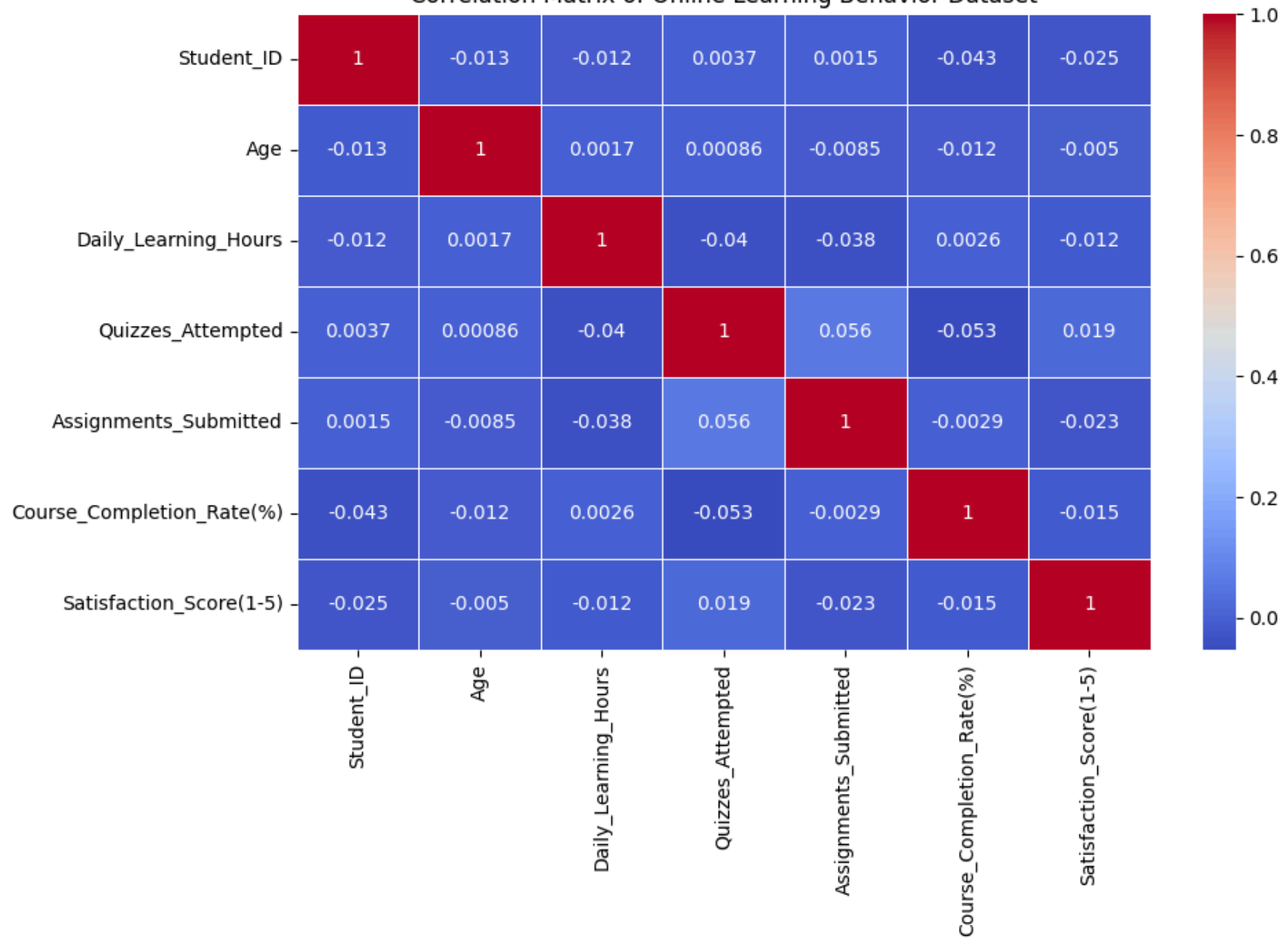
	Course_Completion_Rate(%)	Satisfaction_Score(1-5)
Student_ID	-0.042603	-0.025158
Age	-0.012029	-0.004999
Daily_Learning_Hours	0.002566	-0.012443
Quizzes_Attempted	-0.053488	0.019246
Assignments_Submitted	-0.002943	-0.023392
Course_Completion_Rate(%)	1.000000	-0.014663
Satisfaction_Score(1-5)	-0.014663	1.000000

```
In [70]: plt.figure(figsize=(10,6))

sns.heatmap(correlation_matrix,
            annot=True,
            cmap='coolwarm',
            linewidths=0.5)

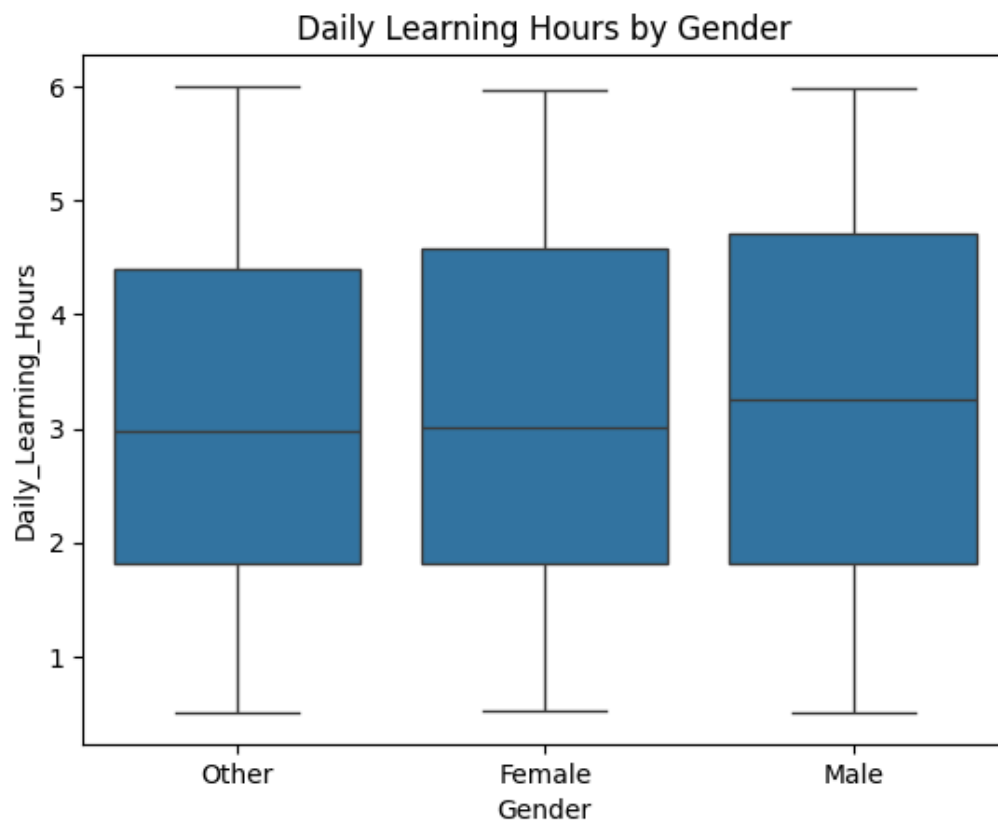
plt.title("Correlation Matrix of Online Learning Behavior Dataset")
plt.show()
```

Correlation Matrix of Online Learning Behavior Dataset



```
In [71]: plt.figure()
sns.boxplot(x='Gender', y='Daily_Learning_Hours', data=df)

plt.title("Daily Learning Hours by Gender")
plt.show()
```



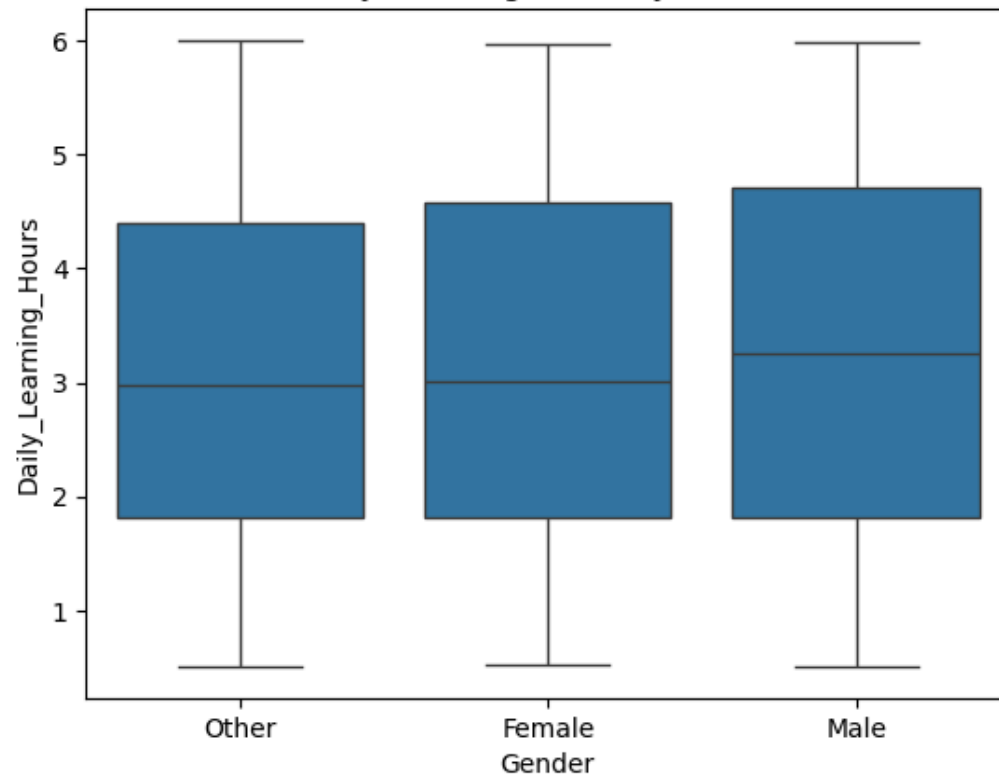
```
In [72]: import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
In [73]: df = pd.read_csv("Online_Learning_Behavior_Dataset_Worldwide.csv")
```

```
In [74]: plt.figure()
sns.boxplot(x='Gender', y='Daily_Learning_Hours', data=df)

plt.title("Daily Learning Hours by Gender")
plt.show()
```

Daily Learning Hours by Gender



```
In [19]: plt.figure()
sns.scatterplot(x='Daily_Learning_Hours', y='Course_Completion_Rate(%)', data=df)

plt.title("Learning Hours vs Course Completion Rate")
plt.show()
```

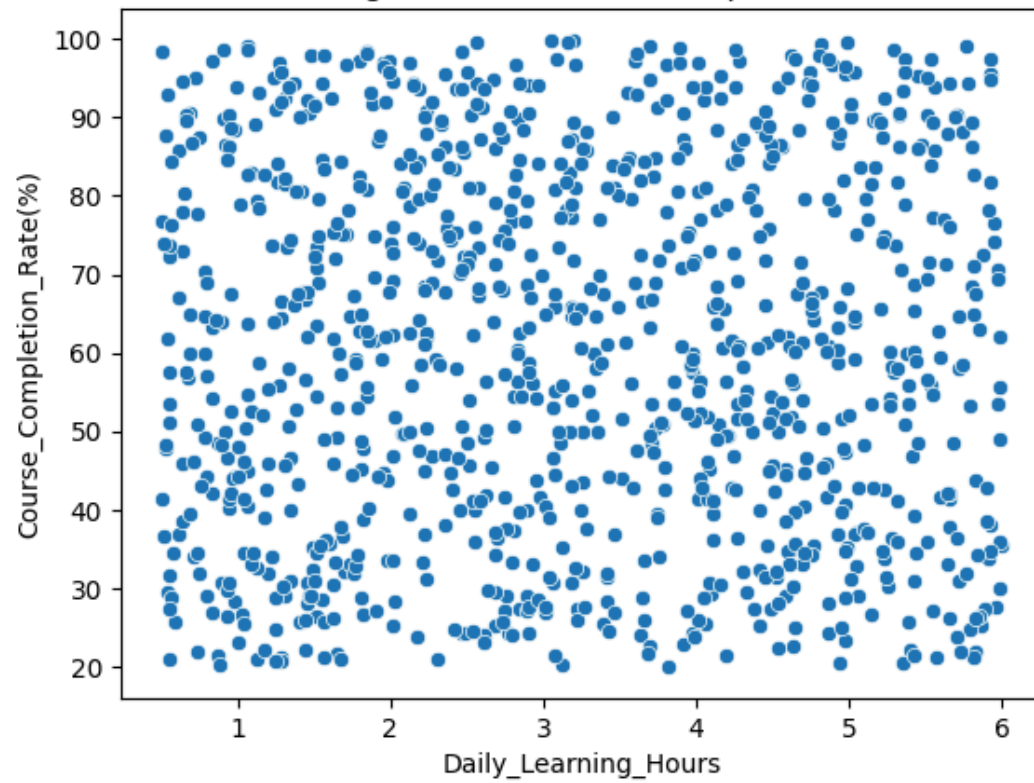
```
-----
AttributeError                                Traceback (most recent call last)
Cell In[19], line 2
      1 plt.figure()
----> 2 sns.scatterplot(x='Daily_Learning_Hours', y='Course_Completion_Rate(%)', data=df)
      4 plt.title("Learning Hours vs Course Completion Rate")
      5 plt.show()

AttributeError: module 'seaborn' has no attribute 'scatterplot'
<Figure size 640x480 with 0 Axes>
```

```
In [75]: plt.figure()
sns.scatterplot(x='Daily_Learning_Hours', y='Course_Completion_Rate(%)', data=df)

plt.title("Learning Hours vs Course Completion Rate")
plt.show()
```

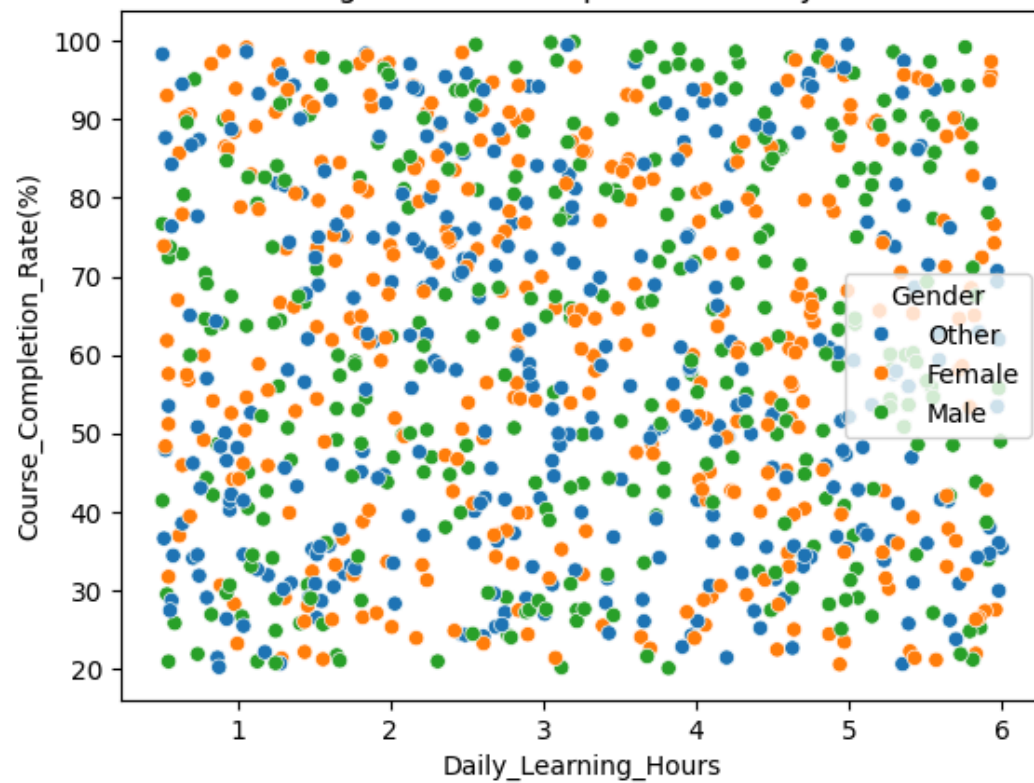
Learning Hours vs Course Completion Rate



```
In [76]: plt.figure()
sns.scatterplot(x='Daily_Learning_Hours',
                y='Course_Completion_Rate(%)',
                hue='Gender',
                data=df)

plt.title("Learning Hours vs Completion Rate by Gender")
plt.show()
```

Learning Hours vs Completion Rate by Gender



```
In [113... from sklearn.preprocessing import StandardScaler
```

```
In [114... import pandas as pd
import numpy as np
from scipy.stats import zscore
```

```
In [115... df = pd.read_csv("Online_Learning_Behavior_Dataset_Worldwide.csv")
```

```
In [116... numeric_columns = [
    'Age',
    'Daily_Learning_Hours',
    'Quizzes_Attempted',
    'Assignments_Submitted',
    'Course_Completion_Rate(%)'
]
```

```
In [117... z_scores = np.abs(zscore(df[numeric_columns]))
```

```
In [118... df_clean = df[(z_scores < 3).all(axis=1)]
```

```
In [119]: X = df[['Age',
                'Daily_Learning_Hours',
                'Quizzes_Attempted',
                'Assignments_Submitted',
                'Course_Completion_Rate(%)']]

y = df['Satisfaction_Score(1-5)']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
In [84]: X = df[['Age',
                'Daily_Learning_Hours',
                'Quizzes_Attempted',
                'Assignments_Submitted',
                'Course_Completion_Rate(%)']]

y = df['Satisfaction_Score(1-5)']
```

```
In [85]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
knn = KNeighborsClassifier(n_neighbors=5)

knn.fit(X_train, y_train)
```

```
Out[85]: 

KNeighborsClassifier ⓘ ?



► Parameters


```

```
In [86]: import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
```

```
In [87]: df = pd.read_csv("Online_Learning_Behavior_Dataset_Worldwide.csv")
```

```
In [88]: X = df[['Age',
                'Daily_Learning_Hours',
                'Quizzes_Attempted',
                'Assignments_Submitted',
                'Course_Completion_Rate(%)']]

y = df['Satisfaction_Score(1-5)']
```

```
In [89]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
In [90]: knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
```

```
Out[90]: KNeighborsClassifier ⓘ ⓘ
  ▶ Parameters
```

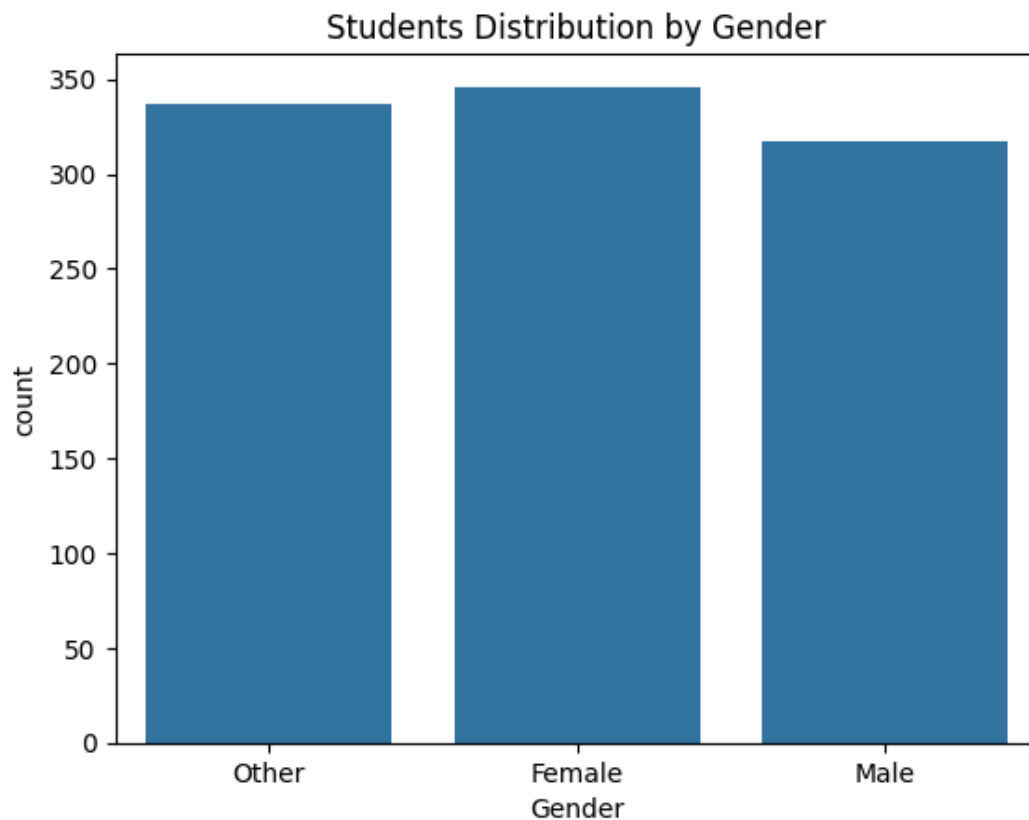
```
In [91]: y_pred = knn.predict(X_test)
```

```
In [92]: accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Accuracy: 0.23

```
In [93]: sns.countplot(x='Gender', data=df)

plt.title("Students Distribution by Gender")
plt.show()
```

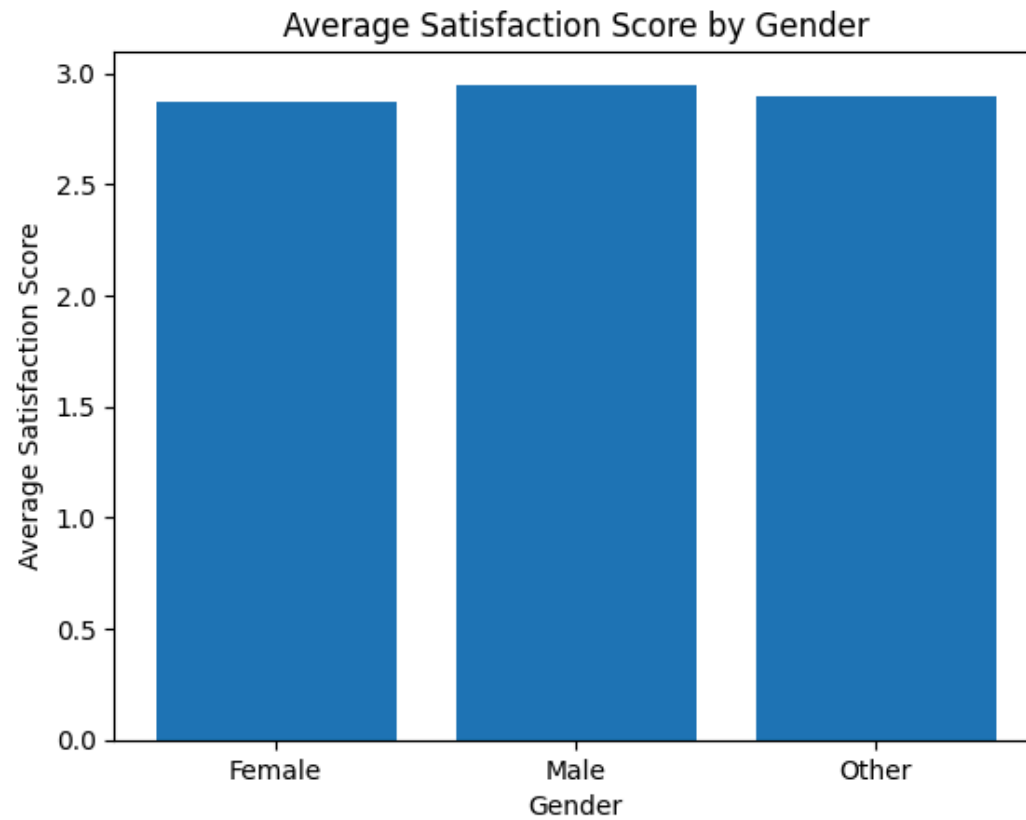


```
In [94]: avg_score = df.groupby('Gender')['Satisfaction_Score(1-5)'].mean()

plt.bar(avg_score.index, avg_score.values)
```

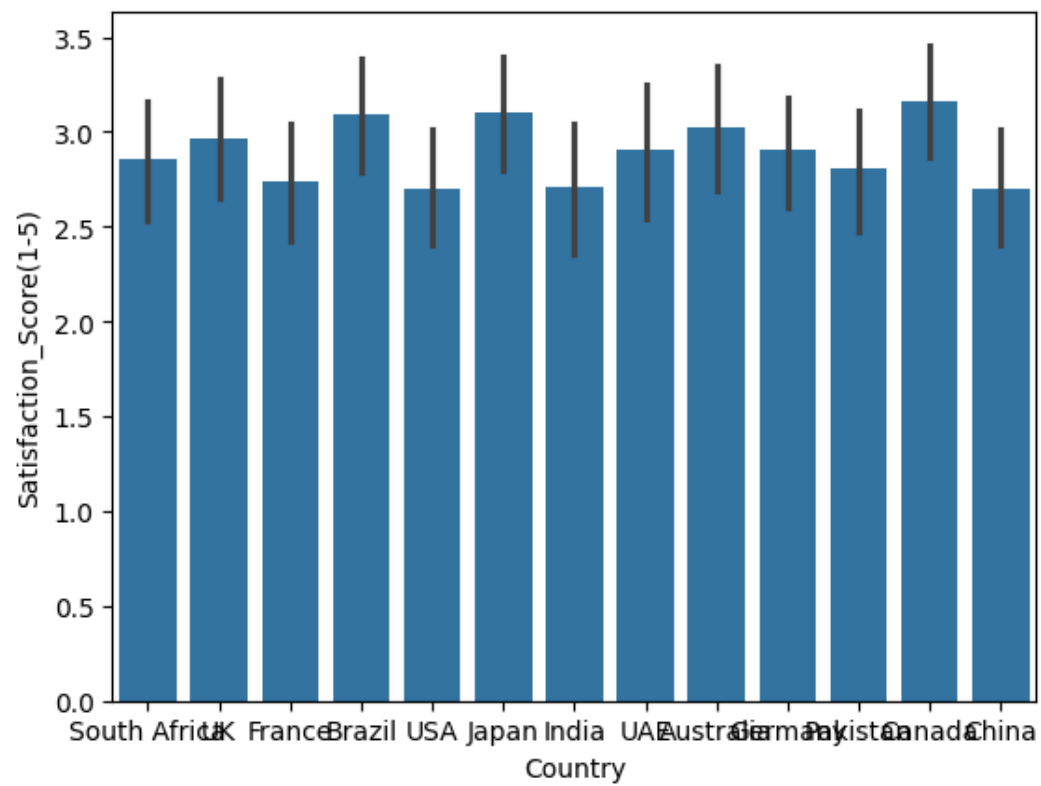
```
plt.title("Average Satisfaction Score by Gender")
plt.xlabel("Gender")
plt.ylabel("Average Satisfaction Score")

plt.show()
```



```
In [120... sns.barplot(x='Country', y='Satisfaction_Score(1-5)', data=df)
```

```
Out[120... <Axes: xlabel='Country', ylabel='Satisfaction_Score(1-5)'>
```



```
In [96]: plt.figure(figsize=(12,6))

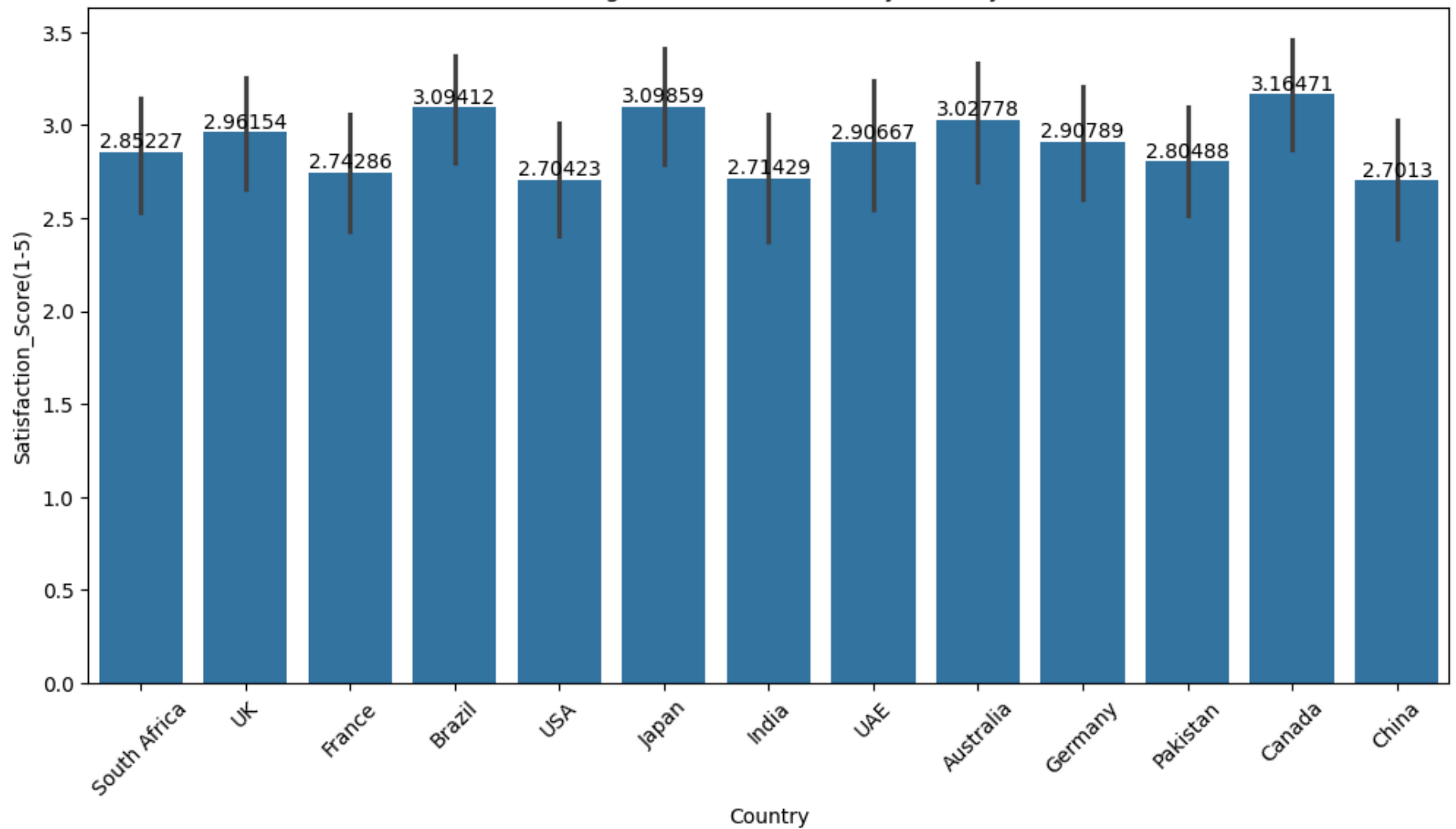
ax = sns.barplot(x='Country', y='Satisfaction_Score(1-5)', data=df)

for i in ax.containers:
    ax.bar_label(i)

plt.xticks(rotation=45)
plt.title("Average Satisfaction Score by Country")

plt.show()
```

Average Satisfaction Score by Country



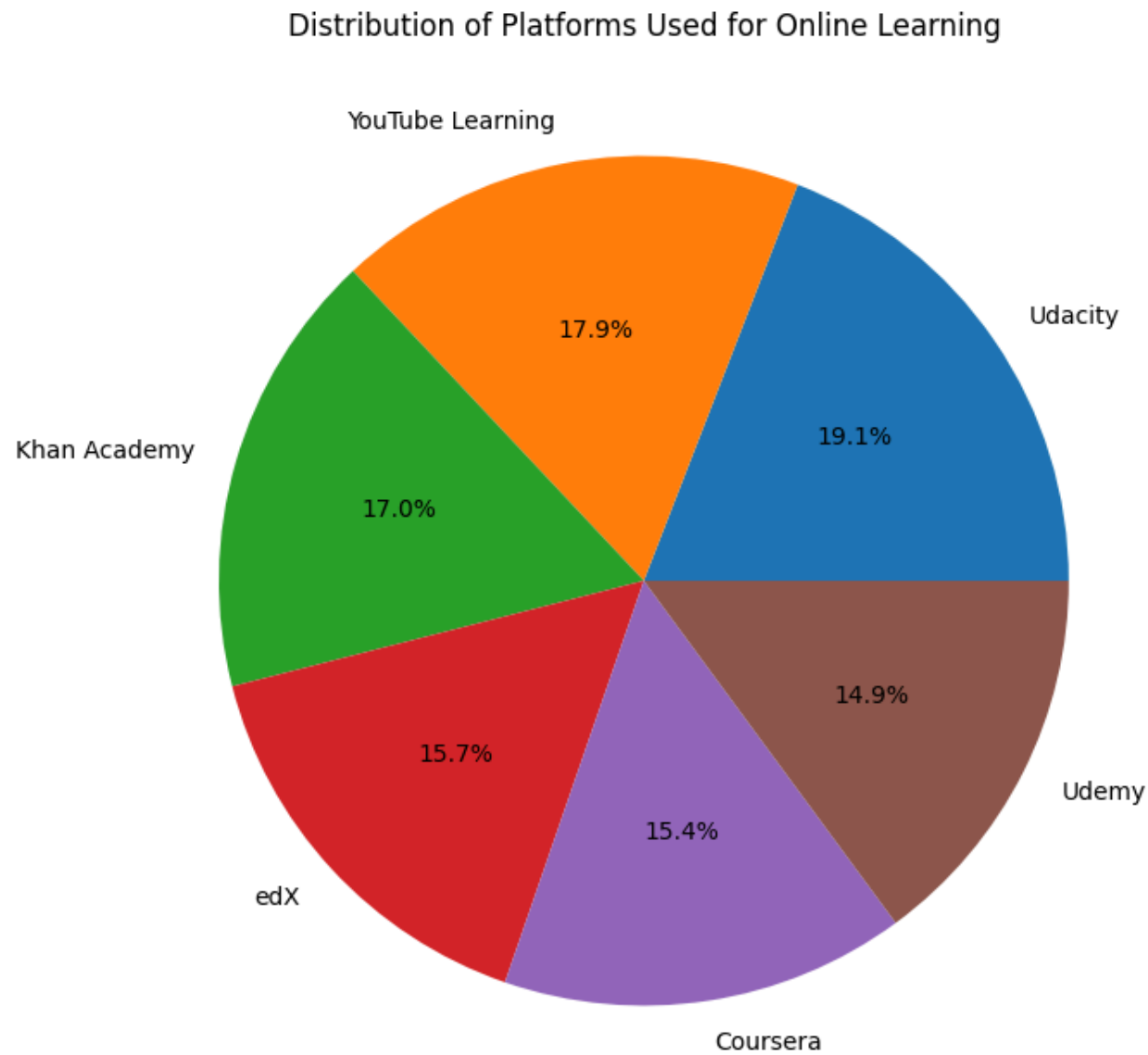
```
In [97]: import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
df = pd.read_csv("Online_Learning_Behavior_Dataset_Worldwide.csv")

# Count occurrences of each platform
platform_counts = df["Platform_Used"].value_counts()

# Create pie chart
plt.figure(figsize=(8,8))
plt.pie(platform_counts,
        labels=platform_counts.index,
        autopct='%1.1f%%')
```

```
plt.title("Distribution of Platforms Used for Online Learning")
plt.show()
```



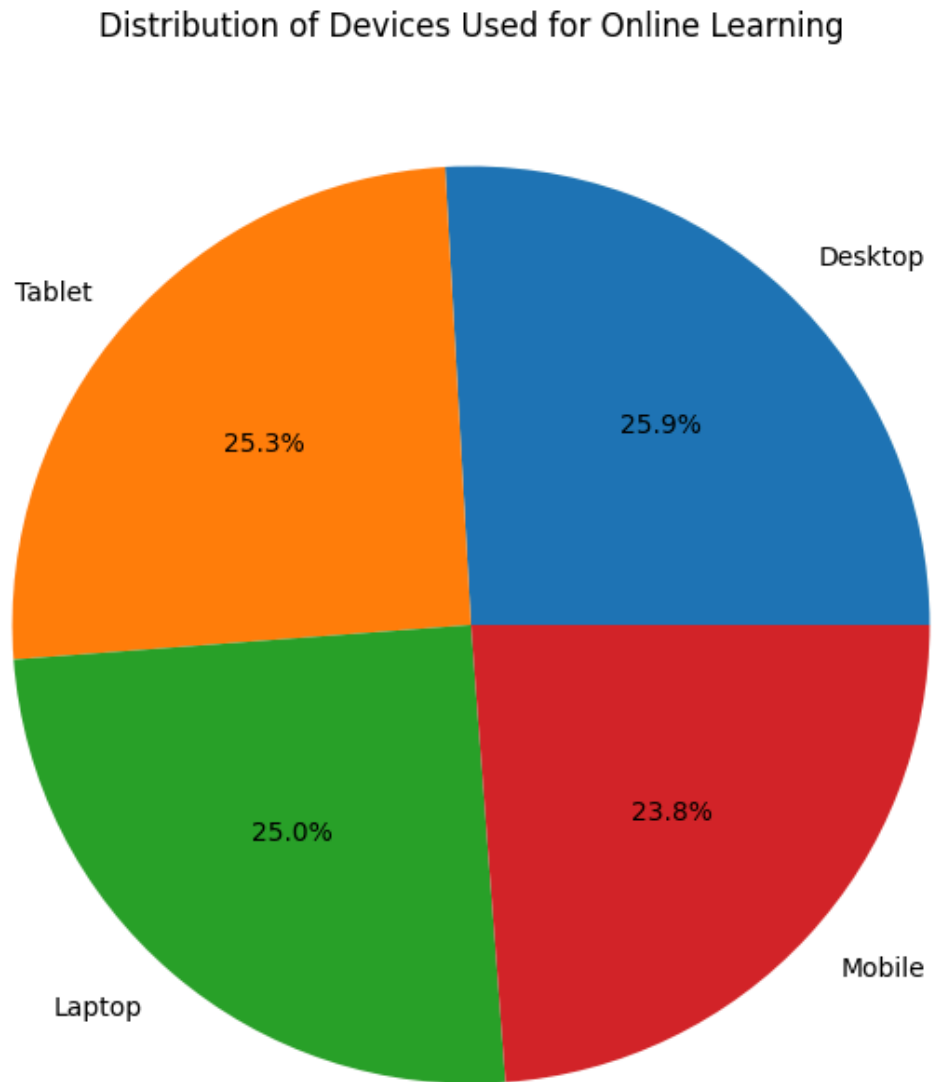
```
In [98]: df = pd.read_csv("Online_Learning_Behavior_Dataset_Worldwide.csv")
```

```
# Count each device used
device_counts = df["Device_Used"].value_counts()
```

```
# Create pie chart
plt.figure(figsize=(8,8))
plt.pie(device_counts.values,
        labels=device_counts.index,
```

```
autopct='%1.1f%%')
```

```
plt.title("Distribution of Devices Used for Online Learning")  
plt.show()
```



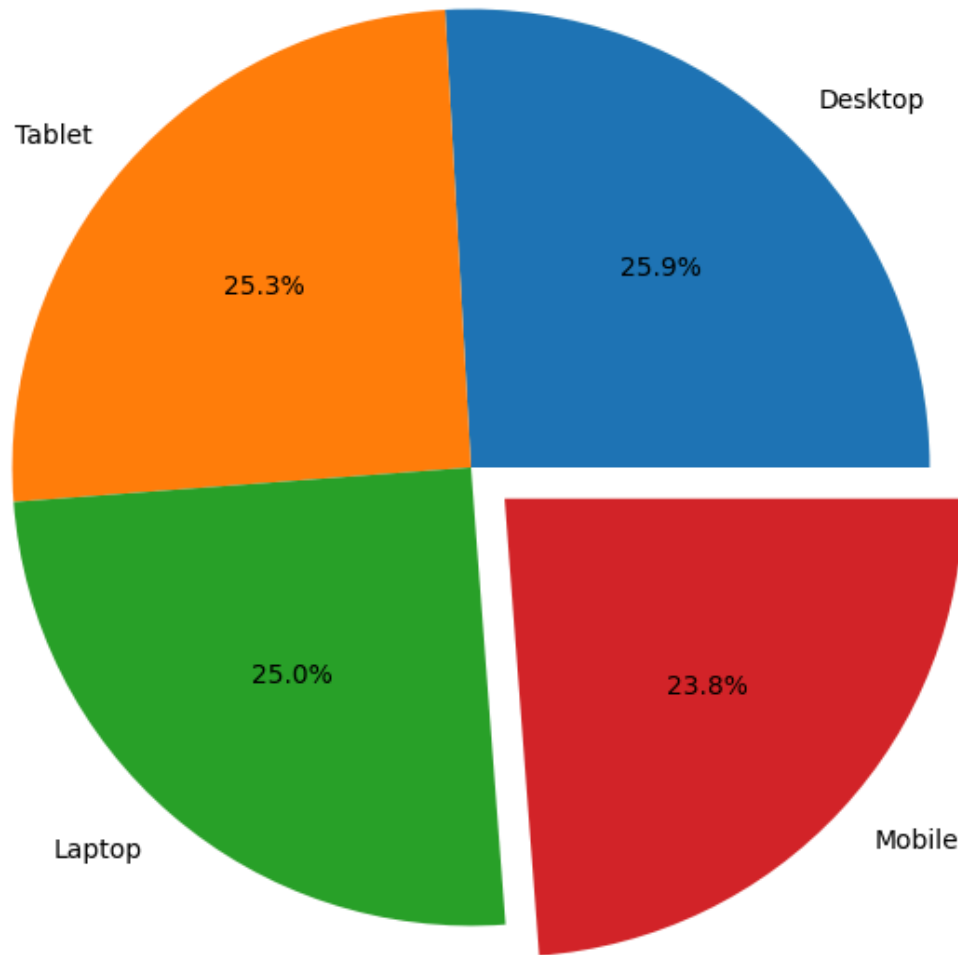
```
In [99]: device_counts = df["Device_Used"].value_counts()  
  
# Create explode list (0 = normal, >0 = separated)  
explode = []  
  
for device in device_counts.index:  
    if device.lower() in ["smartphone", "mobile"]:
```

```
        explode.append(0.1) # separate mobile slice
    else:
        explode.append(0)

# Plot pie chart
plt.figure(figsize=(8,8))
plt.pie(device_counts.values,
        labels=device_counts.index,
        autopct='%1.1f%%',
        explode=explode)

plt.title("Distribution of Devices Used for Online Learning")
plt.show()
```

Distribution of Devices Used for Online Learning



```
In [100... device_counts = df["Device_Used"].value_counts()

# Explode multiple slices
explode = []

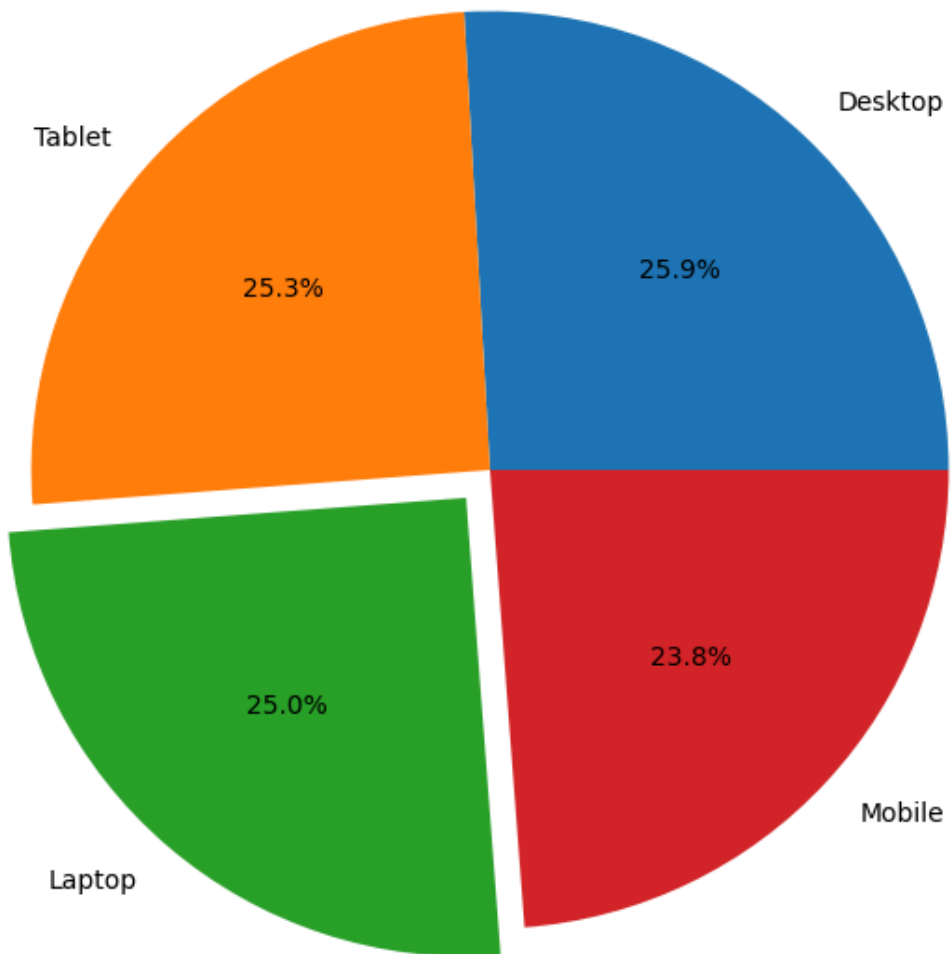
for device in device_counts.index:
    if device.lower() == "smartphone":
        explode.append(0.1) # separate smartphone
    elif device.lower() == "laptop":
        explode.append(0.08) # separate laptop
    else:
```

```
explode.append(0)

# Create pie chart
plt.figure(figsize=(8,8))
plt.pie(device_counts.values,
        labels=device_counts.index,
        autopct='%1.1f%%',
        explode=explode)

plt.title("Distribution of Devices Used for Online Learning")
plt.show()
```

Distribution of Devices Used for Online Learning



In [101...

```
device_counts = df["Device_Used"].value_counts()

# Separate two important slices
explode = []
for device in device_counts.index:
    if device.lower() == "smartphone":
        explode.append(0.08)
    elif device.lower() == "laptop":
        explode.append(0.06)
    else:
        explode.append(0)

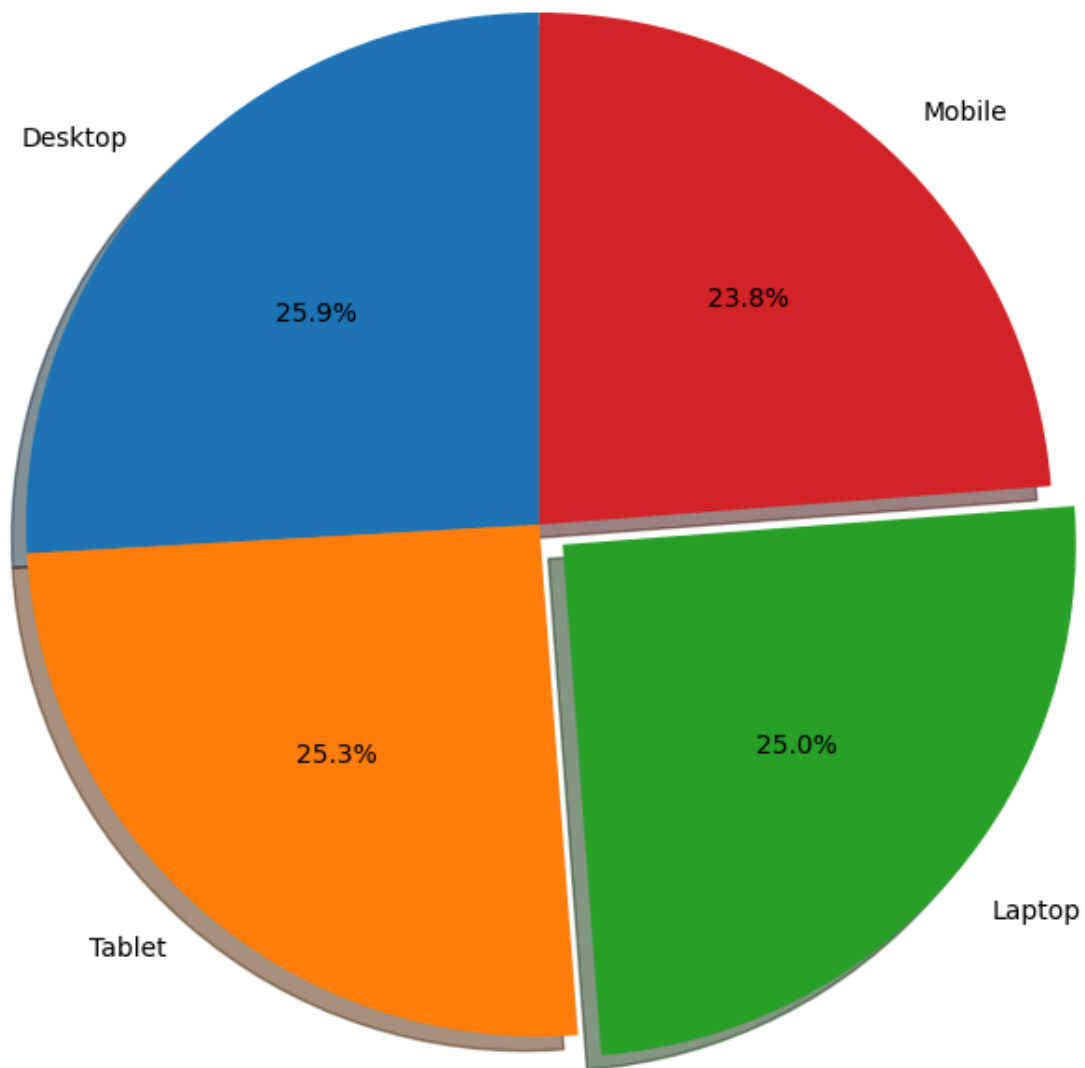
# Create figure
plt.figure(figsize=(8,8))

plt.pie(
    device_counts.values,
    labels=device_counts.index,
    autopct='%1.1f%%',
    explode=explode,
    shadow=True,          #Shadow effect
    startangle=90         #Start angle 90°
)

plt.title("Device Usage for Online Learning", fontsize=14)
plt.axis('equal')      # Makes pie a perfect circle

plt.show()
```

Device Usage for Online Learning



```
In [102... device_counts = df["Device_Used"].value_counts()
```

```
# Create figure with two charts
```

```
fig, ax = plt.subplots(1, 2, figsize=(12,6))
```

```
# ----- Bar Chart -----
```

```
ax[0].bar(device_counts.index, device_counts.values)
```

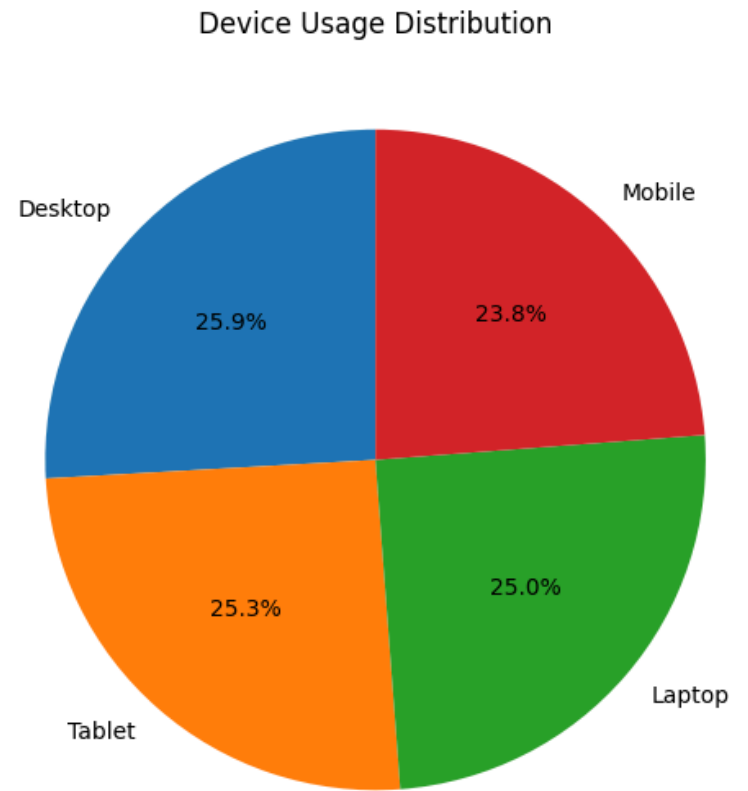
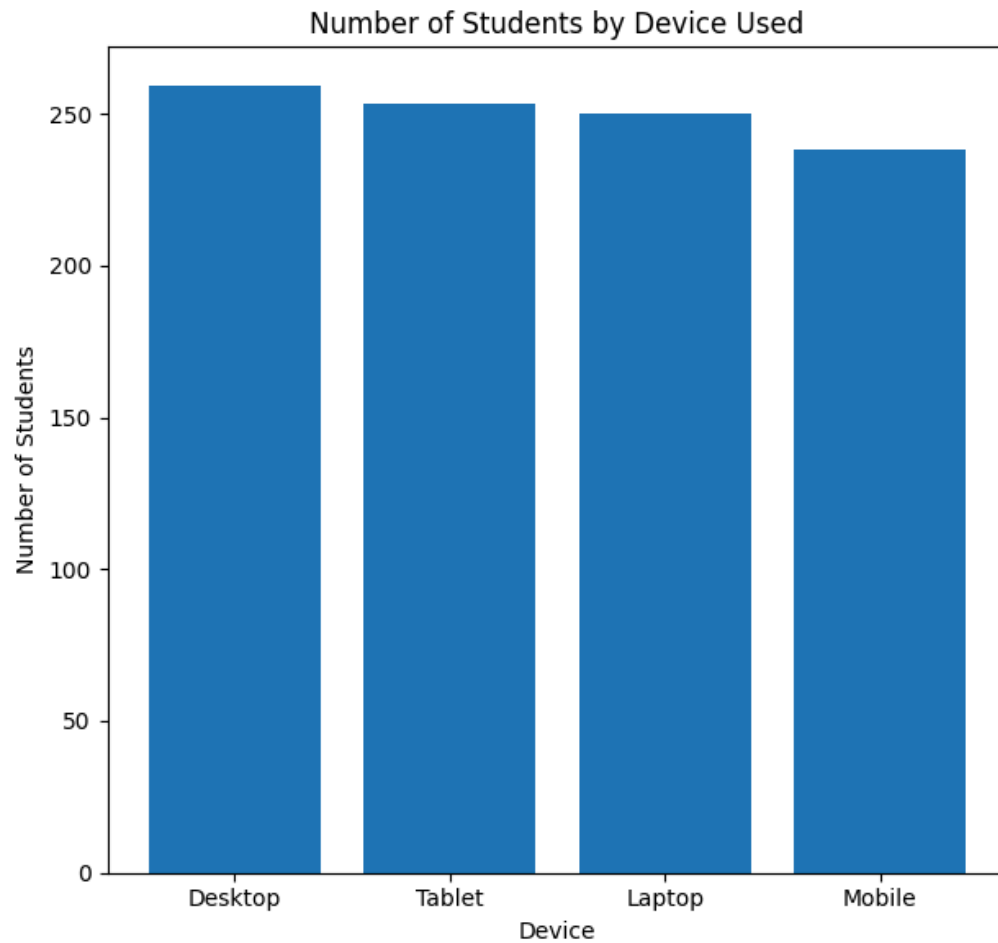
```
ax[0].set_title("Number of Students by Device Used")
```

```
ax[0].set_xlabel("Device")
```

```
ax[0].set_ylabel("Number of Students")
```

```
# ----- Pie Chart -----
ax[1].pie(device_counts.values,
          labels=device_counts.index,
          autopct='%1.1f%%',
          startangle=90)
ax[1].set_title("Device Usage Distribution")

plt.tight_layout()
plt.show()
```



```
In [103... df["Enrollment_Date"] = pd.to_datetime(df["Enrollment_Date"])

# Count enrollments per month
enrollments = df.groupby(df["Enrollment_Date"].dt.to_period("M")).size()

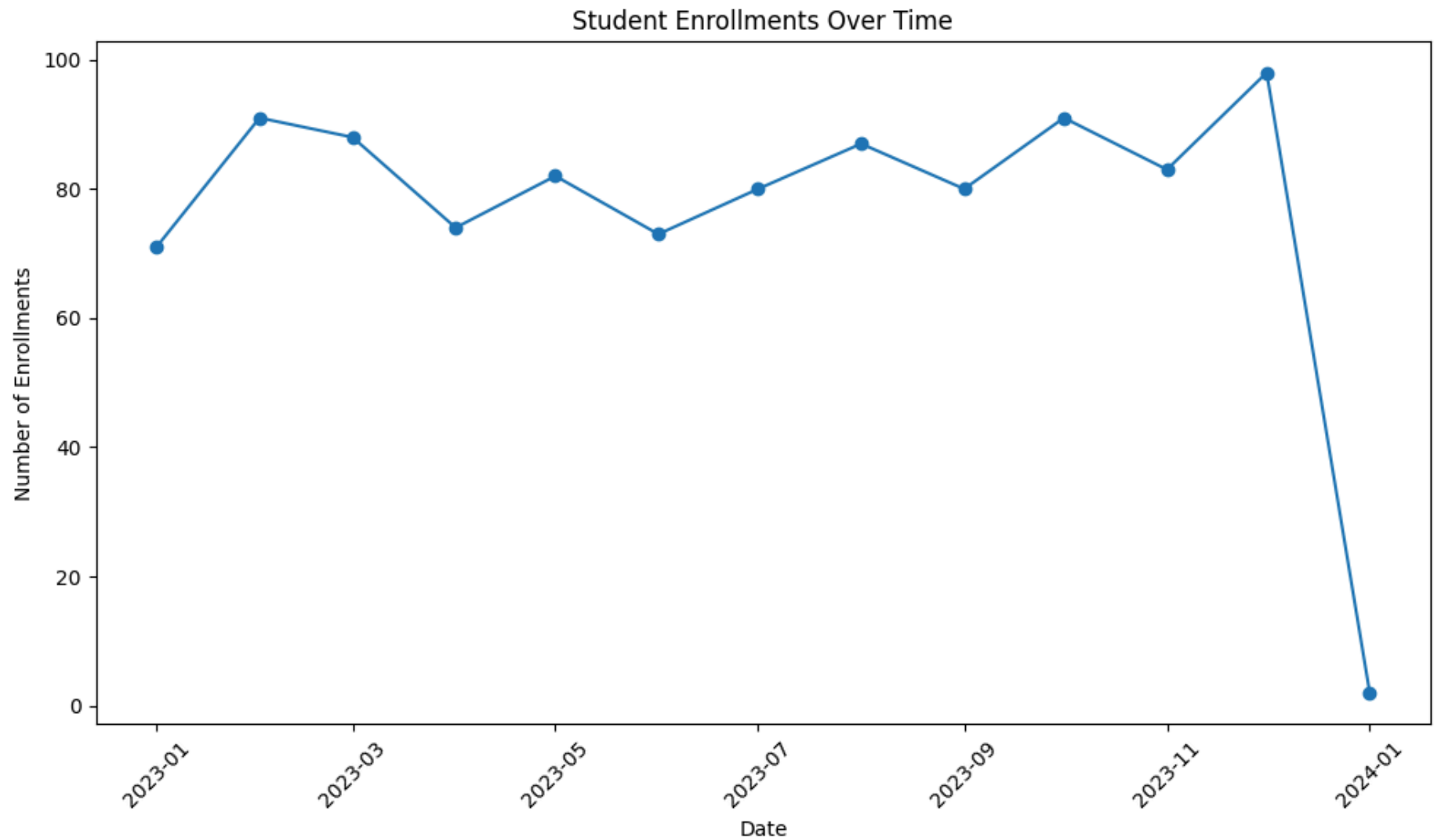
# Convert index to timestamp for plot
enrollments.index = enrollments.index.to_timestamp()

# Plot line chart
plt.figure(figsize=(10,6))
```

```
plt.plot(enrollments.index, enrollments.values, marker='o')

plt.title("Student Enrollments Over Time")
plt.xlabel("Date")
plt.ylabel("Number of Enrollments")
plt.xticks(rotation=45)

plt.tight_layout()
plt.show()
```



In [104...

```
# Convert Enrollment_Date to datetime
df["Enrollment_Date"] = pd.to_datetime(df["Enrollment_Date"])

# Group data by month
monthly_data = df.groupby(df["Enrollment_Date"].dt.to_period("M")).agg({
    "Daily_Learning_Hours": "mean",
```

```
    "Course_Completion_Rate(%)": "mean"
})

# Convert index to timestamp
monthly_data.index = monthly_data.index.to_timestamp()

# ----- Line Chart 1: Learning Hours -----
plt.figure(figsize=(8,5))
plt.plot(monthly_data.index, monthly_data["Daily_Learning_Hours"], marker='o')

plt.title("Average Daily Learning Hours Over Time")
plt.xlabel("Date")
plt.ylabel("Average Learning Hours")

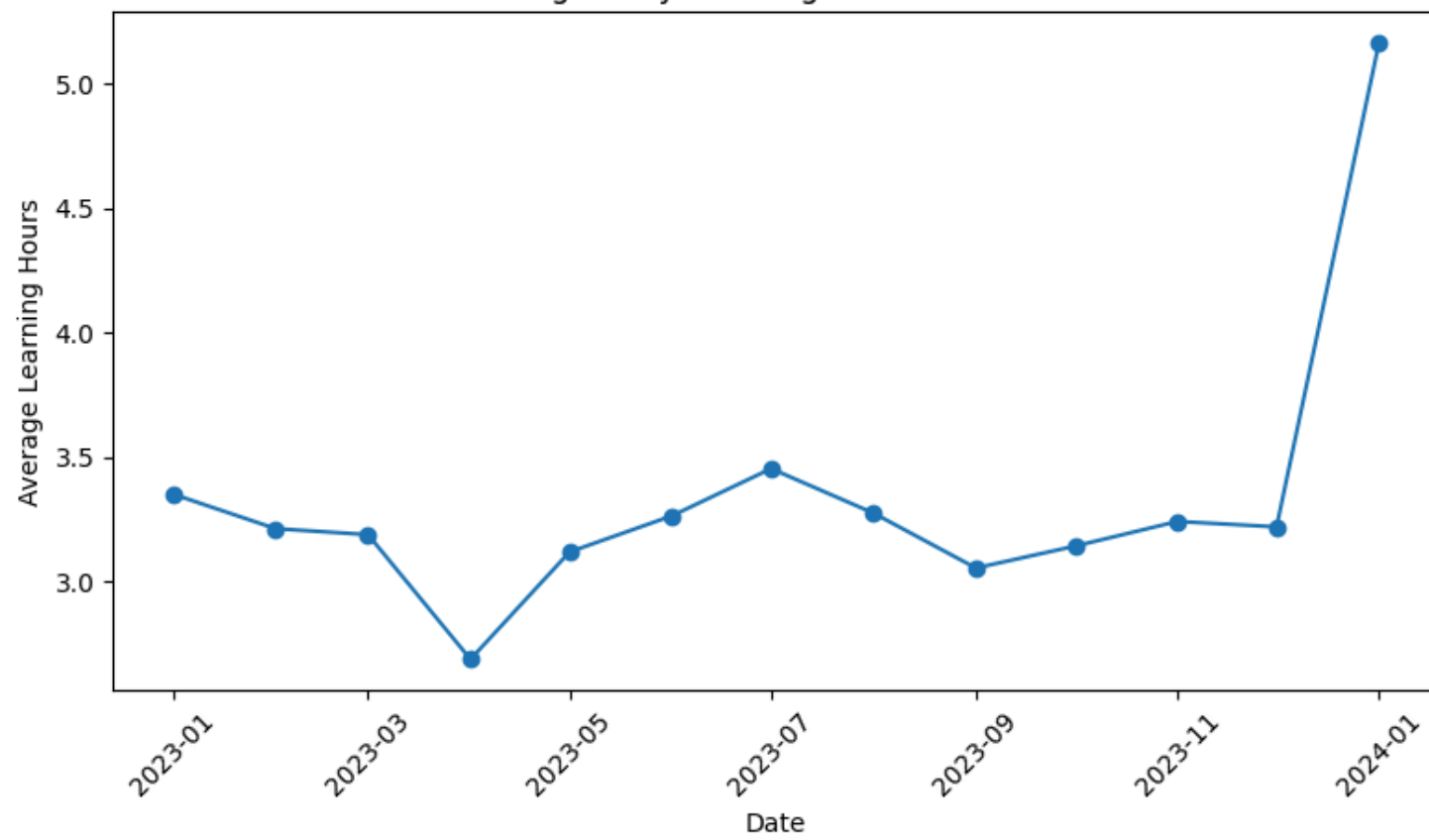
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

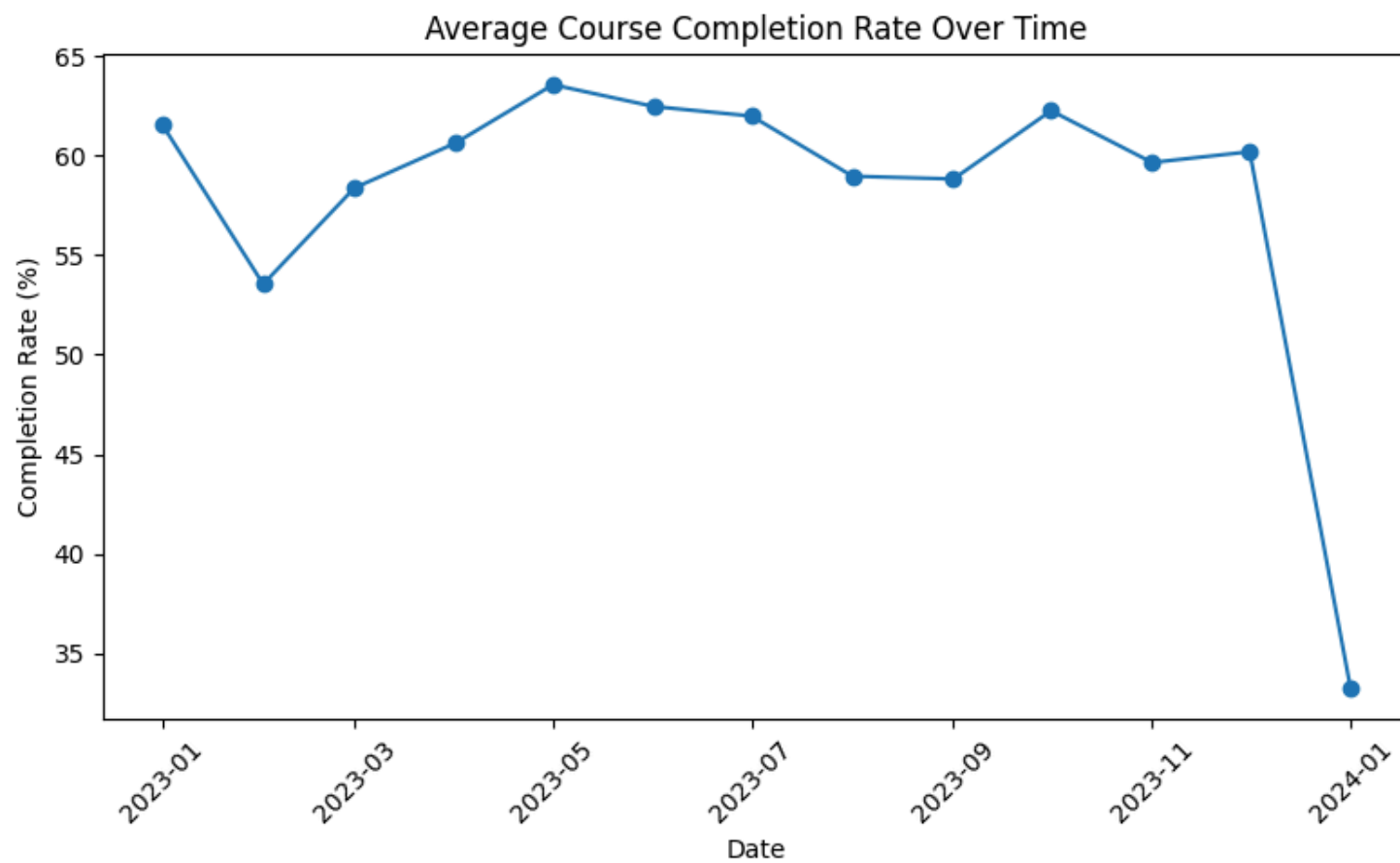
# ----- Line Chart 2: Completion Rate -----
plt.figure(figsize=(8,5))
plt.plot(monthly_data.index, monthly_data["Course_Completion_Rate(%)"], marker='o')

plt.title("Average Course Completion Rate Over Time")
plt.xlabel("Date")
plt.ylabel("Completion Rate (%)")

plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Average Daily Learning Hours Over Time





In [105...

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

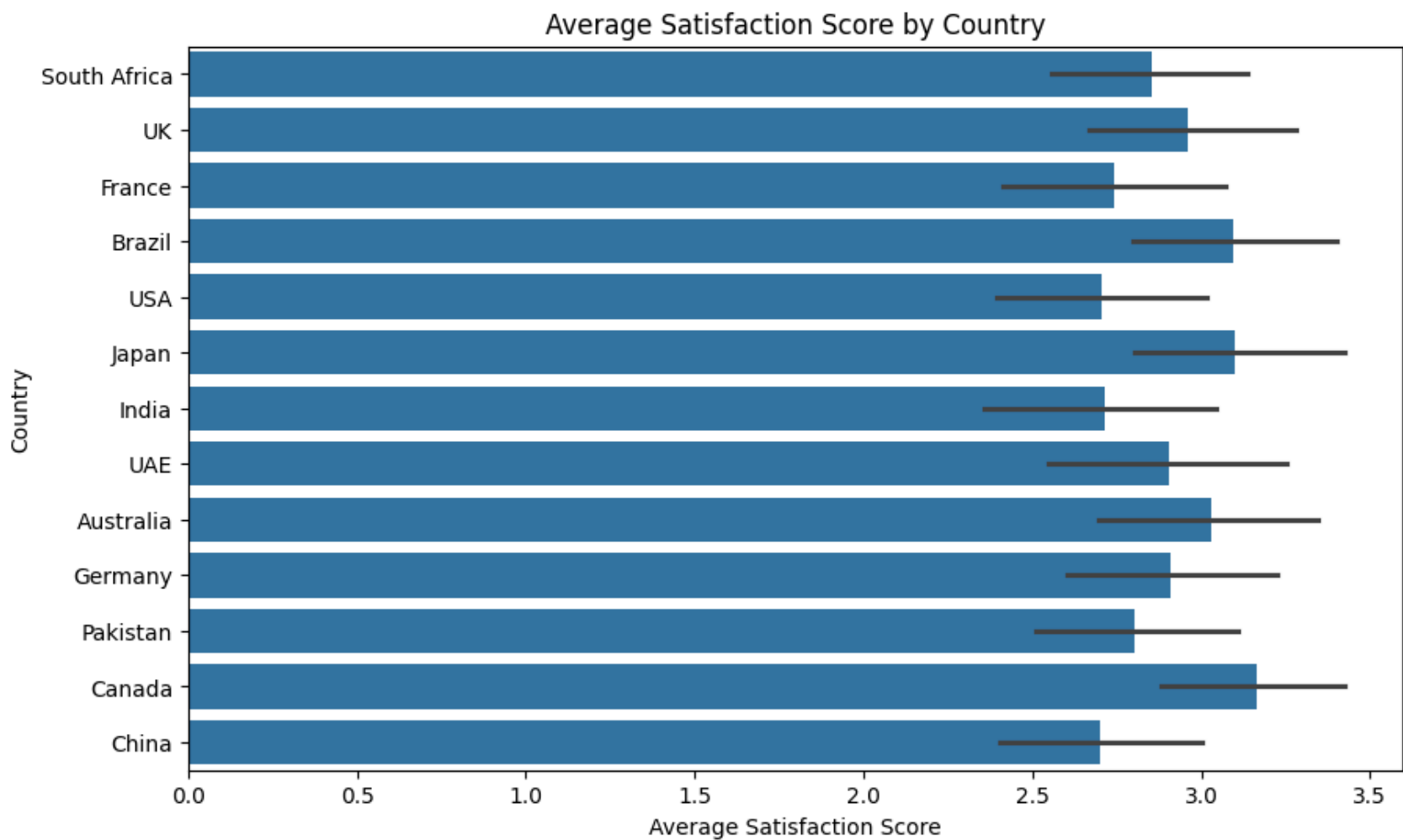
# Load dataset
df = pd.read_csv("Online_Learning_Behavior_Dataset_Worldwide.csv")

plt.figure(figsize=(10,6))

sns.barplot(y='Country', x='Satisfaction_Score(1-5)', data=df)

plt.title("Average Satisfaction Score by Country")
plt.xlabel("Average Satisfaction Score")
plt.ylabel("Country")

plt.show()
```



```
In [106... country_counts = df['Country'].value_counts()

plt.figure(figsize=(8,8))

plt.pie(country_counts,
        labels=country_counts.index,
        autopct='%1.1f%%',
        startangle=90)

# Create the hole in the middle
centre_circle = plt.Circle((0,0),0.70,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)

plt.title("Distribution of Students by Country")

plt.show()
✓ Example if you want Gender Ring Chart
```

```

gender_counts = df['Gender'].value_counts()

plt.figure(figsize=(8,8))

plt.pie(gender_counts,
        labels=gender_counts.index,
        autopct='%1.1f%%')

centre_circle = plt.Circle((0,0),0.70,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)

plt.title("Gender Distribution")

plt.show()

```

Cell In[106], line 18

✓ Example if you want Gender Ring Chart
^

SyntaxError: invalid character '✓' (U+2705)

In [107...

```

import pandas as pd
import matplotlib.pyplot as plt

# Load dataset
df = pd.read_csv("Online_Learning_Behavior_Dataset_Worldwide.csv")

# Count values
country_counts = df['Country'].value_counts()

plt.figure(figsize=(8,8))

plt.pie(country_counts,
        labels=country_counts.index,
        autopct='%1.1f%',
        startangle=90)

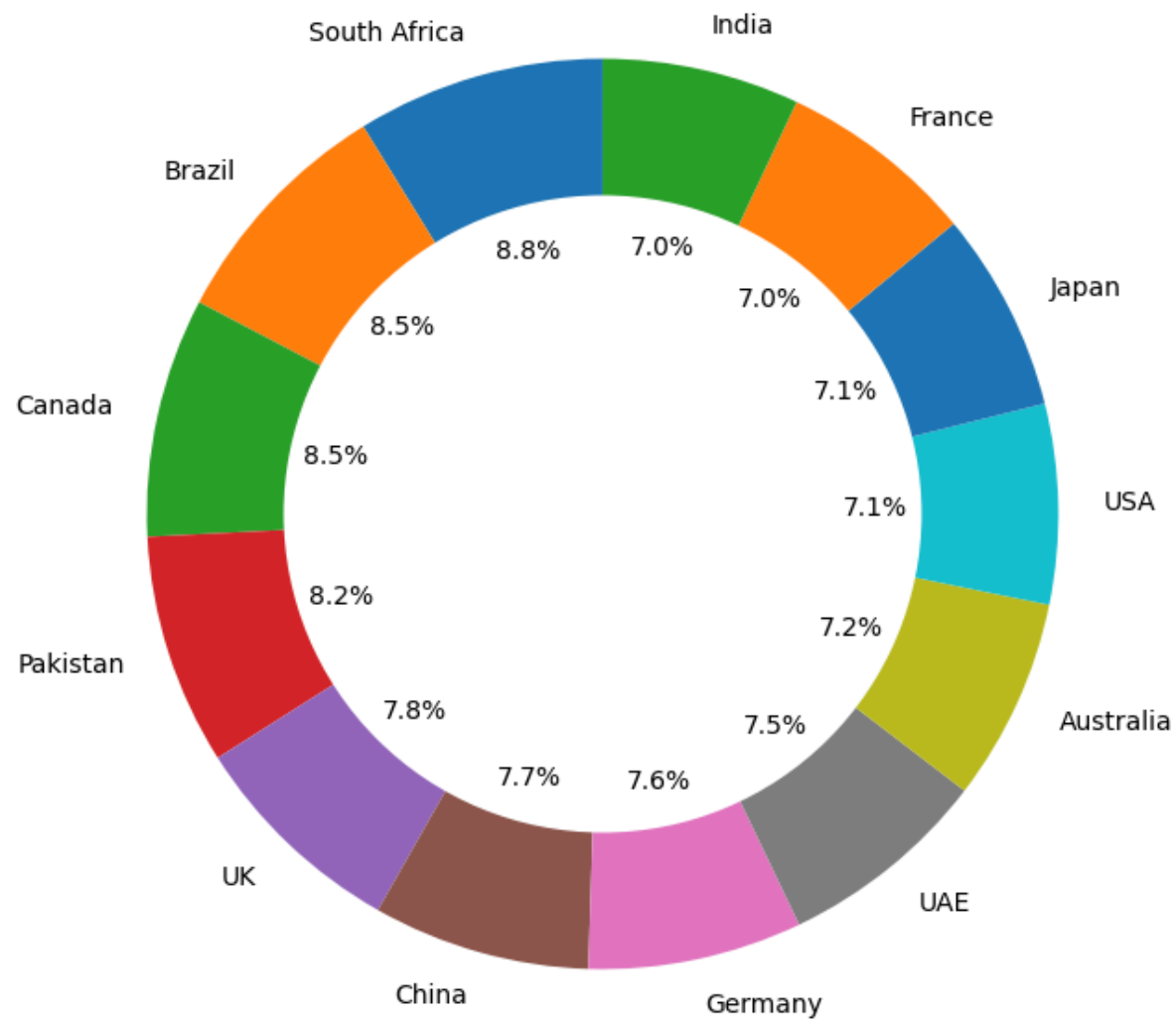
# Create the hole in the middle
centre_circle = plt.Circle((0,0),0.70,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)

plt.title("Distribution of Students by Country")

plt.show()

```

Distribution of Students by Country



```
In [108... plt.figure(figsize=(8,8))

plt.pie(country_counts,
        labels=country_counts.index,
        autopct='%1.1f%%',
        startangle=90,
        colors=colors,
        wedgeprops={'width':0.4})

# Add center text
plt.text(0,0, f'Total\n{country_counts.sum()}',
```

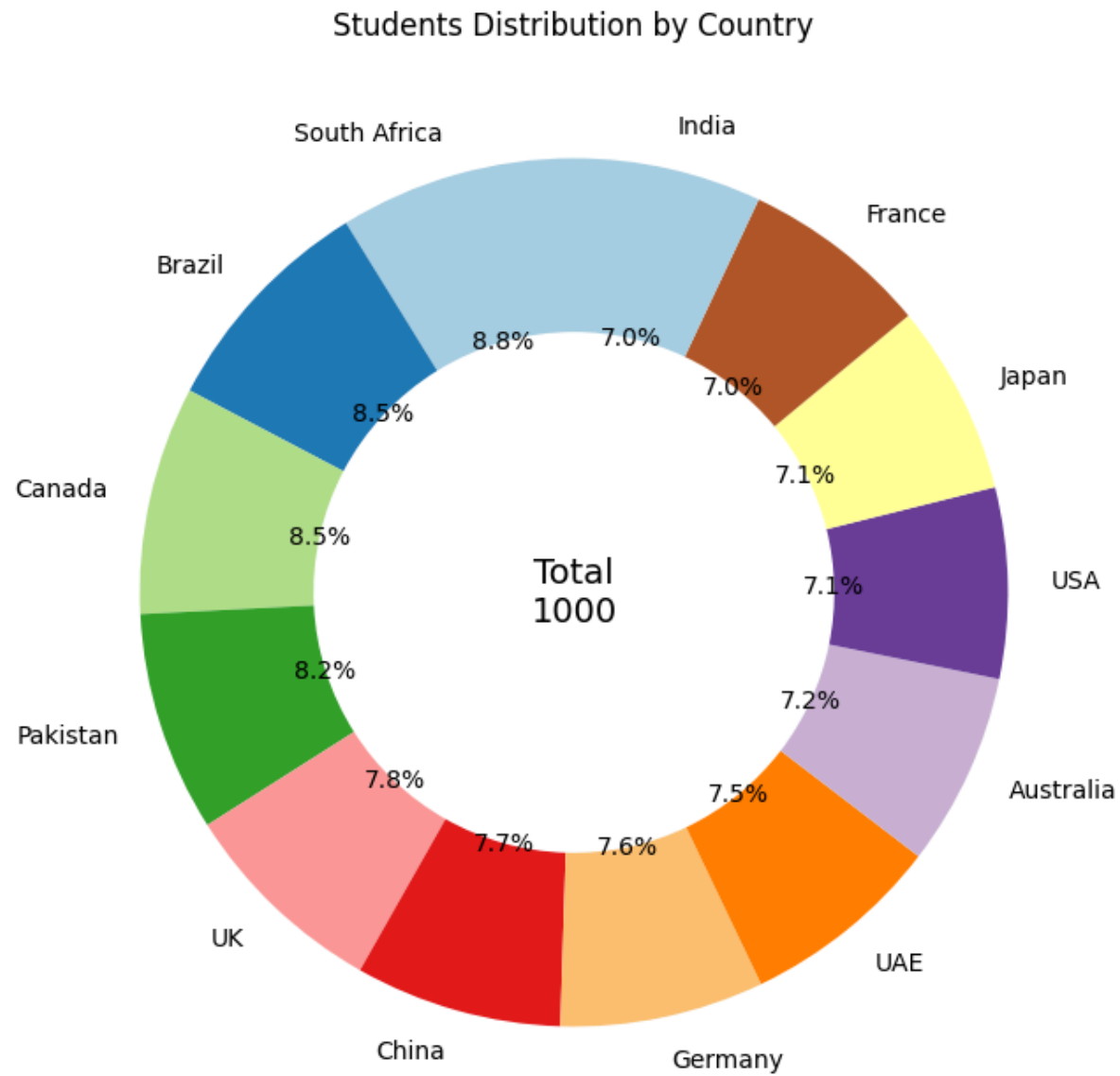
```

        ha='center', va='center', fontsize=14)

plt.title("Students Distribution by Country")

plt.show()

```



```

In [109... plt.figure(figsize=(8,8))

plt.pie(country_counts,
        labels=country_counts.index,
        autopct='%1.1f%%',
        startangle=90,

```

```

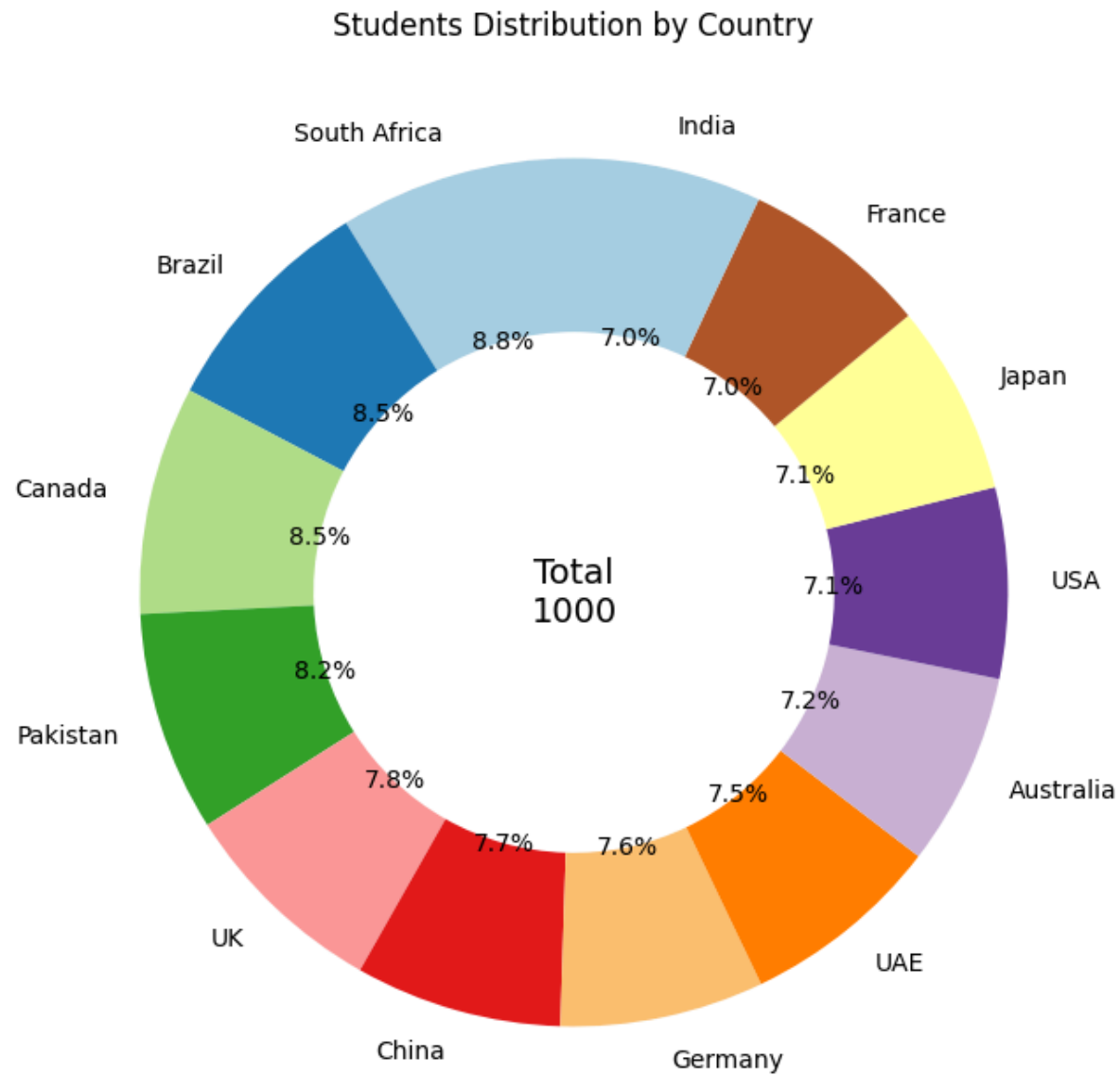
        colors=colors,
        wedgeprops={'width':0.4})

# Add center text
plt.text(0,0, f'Total\n{country_counts.sum()} ',
        ha='center', va='center', fontsize=14)

plt.title("Students Distribution by Country")

plt.show()

```



```
In [23]: import pandas as pd
import matplotlib.pyplot as plt

# Load dataset
df = pd.read_csv("Online_Learning_Behavior_Dataset_Worldwide.csv")

# Example: Country distribution
country_counts = df['Country'].value_counts()

plt.figure(figsize=(8,8))

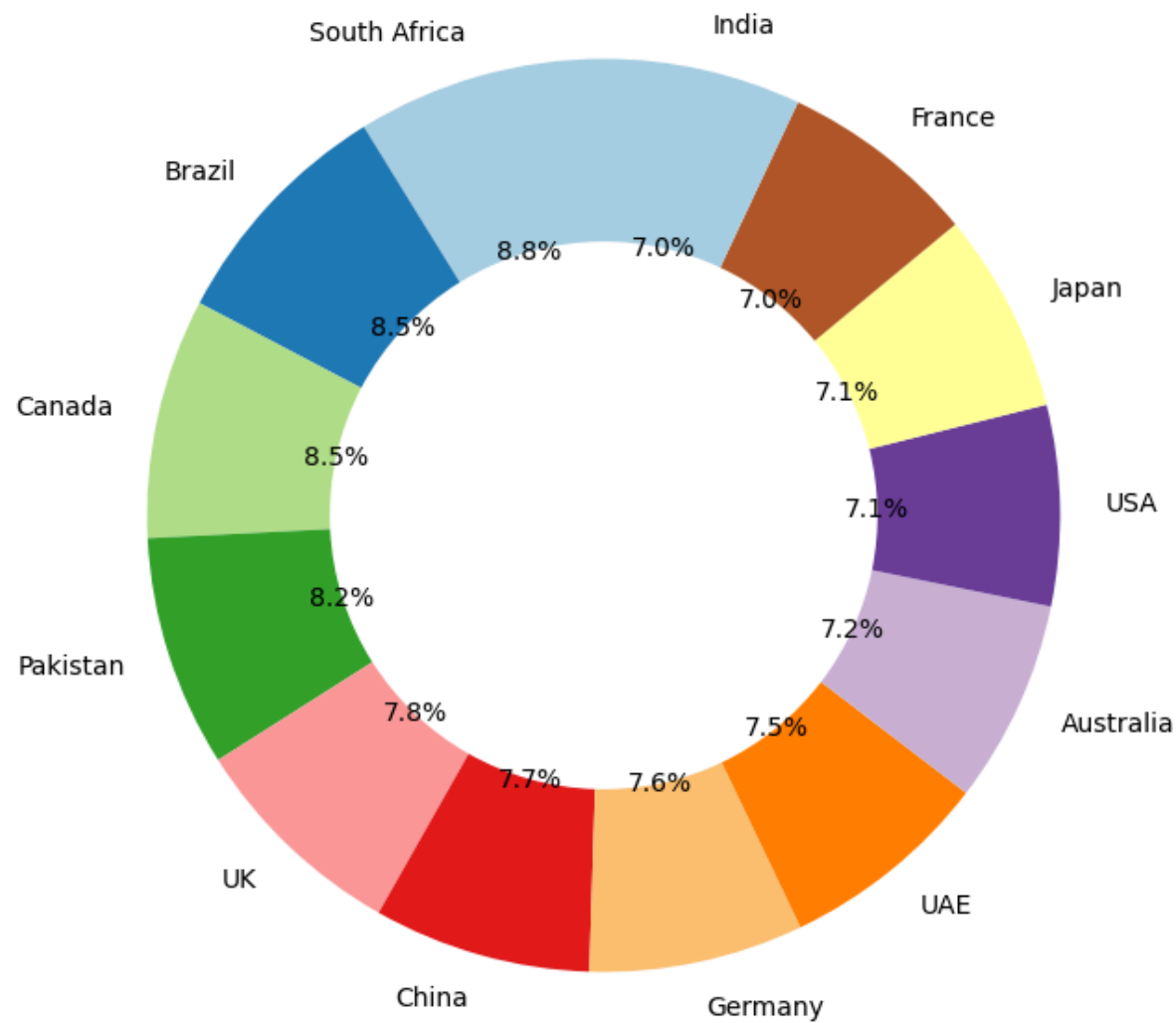
colors = plt.cm.Paired.colors # nicer color palette

plt.pie(country_counts,
        labels=country_counts.index,
        autopct='%1.1f%%',
        startangle=90,
        colors=colors,
        wedgeprops={'width':0.4}) # this creates the donut effect

plt.title("Students Distribution by Country", fontsize=14)

plt.show()
```

Students Distribution by Country



```
In [24]: plt.figure(figsize=(8,8))

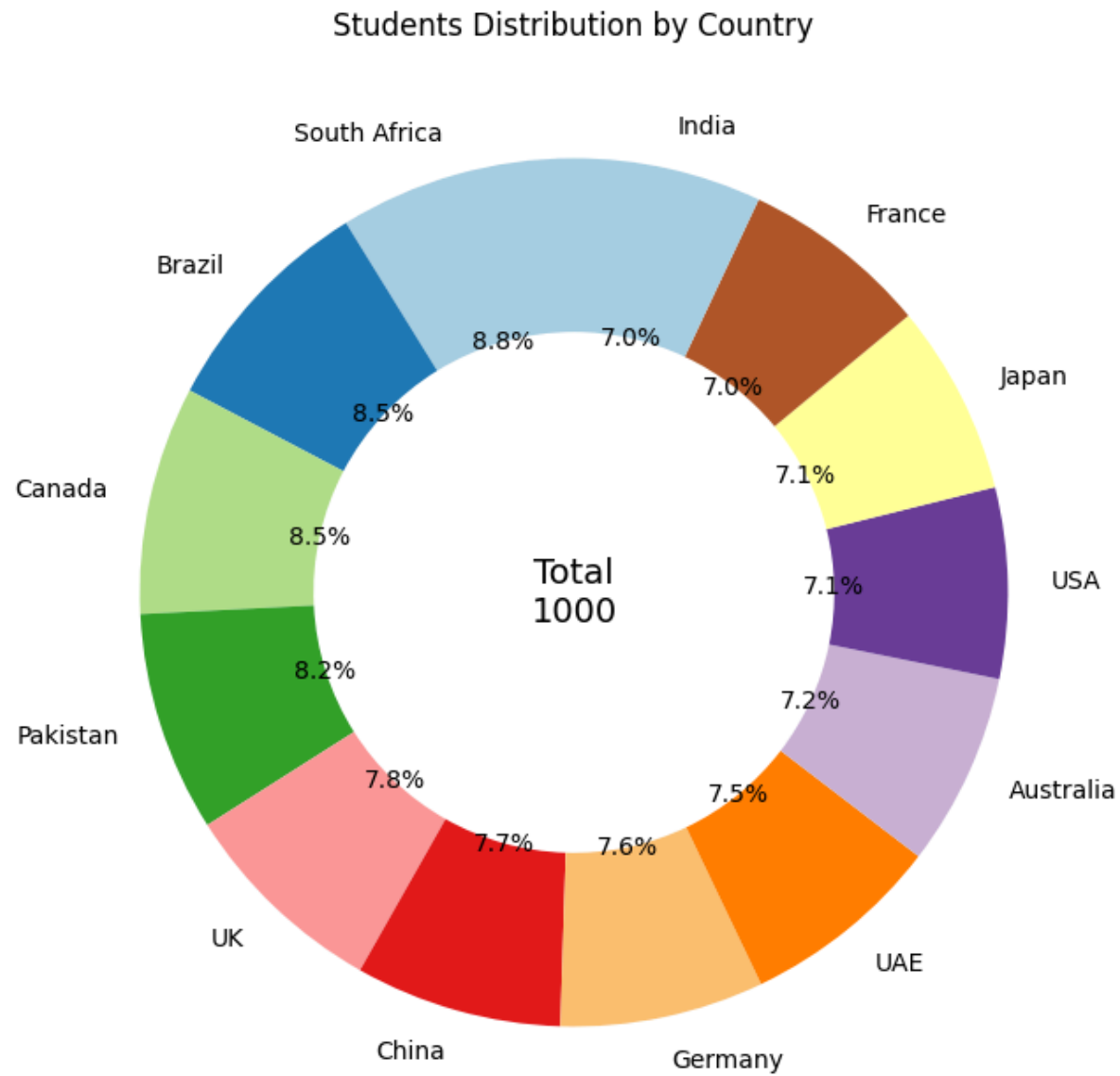
plt.pie(country_counts,
        labels=country_counts.index,
        autopct='%1.1f%%',
        startangle=90,
        colors=colors,
        wedgeprops={'width':0.4})

# Add center text
plt.text(0,0, f'Total\n{country_counts.sum()}',
```

```
ha='center', va='center', fontsize=14)

plt.title("Students Distribution by Country")

plt.show()
```



In []: