```python
# === bike_trippin.ipynb ===

#%%
%matplotlib notebook
get_ipython().run_line_magic('matplotlib', 'notebook')


#%%
# Import Dependencies
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np


#%%
# Import our data into pandas from CSV
string_thing = '../Resources/trip.csv'
bike_trips_df = pd.read_csv(string_thing, low_memory=False)

bike_trips_df


#%%
# ---------------------------------------------------------------------------
# the original DataFrame is grouped by the values within
# the "gender" column and counted

# Split up our data into groups based upon 'gender'
gender_groups = bike_trips_df.groupby('gender')

# Find out how many of each gender took bike trips
gender_trips = gender_groups['tripduration'].count()

# ---------------------------------------------------------------------------
# Need to"stoptime" because of the buggy data so that
# the data can be charted accurately:
#   Drop the 'stoptime' row that is contained within our group
gender_trips = gender_trips.drop(gender_trips.index[3])

# The title for the chart is set within the `DataFrame.plot()` method
# Chart our data, give it a title, and label the axes
gender_chart = gender_trips.plot(kind="bar", title="Bike Trips by Gender")


# The X and Y labels are set using `PandasPlot.set_xlabel()`
# and `PandasPlot.set_ylabel()
gender_chart.set_xlabel("Gender")
gender_chart.set_ylabel("Number of Trips Taken")

plt.show()
plt.tight_layout()


#%%
# ---------------------------------------------------------------------------
# The DataFrame is grouped by both the "bikeid" and "gender" columns. When
# the sum analyses is performed, therefore, a DataFrame containing multiple
# indexes is returned so that the duration is calculated per gender per bike
#
Split up our data into groups based upon 'bikeid' and 'gender'
bike_groups = bike_trips_df.groupby(['bikeid','gender'])

# Create a new variable that holds the sum of our groups
sum_it_up = bike_groups.sum()
sum_it_up.head(12)


#%%
# ----------------------------------------------------------------------------
# In order to create a chart based on one bike alone, `loc[]` must be used
# in order and a single "bikeid" must be passed. This returns a Series with
# only the "gender" column as the index and "tripduration" as the value
#
# Make a variable called bike_id and store a 'bikeid' in it
bike_id = "SEA00001"

# Collect the trips of the 'bikeid' above
just_one_bike = sum_it_up.loc[bike_id]

# Place the gender keys for that single bike into a list
gender_list = just_one_bike.keys()

# ----------------------------------------------------------------------------
# When creating a pie chart, a Y value must be passed into the `plot()` method
# This lets Pandas know what values should be set as labels, AND ...
# The title for the pie chart is being dynamically set by concatenating strings
#
# Create a pie chart based upon the trip duration of that single bike
bike_pie = just_one_bike.plot(kind="pie", y=gender_list, title=("Trips of " + bike_id))
bike_pie.set_ylabel("Trip Duration")

plt.show()
plt.tight_layout()
plt.axis("equal")
```