```python
# === sorting.ipynb ===
#%%
# Import Dependencies
import pandas as pd

#%%
# note: data is pre-sorted on happiness ranking
# from most happy to least happy (descending)
csv_path = "Resources/Happiness_2017.csv"
happiness_df = pd.read_csv(csv_path)
happiness_df.head()

#%%
# ------------------------------------------------------------------------
# Sorting the DataFrame based on "Freedom" column. The parameter `ascedning`
# is always set to True by default, which means sort order is ascending
# (lowest to highest), and an optional argument.
# `DataFrame.sort_values(column)`
# ------------------------------------------------------------------------

freedom_df = happiness_df.sort_values("Freedom")
freedom_df.head()

#%%
# To sort from highest to lowest (descending) order
# `ascending=False` must be passed in
freedom_df = happiness_df.sort_values("Freedom", ascending=False)
freedom_df.head()


#%%
# ------------------------------------------------------------------------
# It is possible to sort based upon multiple columns for which a **list**
# must be passed in. The first column will be the primary sorting method with
# ties being broken by the second column
family_and_generosity = happiness_df.sort_values(
    ["Family", "Generosity"], ascending=False)
family_and_generosity.head()


#%%
# ------------------------------------------------------------------------
# The `reset_index` method will recalculate the index for each row based
# upon their position within the new DataFrame and, as such, will allow for
# far simpler referencing of rows in the future.
#
# Passing `drop=True` into `df.reset_index()` will ensure no new column is
# created when the index is reset
# ------------------------------------------------------------------------

# The index can be reset to provide index numbers based on the new rankings.
new_index = family_and_generosity.reset_index(drop=True)
new_index.head()
```