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


#%%
# Create the first dataframe
raw_data_info = {
    "customer_id": [112, 403, 999, 543, 123],
    "name": ["John", "Kelly", "Sam", "April", "Bobbo"],
    "email": ["jman@gmail", "kelly@aol.com", "sports@school.edu",
              "April@yahoo.com", "HeyImBobbo@msn.com"]
}
info_pd = pd.DataFrame(raw_data_info, columns=["customer_id", "name", "email"])
info_pd


#%%
# Create a second DataFrame
raw_data_items = {
    "customer_id": [403, 112, 543, 999, 654],
    "item": ["soda", "chips", "TV", "Laptop", "Cooler"],
    "cost": [3.00, 4.50, 600, 900, 150]
}
items_pd = pd.DataFrame(raw_data_items, columns=[
                        "customer_id", "item", "cost"])
items_pd


#%%
# ------------------------------------------------------------------------
# Merge two dataframes using an inner join.
# Inner joins are the default means through which DataFrames are combined
# using the `pd.merge()` method and will only return data whose values match.
# Any rows that do not include matching data will be dropped from the combined
# DataFrame
#
# These two DataFrames below share the "customer_id" column in common.
# ------------------------------------------------------------------------

merge_table = pd.merge(info_pd, items_pd, on="customer_id")
merge_table


#%%
# ------------------------------------------------------------------------
# An outer join is the opposite of an inner join. It combines the DataFrames
# regardless of whether any of the rows match and must be declared as a
# parameter within the `pd.merge()` method using the syntax `how="outer"`.
# Any rows that do not include matching data will have the values within
# replaced with `NaN` instead.
#
# Merge two dataframes using an outer join
# ------------------------------------------------------------------------
```

```
merge_table = pd.merge(info_pd, items_pd, on="customer_id", how="outer")
merge_table


#%%
# ------------------------------------------------------------------------
# A left-outer join protects info on the left dataframe while keeping only
# rows of the right dataframe that have customer_id in common the `pd.merge()`
# method is used and 4 parameters are passed into it referencing both of
# the DataFrames and the value `on="customer_id"`
#
# Merge two dataframes using a left join
# ------------------------------------------------------------------------
merge_table = pd.merge(info_pd, items_pd, on="customer_id", how="left")
merge_table


#%%
# ------------------------------------------------------------------------
# A right-outer join is the opposite of the left-outer join in that it protects
# info on the right dataframe while keeping only rows of the left that have
# customer_id in common
#
# Merge two dataframes using a right join
# ------------------------------------------------------------------------
merge_table = pd.merge(info_pd, items_pd, on="customer_id", how="right")
merge_table
```