Merging and Concatenating Dataframes

In this section, you will merge and concatenate multiple dataframes. Merging is one of the most common operations you will do, since data often comes in various files.

In our case, we have sales data of a retail store spread across multiple files. We will now work with all these data files and learn to:

- Merge multiple dataframes using common columns/keys using `pd.merge()`
- Concatenate dataframes using `pd.concat()`

Let's first read all the data files.

In [1]:
```
# loading libraries and reading the data
import numpy as np
import pandas as pd

market_df = pd.read_csv("../global_sales_data/market_fact.csv")
customer_df = pd.read_csv("../global_sales_data/cust_dimen.csv")
product_df = pd.read_csv("../global_sales_data/prod_dimen.csv")
shipping_df = pd.read_csv("../global_sales_data/shipping_dimen.csv")
orders_df = pd.read_csv("../global_sales_data/orders_dimen.csv")
```

Merging Dataframes Using `pd.merge()`

There are five data files:

1. The `market_fact` table contains the sales data of each order
2. The other 4 files are called 'dimension tables/files' and contain metadata about customers, products, shipping details, order details etc.

If you are familiar with star schemas and data warehouse designs, you will note that we have one fact table and four dimension tables.

In [2]:
```
# Already familiar with market data: Each row is an order
market_df.head()
```

Out[2]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipping
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.81	0.01	23	-30.51	
1	Ord_5406	Prod_13	SHP_7549	Cust_1818	42.27	0.01	13	4.56	
2	Ord_5446	Prod_4	SHP_7610	Cust_1818	4701.69	0.00	26	1148.90	
3	Ord_5456	Prod_6	SHP_7625	Cust_1818	2337.89	0.09	43	729.34	
4	Ord_5485	Prod_17	SHP_7664	Cust_1818	4233.15	0.08	35	1219.87	

In [3]:
```
# Customer dimension table: Each row contains metadata about customers
customer_df.head()
```

Out[3]:

	Customer_Name	Province	Region	Customer_Segment	Cust_id
0	MUHAMMED MACINTYRE	NUNAVUT	NUNAVUT	SMALL BUSINESS	Cust_1
1	BARRY FRENCH	NUNAVUT	NUNAVUT	CONSUMER	Cust_2
2	CLAY ROZENDAL	NUNAVUT	NUNAVUT	CORPORATE	Cust_3
3	CARLOS SOLTERO	NUNAVUT	NUNAVUT	CONSUMER	Cust_4
4	CARL JACKSON	NUNAVUT	NUNAVUT	CORPORATE	Cust_5

In [4]:
```
# Product dimension table
product_df.head()
```

Out[4]:

	Product_Category	Product_Sub_Category	Prod_id
0	OFFICE SUPPLIES	STORAGE & ORGANIZATION	Prod_1
1	OFFICE SUPPLIES	APPLIANCES	Prod_2
2	OFFICE SUPPLIES	BINDERS AND BINDER ACCESSORIES	Prod_3
3	TECHNOLOGY	TELEPHONES AND COMMUNICATION	Prod_4
4	FURNITURE	OFFICE FURNISHINGS	Prod_5

In [5]:
```
# Shipping metadata
shipping_df.head()
```

Out[5]:

	Order_ID	Ship_Mode	Ship_Date	Ship_id
0	3	REGULAR AIR	20-10-2010	SHP_1
1	293	DELIVERY TRUCK	02-10-2012	SHP_2
2	293	REGULAR AIR	03-10-2012	SHP_3
3	483	REGULAR AIR	12-07-2011	SHP_4
4	515	REGULAR AIR	30-08-2010	SHP_5

In [6]:
```
# Orders dimension table
orders_df.head()
```

Out[6]:

	Order_ID	Order_Date	Order_Priority	Ord_id
0	3	13-10-2010	LOW	Ord_1
1	293	01-10-2012	HIGH	Ord_2
2	483	10-07-2011	HIGH	Ord_3
3	515	28-08-2010	NOT SPECIFIED	Ord_4
4	613	17-06-2011	HIGH	Ord_5

Merging Dataframes

Say you want to select all orders and observe the `Sales` of the customer segment *Corporate*. Since customer segment details are present in the dataframe `customer_df`, we will first need to merge it with `market_df`.

In [7]:
```
# Merging the dataframes
# Note that Cust_id is the common column/key, which is provided to the 'on' argume
# how = 'inner' makes sure that only the customer ids present in both dfs are inc
df_1 = pd.merge(market_df, customer_df, how='inner', on='Cust_id')
df_1.head()
```

Out[7]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipping
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.81	0.01	23	-30.51	
1	Ord_5406	Prod_13	SHP_7549	Cust_1818	42.27	0.01	13	4.56	
2	Ord_5446	Prod_4	SHP_7610	Cust_1818	4701.69	0.00	26	1148.90	
3	Ord_5456	Prod_6	SHP_7625	Cust_1818	2337.89	0.09	43	729.34	
4	Ord_5485	Prod_17	SHP_7664	Cust_1818	4233.15	0.08	35	1219.87	

In [8]:
```
# Now, you can subset the orders made by customers from 'Corporate' segment
df_1.loc[df_1['Customer_Segment'] == 'CORPORATE', :]
```

Out[8]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.8100	0.01	23	-30.51
1	Ord_5406	Prod_13	SHP_7549	Cust_1818	42.2700	0.01	13	4.56
2	Ord_5446	Prod_4	SHP_7610	Cust_1818	4701.6900	0.00	26	1148.90
3	Ord_5456	Prod_6	SHP_7625	Cust_1818	2337.8900	0.09	43	729.34
4	Ord_5485	Prod_17	SHP_7664	Cust_1818	4233.1500	0.08	35	1219.87
5	Ord_5446	Prod_6	SHP_7608	Cust_1818	164.0200	0.03	23	-47.64
6	Ord_31	Prod_12	SHP_41	Cust_26	14.7600	0.01	5	1.32

In [9]:
```
# Example 2: Select all orders from product category = office supplies and from t
# We now need to merge the product_df

df_2 = pd.merge(df_1, product_df, how='inner', on='Prod_id')
df_2.head()
```

Out[9]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipping_C
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.81	0.01	23	-30.51	3
1	Ord_2978	Prod_16	SHP_4112	Cust_1088	305.05	0.04	27	23.12	3
2	Ord_5484	Prod_16	SHP_7663	Cust_1820	322.82	0.05	35	-17.58	3
3	Ord_3730	Prod_16	SHP_5175	Cust_1314	459.08	0.04	34	61.57	3
4	Ord_4143	Prod_16	SHP_5771	Cust_1417	207.21	0.06	24	-78.64	6

In [10]:
```
# Select all orders from product category = office supplies and from the corporat
df_2.loc[(df_2['Product_Category']=='OFFICE SUPPLIES') & (df_2['Customer_Segment'
```

Out[10]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	S
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.81	0.01	23	-30.51	
3	Ord_3730	Prod_16	SHP_5175	Cust_1314	459.08	0.04	34	61.57	
7	Ord_4506	Prod_16	SHP_6273	Cust_1544	92.02	0.07	9	-24.88	
9	Ord_1551	Prod_16	SHP_2145	Cust_531	184.77	0.00	29	-71.96	
11	Ord_1429	Prod_16	SHP_1976	Cust_510	539.06	0.05	42	-123.07	
14	Ord_43	Prod_16	SHP_56	Cust_34	9620.82	0.04	6	-1759.58	
15	Ord_956	Prod_16	SHP_1324	Cust_346	503.17	0.01	46	-10.21	

Similary, you can merge the other dimension tables - `shipping_df` and `orders_df` to create a `master_df` and perform indexing using any column in the master dataframe.

In [11]:
```
# Merging shipping_df
df_3 = pd.merge(df_2, shipping_df, how='inner', on='Ship_id')
df_3.shape
```

Out[11]: (8399, 19)

In [12]:
```
# Merging the orders table to create a master df
master_df = pd.merge(df_3, orders_df, how='inner', on='Ord_id')
master_df.shape
master_df.head()
```

Out[12]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipping_
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.81	0.01	23	-30.51	
1	Ord_5446	Prod_4	SHP_7610	Cust_1818	4701.69	0.00	26	1148.90	
2	Ord_5446	Prod_6	SHP_7608	Cust_1818	164.02	0.03	23	-47.64	
3	Ord_2978	Prod_16	SHP_4112	Cust_1088	305.05	0.04	27	23.12	
4	Ord_5484	Prod_16	SHP_7663	Cust_1820	322.82	0.05	35	-17.58	

5 rows × 22 columns

Similary, you can perform left, right and outer merges (joins) by using the argument `how = 'left' / 'right' / 'outer'` .

Concatenating Dataframes

Concatenation is much more straightforward than merging. It is used when you have dataframes having the same columns and want to append them (pile one on top of the other), or having the same rows and want to append them side-by-side.

Concatenating Dataframes Having the Same columns

Say you have two dataframes having the same columns, like so:

In [13]:
```
# dataframes having the same columns
df1 = pd.DataFrame({'Name': ['Aman', 'Joy', 'Rashmi', 'Saif'],
                    'Age': ['34', '31', '22', '33'],
                    'Gender': ['M', 'M', 'F', 'M']}
                   )

df2 = pd.DataFrame({'Name': ['Akhil', 'Asha', 'Preeti'],
                    'Age': ['31', '22', '23'],
                    'Gender': ['M', 'F', 'F']}
                   )
df1
```

Out[13]:

	Age	Gender	Name
0	34	M	Aman
1	31	M	Joy
2	22	F	Rashmi
3	33	M	Saif

In [14]:
```
df2
```

Out[14]:

	Age	Gender	Name
0	31	M	Akhil
1	22	F	Asha
2	23	F	Preeti

In [15]:
```
# To concatenate them, one on top of the other, you can use pd.concat
# The first argument is a sequence (list) of dataframes
# axis = 0 indicates that we want to concat along the row axis
pd.concat([df1, df2], axis = 0)
```

Out[15]:

	Age	Gender	Name
0	34	M	Aman
1	31	M	Joy
2	22	F	Rashmi
3	33	M	Saif
0	31	M	Akhil
1	22	F	Asha
2	23	F	Preeti

In [16]:
```
# A useful and intuitive alternative to concat along the rows is the append() fun
# It concatenates along the rows
df1.append(df2)
```

Out[16]:

	Age	Gender	Name
0	34	M	Aman
1	31	M	Joy
2	22	F	Rashmi
3	33	M	Saif
0	31	M	Akhil
1	22	F	Asha
2	23	F	Preeti

Concatenating Dataframes Having the Same Rows

You may also have dataframes having the same rows but different columns (and having no common columns). In this case, you may want to concat them side-by-side. For e.g.:

In [17]:
```
df1 = pd.DataFrame({'Name': ['Aman', 'Joy', 'Rashmi', 'Saif'],
                    'Age': ['34', '31', '22', '33'],
                    'Gender': ['M', 'M', 'F', 'M']}
                   )
df1
```

Out[17]:

	Age	Gender	Name
0	34	M	Aman
1	31	M	Joy
2	22	F	Rashmi
3	33	M	Saif

In [18]:
```
df2 = pd.DataFrame({'School': ['RK Public', 'JSP', 'Carmel Convent', 'St. Paul'],
                    'Graduation Marks': ['84', '89', '76', '91']}
                   )
df2
```

Out[18]:

	Graduation Marks	School
0	84	RK Public
1	89	JSP
2	76	Carmel Convent
3	91	St. Paul

In [19]:
```
# To join the two dataframes, use axis = 1 to indicate joining along the columns
# The join is possible because the corresponding rows have the same indices
pd.concat([df1, df2], axis = 1)
```

Out[19]:

	Age	Gender	Name	Graduation Marks	School
0	34	M	Aman	84	RK Public
1	31	M	Joy	89	JSP
2	22	F	Rashmi	76	Carmel Convent
3	33	M	Saif	91	St. Paul

Note that you can also use the `pd.concat()` method to merge dataframes using common keys, though here we will not discuss that. For simplicity, we have used the `pd.merge()` method for database-style merging and `pd.concat()` for appending dataframes having no common columns.