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
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' ar
gument
# how = 'inner' makes sure that only the customer ids present in both dfs are
 included in the result
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
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	
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.8100	0.01	23	-3
1	Ord_5406	Prod_13	SHP_7549	Cust_1818	42.2700	0.01	13	4.
2	Ord_5446	Prod_4	SHP_7610	Cust_1818	4701.6900	0.00	26	11
3	Ord_5456	Prod_6	SHP_7625	Cust_1818	2337.8900	0.09	43	72
4	Ord_5485	Prod_17	SHP_7664	Cust_1818	4233.1500	0.08	35	12
5	Ord_5446	Prod_6	SHP_7608	Cust_1818	164.0200	0.03	23	-4
6	Ord_31	Prod_12	SHP_41	Cust_26	14.7600	0.01	5	1.
45	Ord_4768	Prod_12	SHP_6650	Cust_1579	3.8500	0.08	1	-1
46	Ord_4690	Prod_2	SHP_6545	Cust_1579	904.1200	0.07	4	28
47	Ord_4659	Prod_6	SHP_6493	Cust_1579	1451.5900	0.06	26	43
48	Ord_4622	Prod_12	SHP_6437	Cust_1579	56.2600	0.00	14	28
49	Ord_4657	Prod_5	SHP_6490	Cust_1579	336.8600	0.06	23	36
50	Ord_4682	Prod_5	SHP_6531	Cust_1579	63.9100	0.04	14	-4
51	Ord_4768	Prod_4	SHP_6651	Cust_1579	547.0005	0.08	10	-9
52	Ord_4755	Prod_13	SHP_6628	Cust_1579	3.4100	0.06	1	-1
53	Ord_4659	Prod_6	SHP_6492	Cust_1579	391.9000	0.10	11	-4
54	Ord_4682	Prod_4	SHP_6531	Cust_1579	1909.0065	0.01	32	34
55	Ord_4657	Prod_5	SHP_6489	Cust_1579	418.0300	0.00	47	-1
56	Ord_4410	Prod_5	SHP_6146	Cust_1474	605.7700	0.05	42	12

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	
57	Ord_4546	Prod_1	SHP_6327	Cust_1474	5208.7800	0.05	34	15
58	Ord_4499	Prod_6	SHP_6264	Cust_1474	67.4900	0.06	1	-2
59	Ord_4552	Prod_4	SHP_6335	Cust_1474	284.4525	0.06	5	-2
60	Ord_4552	Prod_12	SHP_6335	Cust_1474	104.7000	0.09	29	42
61	Ord_4497	Prod_13	SHP_6261	Cust_1474	211.9700	0.06	10	-4
62	Ord_4410	Prod_6	SHP_6146	Cust_1474	1480.9100	0.00	44	48
63	Ord_4499	Prod_13	SHP_6265	Cust_1474	6.7600	0.02	3	-4
64	Ord_4475	Prod_4	SHP_6234	Cust_1474	7640.2250	0.07	46	20
65	Ord_4547	Prod_6	SHP_6328	Cust_1474	23.4400	0.05	3	-1
66	Ord_4360	Prod_11	SHP_6079	Cust_1474	1736.4100	0.10	12	-7
67	Ord_4541	Prod_2	SHP_6322	Cust_1474	336.6500	0.07	30	-7
...	...	...	...	...	...	...	...	...
8351	Ord_3584	Prod_15	SHP_4961	Cust_1266	2502.6700	0.10	9	-1
8352	Ord_3638	Prod_6	SHP_5037	Cust_1266	36.4300	0.04	5	-2
8353	Ord_3610	Prod_13	SHP_4999	Cust_1266	88.5900	0.05	7	-2
8354	Ord_3592	Prod_4	SHP_4973	Cust_1266	2614.3705	0.07	25	38
8355	Ord_3592	Prod_17	SHP_4973	Cust_1266	636.5000	0.01	46	-7
8356	Ord_3639	Prod_5	SHP_5040	Cust_1266	40.3400	0.03	2	-2
8357	Ord_3610	Prod_14	SHP_4998	Cust_1266	1759.6500	0.06	3	-2
8358	Ord_3582	Prod_6	SHP_4959	Cust_1266	539.6600	0.00	26	14
8359	Ord_3543	Prod_3	SHP_4906	Cust_1266	514.2200	0.02	17	18
8360	Ord_3639	Prod_10	SHP_5038	Cust_1266	2173.2600	0.00	35	-4
8370	Ord_2696	Prod_13	SHP_3690	Cust_1006	62.7800	0.04	20	-1
8371	Ord_2624	Prod_4	SHP_3591	Cust_1006	4924.1350	0.07	28	10
8372	Ord_2772	Prod_9	SHP_3806	Cust_1006	56.9000	0.03	7	12

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	
8373	Ord_2600	Prod_16	SHP_3560	Cust_1006	106.6400	0.10	30	-3
8374	Ord_2658	Prod_5	SHP_3637	Cust_1006	1082.6600	0.08	14	-2
8375	Ord_2772	Prod_3	SHP_3806	Cust_1006	1413.8200	0.10	47	22
8376	Ord_2624	Prod_8	SHP_3590	Cust_1006	1211.0000	0.00	36	-2
8377	Ord_2722	Prod_12	SHP_3729	Cust_1006	34.0100	0.00	12	10
8378	Ord_2706	Prod_2	SHP_3705	Cust_1006	1361.9100	0.05	20	31
8379	Ord_2722	Prod_5	SHP_3730	Cust_1006	1008.9500	0.04	41	69
8380	Ord_2772	Prod_6	SHP_3807	Cust_1006	308.9200	0.04	45	-1
8381	Ord_2696	Prod_4	SHP_3691	Cust_1006	2836.0505	0.01	25	56
8382	Ord_2658	Prod_3	SHP_3636	Cust_1006	120.9800	0.00	28	-9
8383	Ord_2722	Prod_1	SHP_3731	Cust_1006	3508.3300	0.04	21	-5
8384	Ord_4620	Prod_3	SHP_6435	Cust_1577	59.6200	0.04	10	-5
8385	Ord_1833	Prod_3	SHP_2527	Cust_637	611.1600	0.04	46	10
8386	Ord_2324	Prod_7	SHP_3189	Cust_851	121.8700	0.07	39	11
8387	Ord_2220	Prod_3	SHP_3019	Cust_851	41.0600	0.04	4	-1
8388	Ord_4424	Prod_1	SHP_6165	Cust_1519	994.0400	0.03	10	-3
8389	Ord_4444	Prod_13	SHP_6192	Cust_1519	159.4100	0.00	44	34

3076 rows × 14 columns

In [9]:
```
# Example 2: Select all orders from product category = office supplies and fro
m the corporate segment
# 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	Sl
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 corpo
rate segment
df_2.loc[(df_2['Product_Category']=='OFFICE SUPPLIES') & (df_2['Customer_Segme
nt']=='CORPORATE'),:]
```

Out[10]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	P
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.81	0.01	23	-30.
3	Ord_3730	Prod_16	SHP_5175	Cust_1314	459.08	0.04	34	61.5
7	Ord_4506	Prod_16	SHP_6273	Cust_1544	92.02	0.07	9	-24.
9	Ord_1551	Prod_16	SHP_2145	Cust_531	184.77	0.00	29	-71.
11	Ord_1429	Prod_16	SHP_1976	Cust_510	539.06	0.05	42	-123
14	Ord_43	Prod_16	SHP_56	Cust_34	9620.82	0.04	6	-175
15	Ord_956	Prod_16	SHP_1324	Cust_346	503.17	0.01	46	-10.
16	Ord_975	Prod_16	SHP_1345	Cust_346	638.91	0.02	49	77.4
21	Ord_208	Prod_16	SHP_284	Cust_67	327.15	0.04	19	47.3
22	Ord_1154	Prod_16	SHP_1590	Cust_444	483.64	0.10	45	28.3
23	Ord_1486	Prod_16	SHP_2052	Cust_513	487.50	0.05	46	-25.
27	Ord_1793	Prod_16	SHP_2484	Cust_826	482.93	0.06	34	-67.
30	Ord_1005	Prod_16	SHP_1392	Cust_377	15.00	0.05	1	-11.
31	Ord_278	Prod_16	SHP_377	Cust_97	302.20	0.04	27	20.2
35	Ord_5062	Prod_16	SHP_7068	Cust_1720	587.71	0.09	49	26.2
36	Ord_4331	Prod_16	SHP_6038	Cust_1441	77.43	0.01	36	-60.
37	Ord_195	Prod_16	SHP_264	Cust_106	110.32	0.09	31	-32.
40	Ord_4864	Prod_16	SHP_6786	Cust_1679	114.53	0.07	21	-39.
41	Ord_4905	Prod_16	SHP_6842	Cust_1679	348.92	0.06	31	28.4

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	P
42	Ord_1624	Prod_16	SHP_2243	Cust_525	452.15	0.06	42	25.7
47	Ord_231	Prod_16	SHP_320	Cust_79	224.47	0.08	16	-49.
48	Ord_2547	Prod_16	SHP_3486	Cust_974	98.24	0.04	9	-10.
49	Ord_213	Prod_16	SHP_294	Cust_49	42.89	0.02	5	-32.
55	Ord_488	Prod_16	SHP_656	Cust_179	134.86	0.04	42	-62.
58	Ord_1499	Prod_16	SHP_2071	Cust_487	248.42	0.02	35	-58.
61	Ord_2350	Prod_16	SHP_3226	Cust_941	67.30	0.02	27	-115
63	Ord_487	Prod_16	SHP_655	Cust_164	343.26	0.04	50	-287
65	Ord_363	Prod_16	SHP_482	Cust_145	63.02	0.07	27	-119
72	Ord_175	Prod_16	SHP_238	Cust_104	501.31	0.05	40	38.2
74	Ord_834	Prod_16	SHP_1142	Cust_269	5.06	0.01	1	-2.6
...	...	...	...	...	...	...	...	...
7467	Ord_4728	Prod_1	SHP_6596	Cust_1607	373.13	0.05	33	-35.
7469	Ord_2097	Prod_1	SHP_2869	Cust_800	2040.95	0.07	42	-123
7470	Ord_2171	Prod_1	SHP_2959	Cust_800	302.91	0.05	24	-36.
7471	Ord_4004	Prod_1	SHP_5575	Cust_1375	236.46	0.05	23	-134
7472	Ord_3286	Prod_1	SHP_4560	Cust_1194	7156.56	0.03	50	187
7473	Ord_1531	Prod_1	SHP_2118	Cust_537	1025.02	0.04	30	87.0
7475	Ord_571	Prod_1	SHP_1160	Cust_280	195.04	0.08	1	-149

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	P
7481	Ord_1778	Prod_1	SHP_2465	Cust_587	669.69	0.03	8	-297
7484	Ord_994	Prod_1	SHP_1375	Cust_369	11103.75	0.10	32	-522
7488	Ord_696	Prod_1	SHP_953	Cust_232	309.62	0.00	21	-80.
7489	Ord_4984	Prod_1	SHP_6957	Cust_1701	298.11	0.08	25	-60.
7490	Ord_5021	Prod_1	SHP_7006	Cust_1701	192.02	0.09	7	-293
7492	Ord_2559	Prod_1	SHP_3504	Cust_988	1200.70	0.06	47	-37.
7493	Ord_2248	Prod_1	SHP_3056	Cust_879	241.89	0.02	15	-83.
7496	Ord_5221	Prod_1	SHP_7294	Cust_1741	135.38	0.05	17	-71.
7497	Ord_615	Prod_1	SHP_842	Cust_214	4538.66	0.00	40	506
7498	Ord_2198	Prod_1	SHP_2996	Cust_829	1610.76	0.10	38	-8.6
7500	Ord_3127	Prod_1	SHP_4340	Cust_1162	1724.82	0.10	32	407
7507	Ord_199	Prod_1	SHP_272	Cust_107	456.91	0.04	28	-328
7510	Ord_5014	Prod_1	SHP_6996	Cust_1698	431.11	0.07	4	-244
7513	Ord_3333	Prod_1	SHP_4626	Cust_1196	1243.45	0.10	39	-121
7520	Ord_4549	Prod_1	SHP_6332	Cust_1530	297.85	0.04	5	-69.
7529	Ord_4758	Prod_1	SHP_6637	Cust_1638	448.10	0.10	35	-15.
7540	Ord_3300	Prod_1	SHP_4578	Cust_1180	312.25	0.05	9	15.5
7541	Ord_3339	Prod_1	SHP_4632	Cust_1180	257.41	0.09	43	-131
7545	Ord_4629	Prod_1	SHP_6447	Cust_1587	848.19	0.06	25	120

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	P
7546	Ord_4604	Prod_1	SHP_6403	Cust_1522	234.24	0.09	24	-151
7551	Ord_3543	Prod_1	SHP_4905	Cust_1266	1184.11	0.07	6	-145
7552	Ord_2722	Prod_1	SHP_3731	Cust_1006	3508.33	0.04	21	-546
7553	Ord_4424	Prod_1	SHP_6165	Cust_1519	994.04	0.03	10	-335

1680 rows × 16 columns

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
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()
 function
# 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 axis
# 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.