Merging data sets & the inner join

By merging data we can answer the following questions

- What is the average transaction value of male customers?
- What is the average age of customers spending more than 200$ per year?

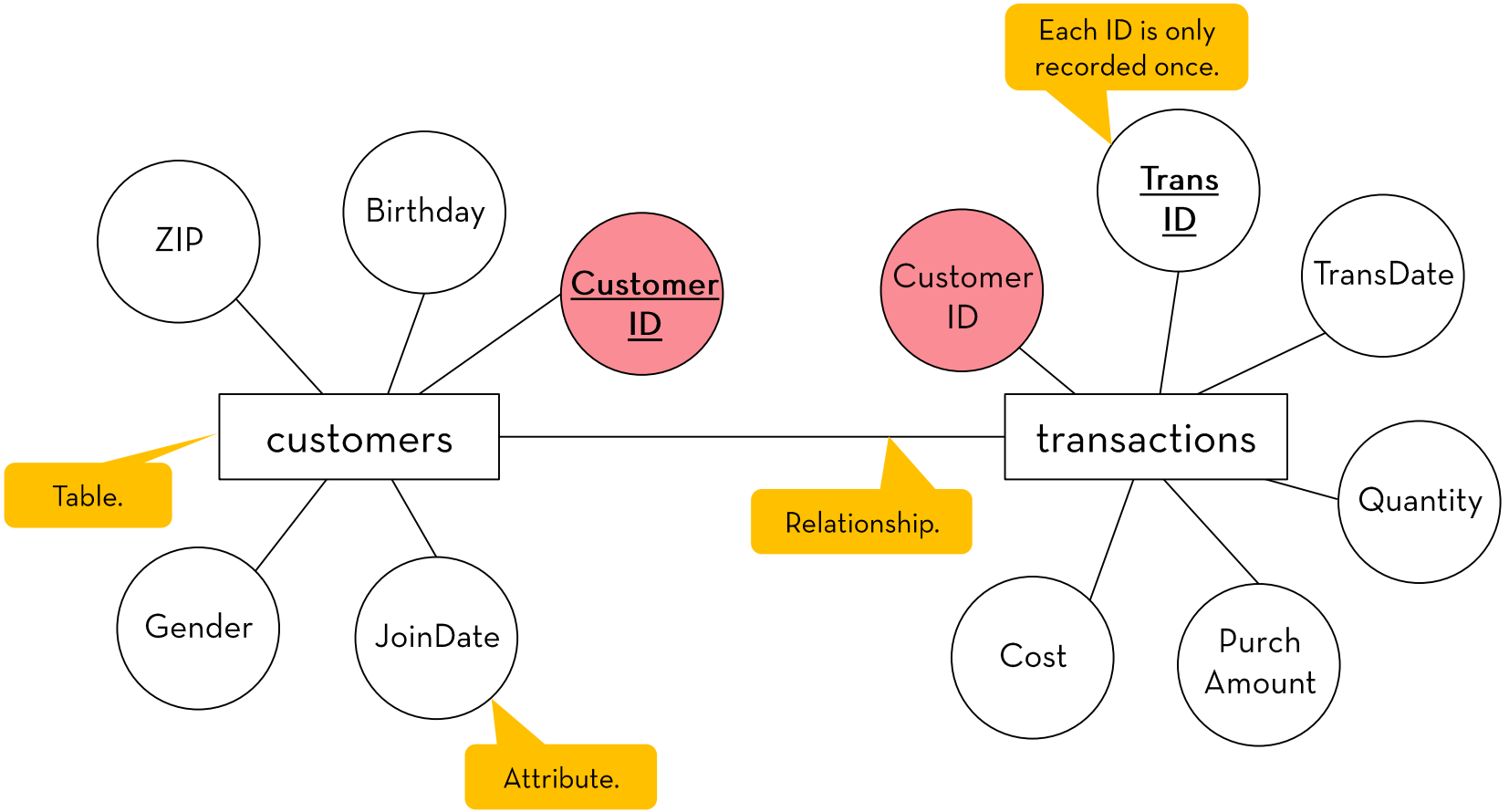
Combining datasets is a common data management task

For practical reasons information is often stored in separate datasets. However, data analysis often requires a combination of multiple datasets.

Requirements for a merge:

- min 2 DataFrames
- a common identifier (key),
- a reason to merge.

Databases reduce redundancies by storing data in tables linked through relationships



Merging - Input

A

Table with order details.

Customer	TransDate	Quantity	PurchAmount	Cost
149332	15.11.2005	1	199.95	107.00
172951	29.08.2008	1	199.95	108.00
120621	19.10.2007	1	99.95	49.00
149236	14.11.2005	1	39.95	18.95
149236	12.06.2001	1	79.95	35.00
...	...	...	...	...

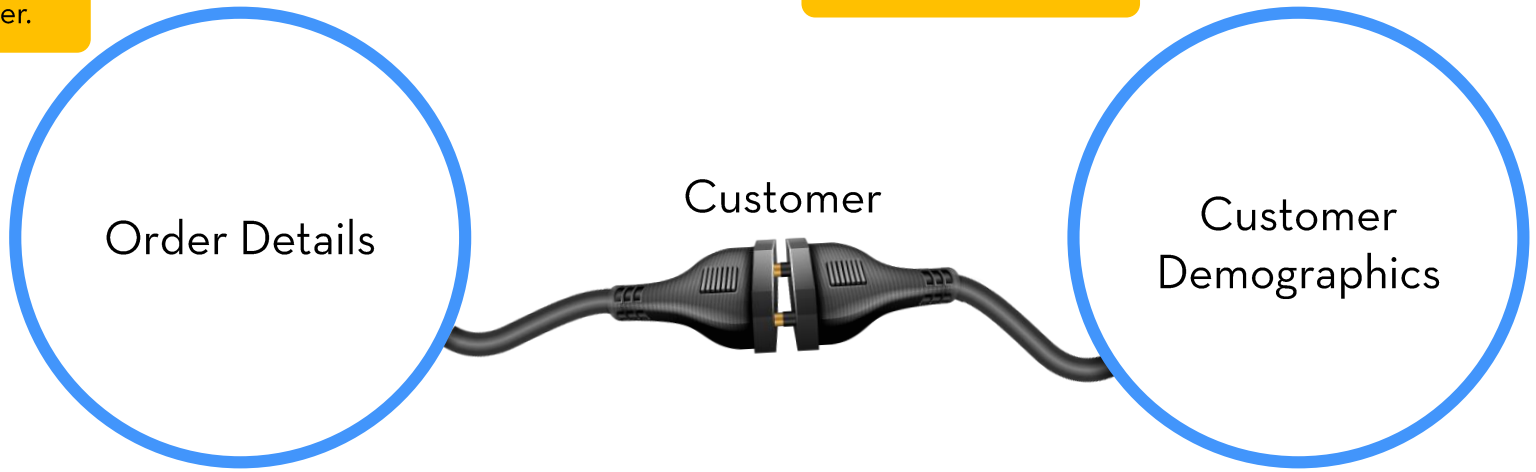
Common identifier.

B

Table with customer demographics.

Customer	Gender	Birthdate	ZIP	JoinDate
80365	f	26.08.1991	US-06332	15.09.2009
42886	f	04.05.1987	US-08055	12.06.2011
84374	m	10.07.1977	US-06400	10.08.1988
42291	m	12.07.1963	US-04533	23.07.1998
100001	m	08.05.1974	US-02332	21.02.1992
...	...	...	...	...

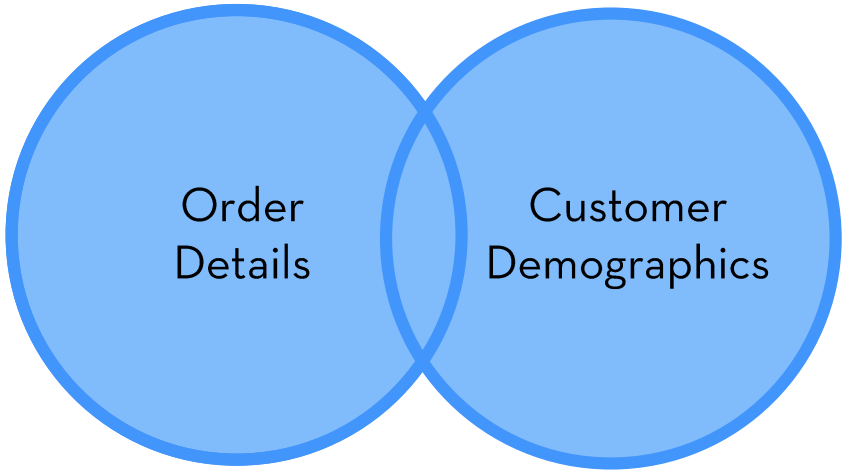
Common identifier.



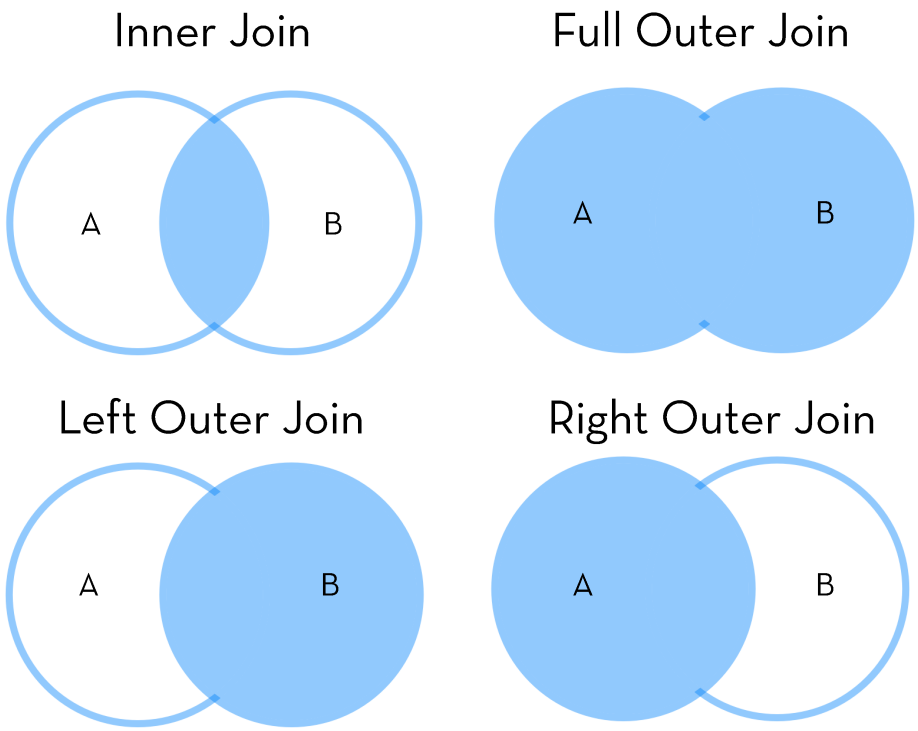
Merging - Output

Customer	TransDate	Quantity	PurchAmount	Cost	Gender	Birthdate	ZIP	JoinDate
149332	15.11.2005	1	199.55	107.00	m	07.07.1998	US-08873	05.11.2005
149332	13.12.2005	1	49.95	24.87	m	07.07.1998	US-08873	05.11.2005
149332	05.10.2006	1	24.95	12.50	m	07.07.1998	US-08873	05.11.2005
172951	29.08.2008	1	199.95	108.00	m	16.11.1963	US-11378	04.04.1980
...	...	...	...	...	...	...	...	...

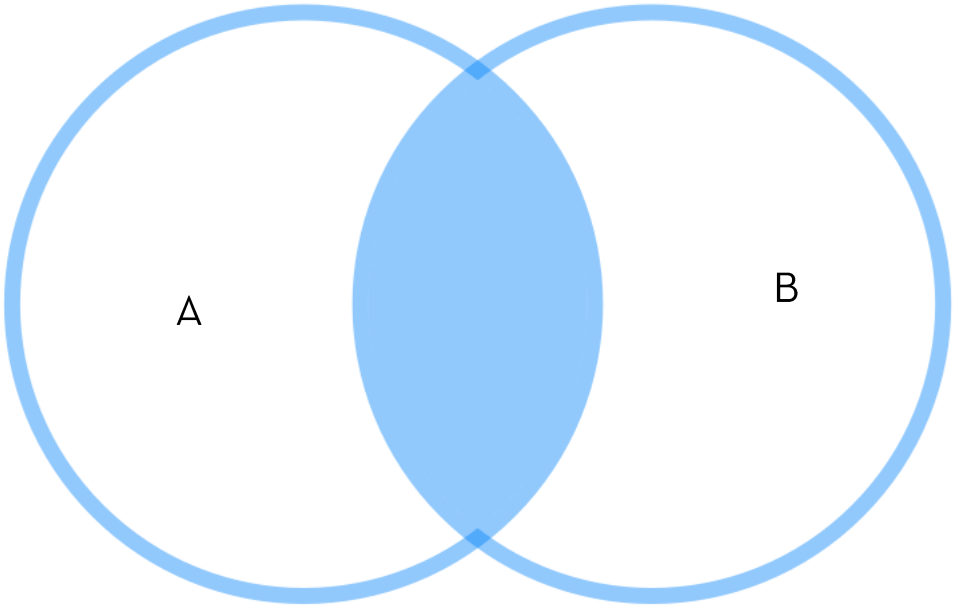
Merged by Customer.



We discuss the most common ways to merge data



Merging: inner join



Common identifier.

```
A.merge(B, how="inner", on="ID")
```

Inner joins return only the observations available in both datasets.

Inner join merges on common identifiers present in both DataFrames

A

Customer	TransDate	Quantity	PurchAmount	Cost
149332	2005-11-15	1	199.95	107.00
172951	2008-08-29	1	199.95	108.00
120621	2007-10-19	1	99.95	49.00
149236	2005-11-14	1	39.95	18.95
149236	2007-12-06	1	79.95	35.00
...	...	...	...	...

B

Customer	Gender	Birthdate	ZIP	JoinDate
80365	f	1991-08-26	US-06332	2009-09-15
149332	m	1998-07-07	US-08873	2005-11-05
84374	m	1977-07-10	US-06400	1988-08-10
149236	f	1955-08-15	US-92646	1971-02-16
100001	m	1974-05-08	US-02332	1992-02-21
...	...	...	...	...

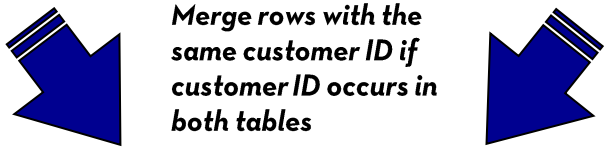
Inner join merges on common identifiers present in both DataFrames

A

Customer	TransDate	Quantity	PurchAmount	Cost
149332	2005-11-15	1	199.95	107.00
172951	2008-08-29	1	199.95	108.00
120621	2007-10-19	1	99.95	49.00
149236	2005-11-14	1	39.95	18.95
149236	2007-12-06	1	79.95	35.00
...	...	...	...	...

B

Customer	Gender	Birthdate	ZIP	JoinDate
80365	f	1991-08-26	US-06332	2009-09-15
149332	m	1998-07-07	US-08873	2005-11-05
84374	m	1977-07-10	US-06400	1988-08-10
149236	f	1955-08-15	US-92646	1971-02-16
100001	m	1974-05-08	US-02332	1992-02-21
...	...	...	...	...



Merge rows with the same customer ID if customer ID occurs in both tables

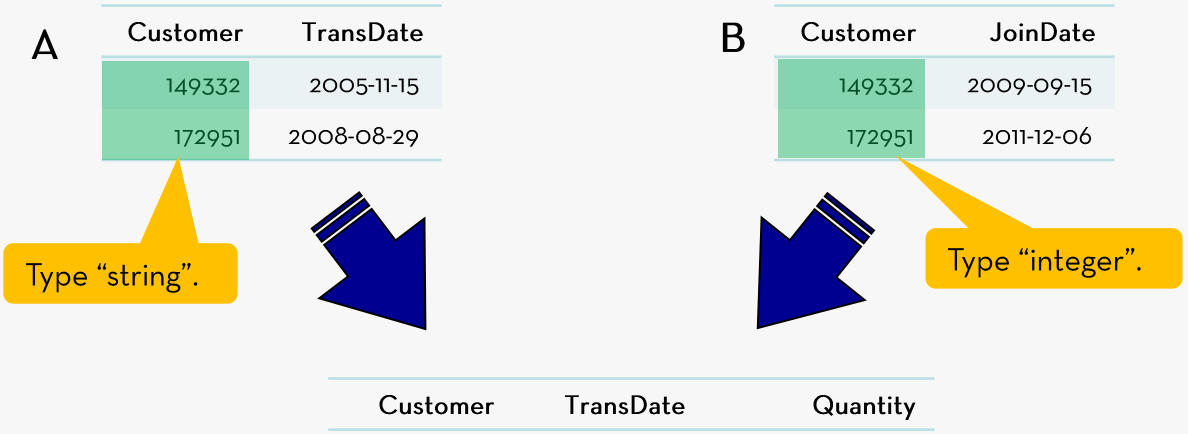
Customer	TransDate	PurchAmount	Cost	Gender	Birthdate	ZIP	JoinDate
149332	2005-11-15	1	199.95	m	1998-07-07	US-08873	05.11.2005
149236	2005-11-14	1	39.95	f	1955-08-15	US-92646	16.02.1971
149236	2001-06-12	1	79.95	f	1955-08-15	US-92646	16.02.1971
...	...	...	...	...	...	...	...

Exercise
Merging data sets & the inner join

1. Merge transactions and demographics by "Customer" using an inner join for customers born after 1980. (Hint: Check that the date is formatted correctly.)

Sidenote: The common identifier needs to be the same class

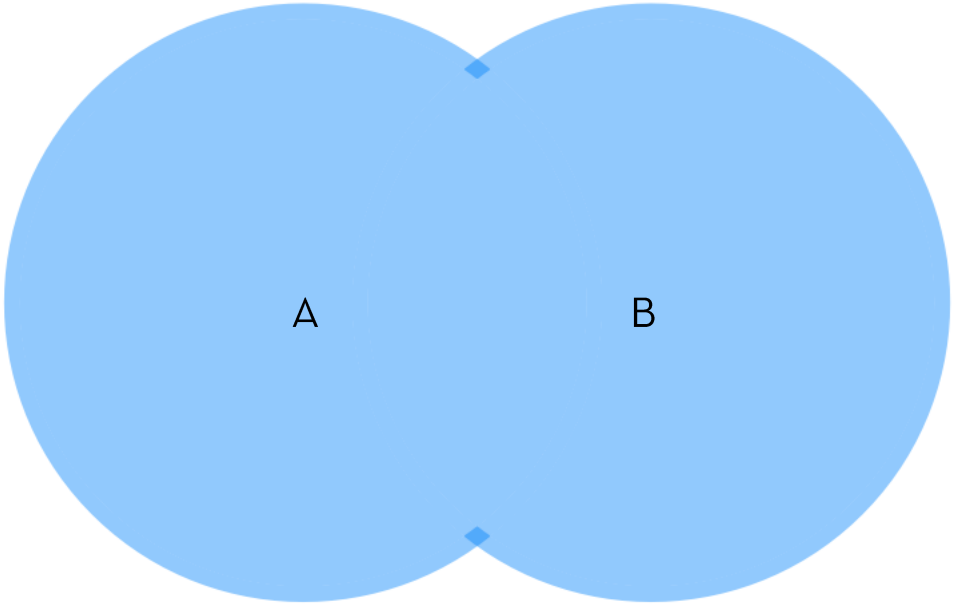
Both customer columns need to be the same class.
If one is an integer and the other is a string, the merge wont work.



Returns an empty DataFrame.

Full outer join

Merging: outer join



Common identifier.

```
A.merge(B, how="outer", on="ID")
```

Outer joins return all the data available.

Full outer join merges on all common identifiers of both DataFrames

A

Customer	TransDate	Quantity	PurchAmount	Cost
149332	2005-11-15	1	199.95	107.00
172951	2008-08-29	1	199.95	108.00
120621	2007-10-19	1	99.95	49.00
149236	2005-11-14	1	39.95	18.95
149236	2007-12-06	1	79.95	35.00
...	...	...	...	...

B

Customer	Gender	Birthdate	ZIP	JoinDate
80365	f	1991-08-26	US-06332	2009-09-15
149332	m	1998-07-07	US-08873	2005-11-05
84374	m	1977-07-10	US-06400	1988-08-10
149236	f	1955-08-15	US-92646	1971-02-16
100001	m	1974-05-08	US-02332	1992-02-21
...	...	...	...	...

Full outer join merges on all common identifiers of both DataFrames

A

Customer	TransDate	Quantity	PurchAmount	Cost
149332	2005-11-15	1	199.95	107.00
172951	2008-08-29	1	199.95	108.00
120621	2007-10-19	1	99.95	49.00
149236	2005-11-14	1	39.95	18.95
149236	2001-12-06	1	79.95	35.00
...	...	...	...	...

B

Customer	Gender	Birthdate	ZIP	JoinDate
80365	f	1991-08-26	US-06332	2009-09-15
149332	m	1998-07-07	US-08873	2005-11-05
84374	m	1977-07-10	US-06400	1988-08-10
149236	f	1955-08-15	US-92646	1971-02-16
100001	m	1974-05-08	US-02332	1992-02-21
...	...	...	...	...

Merge rows with the same customer IDs in both tables, otherwise fill entries of table with missing ID with NAs

Customer	TransDate	PurchAmount	Cost	Gender	DOB	ZIP	JoinDate
149332	2005-11-15	1	199.95	m	1998-07-07	US-08873	2005-11-05
149236	2005-11-14	1	39.95	f	1955-08-15	US-92646	1971-02-16
149236	2001-06-12	1	79.95	f	1955-08-15	US-92646	1971-02-16
172951	2008-08-29	1	199.95	NaN	NaN	NaN	NaT
120621	2007-10-19	1	99.95	NaN	NaN	NaN	NaT
80365	NaT	NaN	NaN	f	1991-08-26	US-06332	2009-09-15
48374	NaT	NaN	NaN	m	1977-07-10	US-06400	1988-08-10
...	...	...	...	...	...	...	...

Sidenote: Be careful when handling missing values (NaN/NaT)

NaN: Not a number
NaT: Not a time (for datetime objects)

Sidenote: Selecting missing and non-missing values in a DataFrame

- Select missing values:

```
myData.loc[pd.isnull(myData["PurchAmount"]), ]
```

  - Wrong (does not yield any entries):

```
myData.loc[myData["PurchAmount"]== None, ]
```

- Select non-missing values:

```
myData.loc[ ~ pd.isnull(myData["PurchAmount"]), ]
```

Use a bang for the inverse.

  - Select rows with no missing values:

```
myData.dropna()
```

Sidenote: Aggregating with missing values

```
>>> np.mean([1 ,2, np.nan, 3])
 nan
>>> np.nanmean([1,2,np.nan,3])
 2
```

Sum the non-NaN elements.

But: Aggregation omits observations with NaNs:

	Customer	TransDate	Quantity	PurchAmount	Cost	TransID	TransKey
0	149332	2005-11-15	1	NaN	107.00	127998739	100000
13	149332	2005-12-13	1	49.95	24.87	129878743	100013
14	149332	2006-10-05	1	24.95	12.50	129883508	100014

Return the mean of variable PurchAmount for Customer 149332, omit NaNs

```
myData.groupby("Customer")["PurchAmount"].mean()
```

37.45

Does not include NaNs automatically.

Sidenote: Merging with missing values

A

ID	Age
NaN	13
NaN	25
1	31
2	40

B

ID	Gender
NaN	f
1	m
2	f
NaN	f

```
A.merge(B, on="ID")
```

ID	Age	Gender
NaN	13	f
NaN	13	f
NaN	25	f
NaN	25	f
1	31	m
2	40	f

Build the cartesian products of a DataFrame by merging on a common key

A

key	Concentration	ID
1	“low”	1
1	“medium”	2
1	“high”	3

B

Group	key
a	1
b	1

A common key column is needed.

```
merge(A, B, on="key")[["ID", "Concentration", "Group"]]
```

ID	Concentration	Group
1	“low”	a
1	“low”	b
2	“medium”	a
2	“medium”	b
3	“high”	a
3	“high”	b

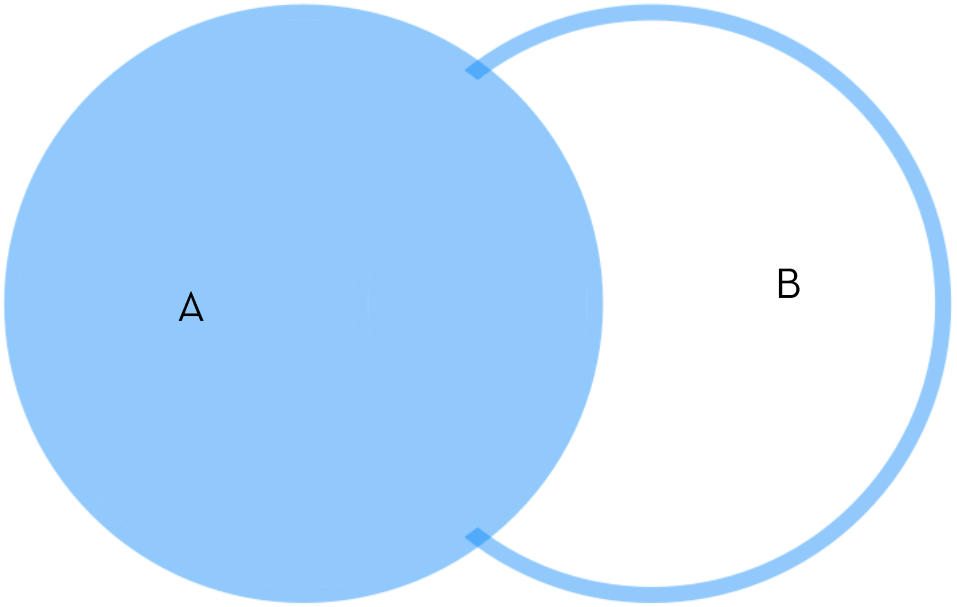
Allows the output table to be larger than the sum of the rows of the inputs.

Exercise
Full outer join

1. Merge transactions and demographics by “Customer” using an outer join for customers that purchased in 2008.

Left and right outer joins

Merging: left outer join



Common identifier.

```
A.merge(B, how="left", on="ID")
```

Use all observations from A, but only the matching from B.

Left join merges on all of the common identifiers in the left specified table

A ← left table

Right table → B

Customer	TransDate	Quantity	PurchAmount	Cost
149332	2005-11-15	1	199.95	107.00
172951	2008-08-29	1	199.95	108.00
120621	2007-10-19	1	99.95	49.00
149236	2005-11-14	1	39.95	18.95
149236	2007-12-06	1	79.95	35.00
...	...	...	...	...

Customer	Gender	Birthdate	ZIP	JoinDate
80365	f	1991-08-26	US-06332	2009-09-15
149332	m	1998-07-07	US-08873	2005-11-05
84374	m	1977-07-10	US-06400	1988-08-10
149236	f	1955-08-15	US-92646	1971-02-16
100001	m	1974-05-08	US-02332	1992-02-21
...	...	...	...	...

Left join merges on all of the common identifiers in the left specified table

A ← Left table

Customer	TransDate	Quantity	PurchAmount	Cost
149332	2005-11-15	1	199.95	107.00
172951	2008-08-29	1	199.95	108.00
120621	2007-10-19	1	99.95	49.00
149236	2005-11-14	1	39.95	18.95
149236	2007-12-06	1	79.95	35.00
...	...	...	...	...

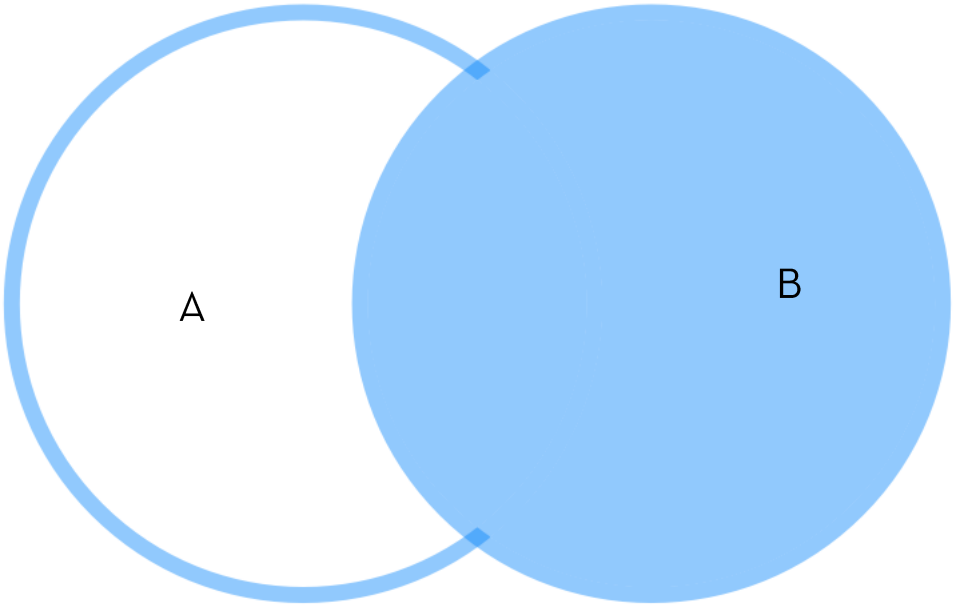
Right table → B

Customer	Gender	Birthdate	ZIP	JoinDate
80365	f	1991-08-26	US-06332	2009-09-15
149332	m	1998-07-07	US-08873	2005-11-05
84374	m	1977-07-10	US-06400	1988-08-10
149236	f	1955-08-15	US-92646	1971-02-16
100001	m	1974-05-08	US-02332	1992-02-21
...	...	...	...	...

Append column entries of table B if customer ID matches with customer ID of table A.

Customer	TransDate	Quantity	PurchAmount	Cost	Gender	Birthdate	ZIP	JoinDate
149332	2005-11-15	1	199.95	107.00	m	1998-07-07	US-08873	2005-11-05
172951	2008-08-29	1	199.95	108.00	NaN	NaT	NaN	NaT
120621	2007-10-19	1	99.95	49.00	NaN	NaT	NaN	NaT
149236	2005-11-14	1	39.95	18.95	m	1998-07-07	US-08873	2005-11-05
149236	2001-06-12	1	79.95	35.00	m	1998-07-07	US-08873	2005-11-05
...	...	...	...	...	...	...	...	...

Left joins are the same as right joins but the other way around



Common identifier.

```
A.merge(B, how="right", on="ID")
```

Use all observations from B, but only the matching from A.

Right join merges on all of the common identifiers in the table specified on the right side

A ← left table

Right table → B

Customer	TransDate	Quantity	PurchAmount	Cost
149332	2005-11-15	1	199.95	107.00
172951	2008-08-29	1	199.95	108.00
120621	2007-10-19	1	99.95	49.00
149236	2005-11-14	1	39.95	18.95
149236	2007-12-06	1	79.95	35.00
...	...	...	...	...

Customer	Gender	Birthdate	ZIP	JoinDate
80365	f	1991-08-26	US-06332	2009-09-15
149332	m	1998-07-07	US-08873	2005-11-05
84374	m	1977-07-10	US-06400	1988-08-10
149236	f	1955-08-15	US-92646	1971-02-16
100001	m	1974-05-08	US-02332	1992-02-21
...	...	...	...	...

Right join merges on all of the common identifiers in the table specified on the right side

A ← Left table

Customer	TransDate	Quantity	PurchAmount	Cost
149332	2005-11-15	1	199.95	107.00
172951	2008-08-29	1	199.95	108.00
120621	2007-10-19	1	99.95	49.00
149236	2005-11-14	1	39.95	18.95
149236	2007-12-06	1	79.95	35.00
...	...	...	...	...

Right table → B

Customer	Gender	Birthdate	ZIP	JoinDate
80365	f	1991-08-26	US-06332	2009-09-15
149332	m	1998-07-07	US-08873	2005-11-05
84374	m	1977-07-10	US-06400	1988-08-10
149236	f	1955-08-15	US-92646	1971-02-16
100001	m	1974-05-08	US-02332	1992-02-21
...	...	...	...	...

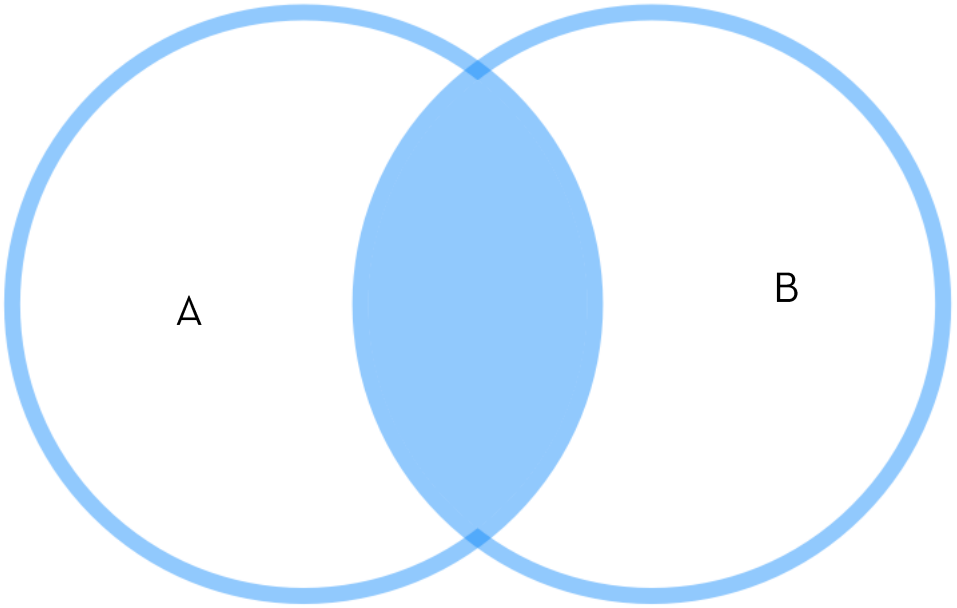
Append column entries of table A if customer ID matches with customer ID of table B.

Customer	TransDate	Quantity	PurchAmount	Cost	Gender	Birthdate	ZIP	JoinDate
80365	NaT	NaN	NaN	NaN	f	1991-08-26	US-06332	2009-09-15
149332	2005-11-15	1	199.95	107.00	m	1987-05-04	US-08873	2005-11-05
84374	NaT	NaN	NaN	NaN	m	1977-07-10	US-06400	1988-08-10
149236	2005-11-14	1	39.95	18.95	f	1955-08-15	US-92646	1971-02-16
100001	NaT	NaN	NaN	NaN	m	1974-05-08	US-02332	1992-02-21
...	...	...	...	...	...	...	...	...

When do we need left and right joins?

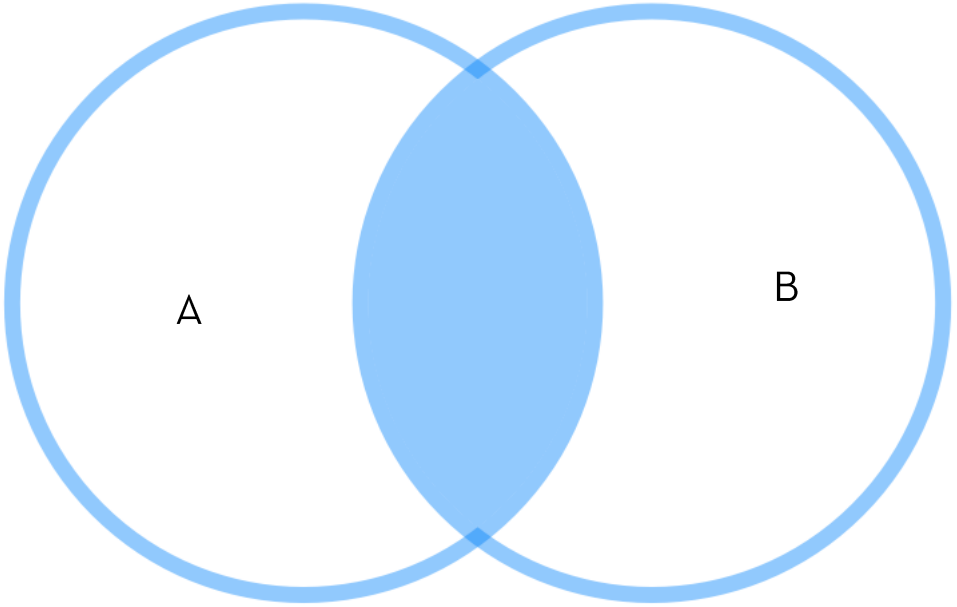
- Left and right joins return all the data available in the left or right DataFrame.
- For example:
  - We want all the data for all customers who made a transaction (left outer).
  - We want all the data for all the customers in our DataFrame (right outer).

Merging by multiple IDs




```
A.merge(B, on=["ID", "TransDate"])
```

IDs have different names



Common identifiers in A.

```
A.merge(B, how ="inner", left_on="ID", right_on="CustID")
```

Common identifiers in B.

Illustration of an inner join merged by multiple IDs

A ← left table

Customer	TransDate	Quantity	PurchAmount	Cost
149332	2005-11-15	1	199.95	107.00
172951	2008-08-29	1	199.95	108.00
120621	2007-10-19	1	99.95	49.00
149236	2005-11-14	1	39.95	18.95
149236	2007-12-06	1	79.95	35.00
...	...	...	...	...

Right table → B

Customer	Gender	Birthdate	ZIP	JoinDate
80365	f	1991-08-26	US-06332	2009-09-15
149332	m	1998-07-07	US-08873	2005-11-05
84374	m	1977-07-10	US-06400	1988-08-10
149236	f	1955-08-15	US-92646	1971-02-16
100001	m	1974-05-08	US-02332	1992-02-21
...	...	...	...	...

```
myData.merge(CustData, how="inner",
             left_on=["Customer","TransDate"],
             right_on=["Customer", "JoinDate"])
```

Illustration of an inner join merged by multiple IDs

A ← Left table

Customer	TransDate	Quantity	PurchAmount	Cost
149332	2005-11-15	1	199.95	107.00
172951	2008-08-29	1	199.95	108.00
120621	2007-10-19	1	99.95	49.00
149236	2005-11-14	1	39.95	18.95
149236	2007-12-06	1	79.95	35.00
...	...	...	...	...

Right table → B

Customer	Gender	Birthdate	ZIP	JoinDate
80365	f	1991-08-26	US-06332	2009-09-15
149332	m	1998-07-07	US-08873	2005-11-05
84374	m	1977-07-10	US-06400	1988-08-10
149236	f	1955-08-15	US-92646	1971-02-16
100001	m	1974-05-08	US-02332	1992-02-21
...	...	...	...	...

Merge rows with the same customer ID and the same transaction date (column name: "TransDate" in A, "JoinDate" in B) by an inner join.

Customer	TransDate	Quantity	PurchAmount	Cost	Gender	Birthdate	ZIP	JoinDate
170823	2008-04-04	2	109.90	45.20	m	1998-03-21	US-99515	2008-04-04
103853	2006-06-07	1	159.95	81.11	f	1990-08-10	US-94114	2006-06-07
102521	2005-09-14	1	159.95	120.00	m	1996-07-19	US-94803	2005-09-14
145017	2005-01-31	1	299.95	215.00	m	1987-11-14	US-29926	2005-01-31
145017	2005-01-31	1	0.00	0.00	m	1987-11-14	US-29926	2005-01-31
...	...	...	...	...	...	...	...	...

Exercise
Left and right outer joins

1. Merge transaction and demographics by “Customer” using an outer left join.
2. Merge transaction and demographics by “Customer” using an outer right join.
3. Merge transaction and demographics by “Customer” and by “TransDate “ resp. “JoinDate” using an **inner** join.