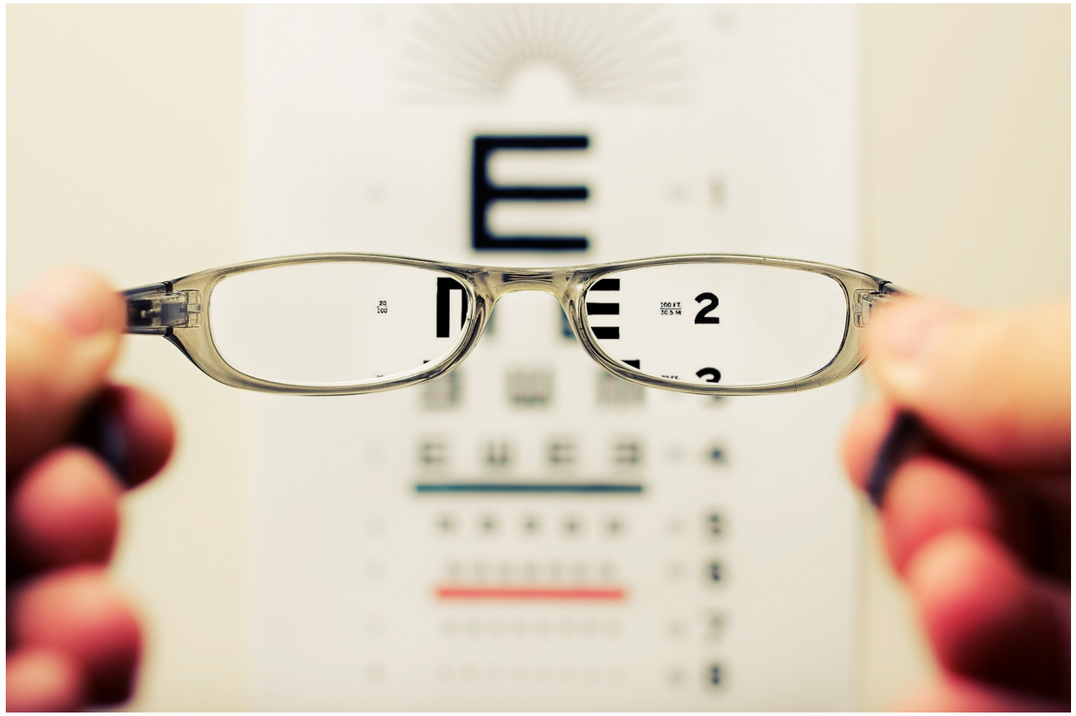
Inner join

JOINING DATA WITH PANDAS

Aaren Stubberfield
Instructor

datacamp

For clarity



Tables = DataFrames

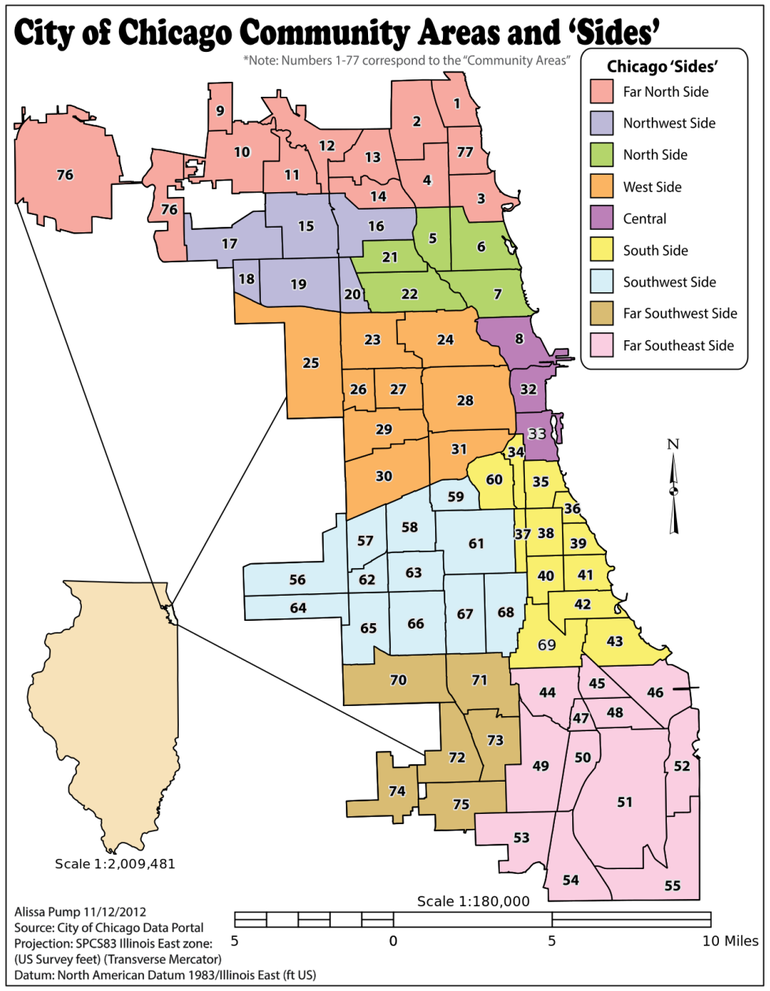
Merging = Joining

[1] Photo by David Travis on Unsplash

Chicago data portal dataset

[1] Photo by Pedro Lastra on Unsplash

Datasets for example



United States® Census Bureau

[1] Ward image By Alissapump, Own work, CC BY-SA 3.0

The ward data

```
wards = pd.read_csv('Ward_Offices.csv')
print(wards.head())
print(wards.shape)
```

```
   ward  alderman          address          zip
0  1     Proco "Joe" ...   2058 NORTH W...  60647
1  2     Brian Hopkins     1400 NORTH  ...  60622
2  3     Pat Dowell        5046 SOUTH S...  60609
3  4     William D. B...   435 EAST 35T...  60616
4  5     Leslie A. Ha...   2325 EAST 71...  60649
(50, 4)
```

Census data

```
census = pd.read_csv('Ward_Census.csv')
print(census.head())
print(census.shape)
```

```
   ward  pop_2000  pop_2010  change  address          zip
0  1     52951     56149     6%      2765 WEST SA...  60647
1  2     54361     55805     3%      WM WASTE MAN...  60622
2  3     40385     53039     31%     17 EAST 38TH...  60653
3  4     51953     54589     5%      31ST ST HARB...  60653
4  5     55302     51455     -7%     JACKSON PARK...  60637
(50, 6)
```

Merging tables

	ward	alderman	address	zip
0	1	Proco "Joe" ...	2058 NORTH W...	60647
1	2	Brian Hopkins	1400 NORTH ...	60622
2	3	Pat Dowell	5046 SOUTH S...	60609
3	4	William D. B...	435 EAST 35T...	60616
4	5	Leslie A. Ha...	2325 EAST 71...	60649

Ward Data

	ward	pop_2000	pop_2010	change	address	zip
0	1	52951	56149	6%	2765 WEST SA...	60647
1	2	54361	55805	3%	WM WASTE MAN...	60622
2	3	40385	53039	31%	17 EAST 38TH...	60653
3	4	51953	54589	5%	31ST ST HARB...	60653
4	5	55302	51455	-7%	JACKSON PARK...	60637

Census Data

Inner join

```
wards_census = wards.merge(census, on='ward')
print(wards_census.head(4))
```

```
  ward  alderman          address_x         zip_x  pop_2000  pop_2010  change  address_y         zip_y
0 1     Proco "Joe" ...   2058 NORTH W...   60647  52951     56149     6%      2765 WEST SA...   60647
1 2     Brian Hopkins     1400 NORTH  ...   60622  54361     55805     3%      WM WASTE MAN...   60622
2 3     Pat Dowell        5046 SOUTH S...   60609  40385     53039     31%     17 EAST 38TH...   60653
3 4     William D. B...   435 EAST 35T...   60616  51953     54589     5%      31ST ST HARB...   60653
```

```
print(wards_census.shape)
```

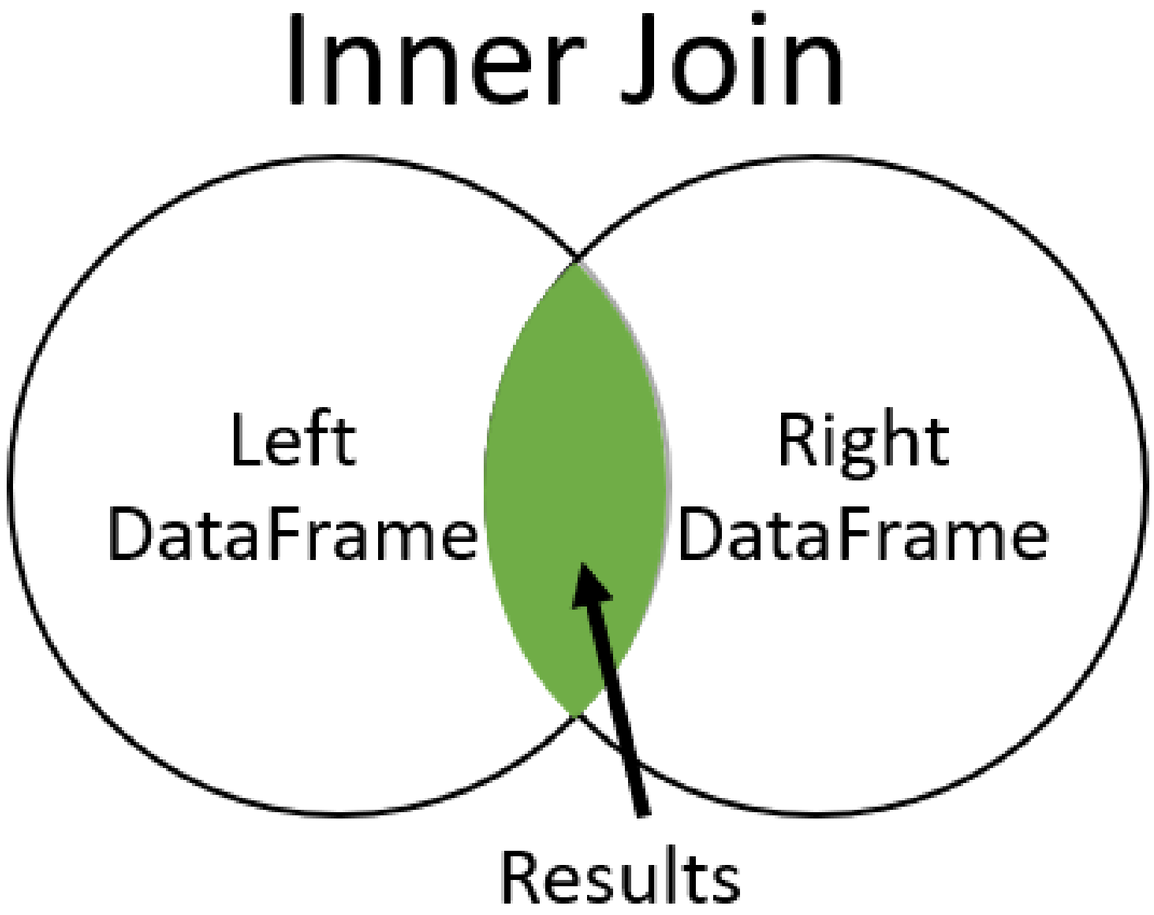
```
(50, 9)
```

8. Inner join

Since we listed the wards table first, its columns will appear first in the output, followed by the columns from the census table. For eg, the merge returns a DataFrame with 50 rows and 9 columns, where the returned rows have matching values for the ward column in both tables.

This is called an **inner join.**

Inner join



9. Inner join
An inner join will only return rows that have matching values in both tables.

Suffixes

```
print(wards_census.columns)
```

```
Index(['ward', 'alderman', 'address_x', 'zip_x', 'pop_2000', 'pop_2010', 'change',
       'address_y', 'zip_y'],
      dtype='object')
```

10. Suffixes
You may have noticed that the merged table has columns with suffixes of underscore x or y. This is because both the wards and census tables contained address and zip columns. To avoid multiple columns with the same name, they are automatically given a suffix by the merge method.

Suffixes

```
wards_census = wards.merge(census, on='ward', suffixes=('_ward','_cen'))
print(wards_census.head())
print(wards_census.shape)
```

```
  ward  alderman          address_ward     zip_ward  pop_2000  pop_2010  change  address_cen       zi
0 1     Proco "Joe" ...   2058 NORTH W...  60647     52951     56149     6%      2765 WEST SA...   60
1 2     Brian Hopkins     1400 NORTH  ...  60622     54361     55805     3%      WM WASTE MAN...   60
2 3     Pat Dowell        5046 SOUTH S...  60609     40385     53039     31%     17 EAST 38TH...   60
3 4     William D. B...   435 EAST 35T...  60616     51953     54589     5%      31ST ST HARB...   60
4 5     Leslie A. Ha...   2325 EAST 71...  60649     55302     51455     -7%     JACKSON PARK...   60
(50, 9)
```

11. Suffixes

We can use the suffix argument of the merge method to control this behavior. We provide a tuple where all of the overlapping columns in the left table are given the suffix '_ward', and those of the right table will be given the suffix '_cen'. This makes it easier for us to tell the difference between the columns.

Let's practice!

JOINING DATA WITH PANDAS

One to many relationships

JOINING DATA WITH PANDAS

Aaren Stubberfield
Instructor

datacamp

One-to-one

A	B	C
A1	B1	C1
A2	B2	C2
A3	B3	C3

C	D
C1	D1
C2	D2
C3	D3

One-To-One = Every row in the left table is related to only one row in the right table

One-to-one example

	ward	alderman	address	zip
0	1	Proco "Joe" ...	2058 NORTH W...	60647
1	2	Brian Hopkins	1400 NORTH ...	60622
2	3	Pat Dowell	5046 SOUTH S...	60609
3	4	William D. B...	435 EAST 35T...	60616
4	5	Leslie A. Ha...	2325 EAST 71...	60649

3. One-to-one example
We looked at a one-to-one relationship earlier. Recall the relationship between the wards table and the census table. Every row in the wards table is related to only one row in the census table, so there is only one row for ward 3 in each table. Practically speaking, it only makes sense that there is one row of population information for each ward. It wouldn't make sense if the census table contained multiple population values in 2000 for the third ward.

	ward	pop_2000	pop_2010	change	address	zip
0	1	52951	56149	6%	2765 WEST SA...	60647
1	2	54361	55805	3%	WM WASTE MAN...	60622
2	3	40385	53039	31%	17 EAST 38TH...	60653
3	4	51953	54589	5%	31ST ST HARB...	60653
4	5	55302	51455	-7%	JACKSON PARK...	60637

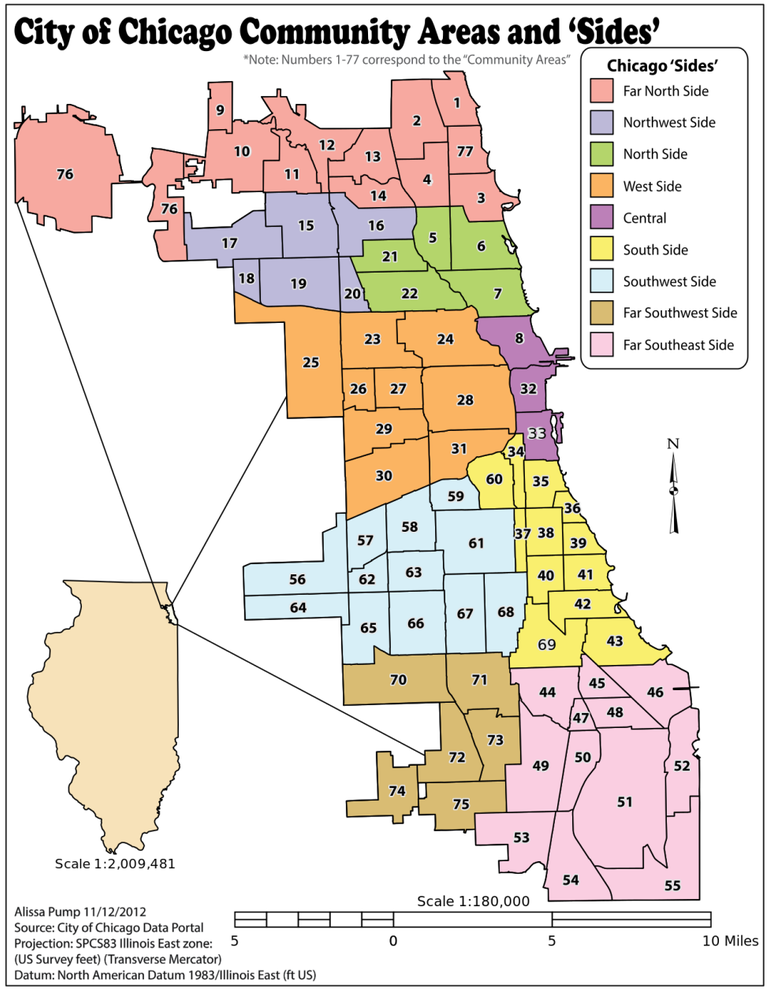
One-to-many

A	B	C
A1	B1	C1
A2	B2	C2
A3	B3	C3

C	D
C1	D1
C1	D2
C1	D3
C2	D4

One-To-Many = Every row in left table is related to one or more rows in the right table

One-to-many example



One-to-many example

```
licenses = pd.read_csv('Business_Licenses.csv')
print(licenses.head())
print(licenses.shape)
```

```
  account  ward  aid  business          address           zip
0 307071   3     743  REGGIE'S BAR...   2105 S STATE ST   60616
1 10       10    829  HONEYBEERS        13200 S HOUS...   60633
2 10002    14    775  CELINA DELI       5089 S ARCHE...   60632
3 10005    12    nan  KRAFT FOODS ...   2005 W 43RD ST    60609
4 10044    44    638  NEYBOUR'S TA...   3651 N SOUTH...   60613
(10000, 6)
```

One-to-many example

	ward	alderman	address	zip
0	1	Proco "Joe" ...	2058 NORTH W...	60647
1	2	Brian Hopkins	1400 NORTH ...	60622
2	3	Pat Dowell	5046 SOUTH S...	60609
3	4	William D. B...	435 EAST 35T...	60616
4	5	Leslie A. Ha...	2325 EAST 71...	60649

	account	ward	aid	business	address	zip
0	307071	3	743	REGGIE'S BAR...	2105 S STATE ST	60616
1	10	10	829	HONEYBEERS	13200 S HOUS...	60633
2	10002	14	775	CELINA DELI	5089 S ARCHE...	60632
3	10005	12	nan	KRAFT FOODS ...	2005 W 43RD ST	60609
4	10044	44	638	NEYBOUR'S TA...	3651 N SOUTH...	60613

One-to-many example

```
ward_licenses = wards.merge(licenses, on='ward', suffixes=('_ward','_lic'))
print(ward_licenses.head())
```

```
   ward  alderman          address_ward     zip_ward  account  aid  business          address_lic
0  1     Proco "Joe" ...   2058 NORTH W...  60647     12024    nan  DIGILOG ELEC...   1038 N ASHLA...
1  1     Proco "Joe" ...   2058 NORTH W...  60647     14446    743  EMPTY BOTTLE...   1035 N WESTE...
2  1     Proco "Joe" ...   2058 NORTH W...  60647     14624    775  LITTLE MEL'S...   2205 N CALIF...
3  1     Proco "Joe" ...   2058 NORTH W...  60647     14987    nan  MR. BROWN'S ...   2301 W CHICA...
4  1     Proco "Joe" ...   2058 NORTH W...  60647     15642    814  Beat Kitchen      2000-2100 W ...
```

8. One-to-many example
When we merge the two tables together with the merge method, setting the 'on' attribute to the column ward, the resulting table has both local ward data and business license data. Notice that ward 1 and its alderman Joe is repeated in the resulting table because the licenses table has many businesses in the 1st ward. Pandas takes care of the one-to-many relationships for us and doesn't require anything special on our end. We can use the same syntax as we did with one-to-one relationships.

One-to-many example

```
print(wards.shape)
```

```
(50, 4)
```

```
print(ward_licenses.shape)
```

```
(10000, 9)
```

9. One-to-many example
By printing the shape, we can see that our original wards table has 50 rows. After merging the wards table with the licenses table, the resulting table has 10,000 rows. When you merge tables that have a one-to-many relationship, the number of rows returned will likely be different than the number in the left table.

Let's practice!

JOINING DATA WITH PANDAS

Merging multiple DataFrames

JOINING DATA WITH PANDAS

Aaren Stubberfield
Instructor

datacamp

Merging multiple tables

A	B	C
A1	B1	C1
A2	B2	C2
A3	B3	C3

C	D
C1	D1
C2	D2
C3	D3

A	B	C
A1	B1	C1
A2	B2	C2
A3	B3	C3

C	E
C1	E1
C2	E2
C3	E3

E	F	G
E1	F1	G1
E2	F2	G2
E3	F3	G3

Remembering the licenses table

```
print(licenses.head())
```

```
  account  ward  aid  business          address           zip
0 307071   3     743  REGGIE'S BAR...   2105 S STATE ST   60616
1 10       10    829  HONEYBEERS        13200 S HOUS...   60633
2 10002    14    775  CELINA DELI       5089 S ARCHE...   60632
3 10005    12    nan  KRAFT FOODS ...   2005 W 43RD ST    60609
4 10044    44    638  NEYBOUR'S TA...   3651 N SOUTH...   60613
```

Remembering the wards table

```
print(wards.head())
```

```
   ward  alderman          address           zip
0  1     Proco "Joe" ...   2058 NORTH W...   60647
1  2     Brian Hopkins     1400 NORTH  ...   60622
2  3     Pat Dowell        5046 SOUTH S...   60609
3  4     William D. B...   435 EAST 35T...   60616
4  5     Leslie A. Ha...   2325 EAST 71...   60649
```

Review new data

```
grants = pd.read_csv('Small_Business_Grant_Agreements.csv')
print(grants.head())
```

```
  address           zip    grant       company
0 1000 S KOSTN...   60624  148914.50   NATIONWIDE F...
1 1000 W 35TH ST    60609  100000.00   SMALL BATCH,...
2 1000 W FULTO...   60612  34412.50    FULTON MARKE...
3 10008 S WEST...   60643  12285.32    LAW OFFICES ...
4 1002 W ARGYL...   60640  28998.75    MASALA'S IND...
```

Tables to merge

	address	zip	grant	company
0	1031 N CICER...	60651	150000.00	1031 HANS LLC
1	1375 W LAKE ST	60612	150000.00	1375 W LAKE ...
2	1800 W LAKE ST	60612	47700.00	1800 W LAKE LLC
3	4311 S HALST...	60609	87350.63	4311 S. HALS...
4	1747 W CARRO...	60612	50000.00	ACE STYLINE ...

	account	ward	aid	business	address	zip
0	307071	3	743	REGGIE'S BAR...	2105 S STATE ST	60616
1	10	10	829	HONEYBEERS	13200 S HOUS...	60633
2	10002	14	775	CELINA DELI	5089 S ARCHE...	60632
3	10005	12	nan	KRAFT FOODS ...	2005 W 43RD ST	60609
4	10044	44	638	NEYBOUR'S TA...	3651 N SOUTH...	60613

Theoretical merge

```
grants_licenses = grants.merge(licenses, on='zip')
print(grants_licenses.loc[grants_licenses['business']=="REGGIE'S BAR & GRILL",
                          ['grant','company','account','ward','business']])
```

```
   grant      company          account  ward  business
0  136443.07  CEDARS MEDIT...  307071   3     REGGIE'S BAR...
1  39943.15   DARRYL & FYL...  307071   3     REGGIE'S BAR...
2  31250.0    JGF MANAGEMENT   307071   3     REGGIE'S BAR...
3  143427.79  HYDE PARK AN...  307071   3     REGGIE'S BAR...
4  69500.0    ZBERRY INC       307071   3     REGGIE'S BAR...
```

7. Theoretical merge

If we merge the two tables only using the zip column, then the 60616 zip of Reggie's bar from the licenses table will be matched to multiple businesses in the grants table with the same zip. Our code sample prints the first few rows and some columns of the merged table. **The output of the merge duplicates Reggie's bar for each matching zip in the grants table, which is not what we want.** If instead, we merged on address only, there's a small risk that the address would repeat in different parts of the city. Therefore, the best option is to merge the tables using the combination of both address and zip code.

Single merge

```
grants.merge(licenses, on=['address','zip'])
```

	address	zip	grant	company	account	ward	aid	business
0	1020 N KOLMA...	60651	68309.8	TRITON INDUS...	7689	37	929	TRITON INDUS...
1	10241 S COMM...	60617	33275.5	SOUTH CHICAG...	246598	10	nan	SOUTH CHICAG...
2	11612 S WEST...	60643	30487.5	BEVERLY RECO...	3705	19	nan	BEVERLY RECO...
3	1600 S KOSTN...	60623	128513.7	CHARTER STEE...	293825	24	nan	LEELO STEEL,...
4	1647 W FULTO...	60612	5634.0	SN PECK BUIL...	85595	27	673	S.N. PECK BU...

8. Single merge
We merge the two DataFrames as shown before, except in this case, we pass a list of the column names we want to merge on to the 'on' argument. This allows us to use multiple columns in the merge. As before, the matching rows between the two DataFrames are returned with the columns from the grant table listed first. However, when we merge on two columns, in this case address and zip code, **we are requiring that both the address and zip code of a row in the left table match the address and zip code of a row in the right table in order for them to be linked to each other in the merge.**

Merging multiple tables

```
grants_licenses_ward = grants.merge(licenses, on=['address','zip']) \
                       .merge(wards, on='ward', suffixes=('_bus','_ward'))
grants_licenses_ward.head()
```

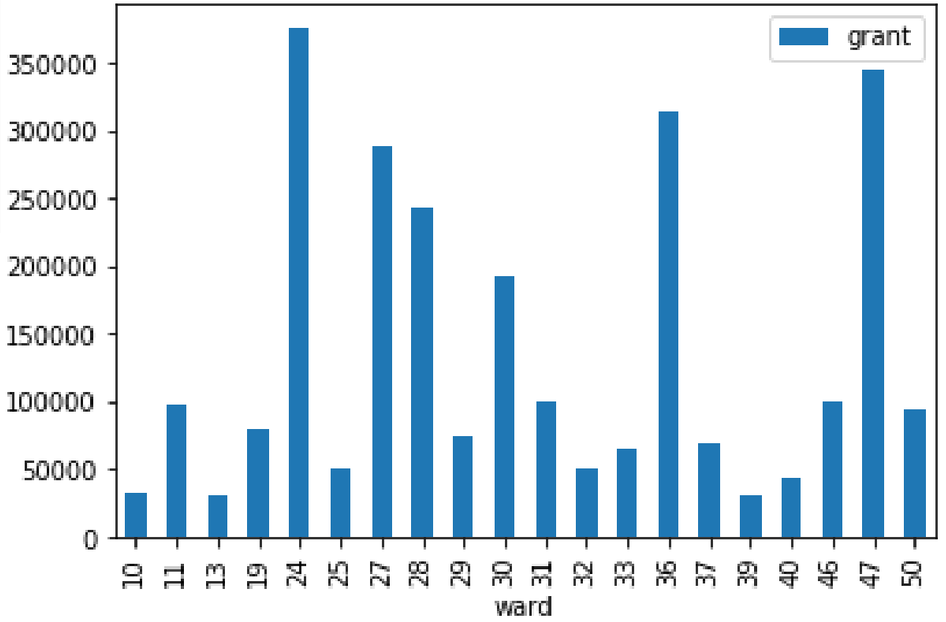
	address_bus	zip_bus	grant	company	account	ward	aid	business	alderma
0	1020 N KOLMA...	60651	68309.8	TRITON INDUS...	7689	37	929	TRITON INDUS...	Emma M.
1	10241 S COMM...	60617	33275.5	SOUTH CHICAG...	246598	10	nan	SOUTH CHICAG...	Susan S
2	11612 S WEST...	60643	30487.5	BEVERLY RECO...	3705	19	nan	BEVERLY RECO...	Matthew
3	3502 W 111TH ST	60655	50000.0	FACE TO FACE...	263274	19	704	FACE TO FACE	Matthew
4	1600 S KOSTN...	60623	128513.7	CHARTER STEE...	293825	24	nan	LEELO STEEL,...	Michael

9. Merging multiple tables
We can now extend this example to a third table. First, we merge the grants table with the wards table on the ward column again, adding suffixes to the repeated column names. Note that we're using Python's backslash line continuation method to add the second merge on the next line. Python will read this as just one line of code. Without this, Python will throw a syntax error since it will parse it as two separate lines of code, so don't forget your backslash. Now our output table has information about grants, business, and wards. We can now complete our analysis.

Results

```
import matplotlib.pyplot as plt
grant_licenses_ward.groupby('ward').agg('sum').plot(kind='bar', y='grant')
plt.show()
```

10. Results
We can now sum the grants by ward and plot the results. Some wards have received more grants than others.



Merging even more...

Three tables:

```
df1.merge(df2, on='col') \
    .merge(df3, on='col')
```

Four tables:

```
df1.merge(df2, on='col') \
    .merge(df3, on='col') \
    .merge(df4, on='col')
```

Let's practice!

JOINING DATA WITH PANDAS