DAT Class 6

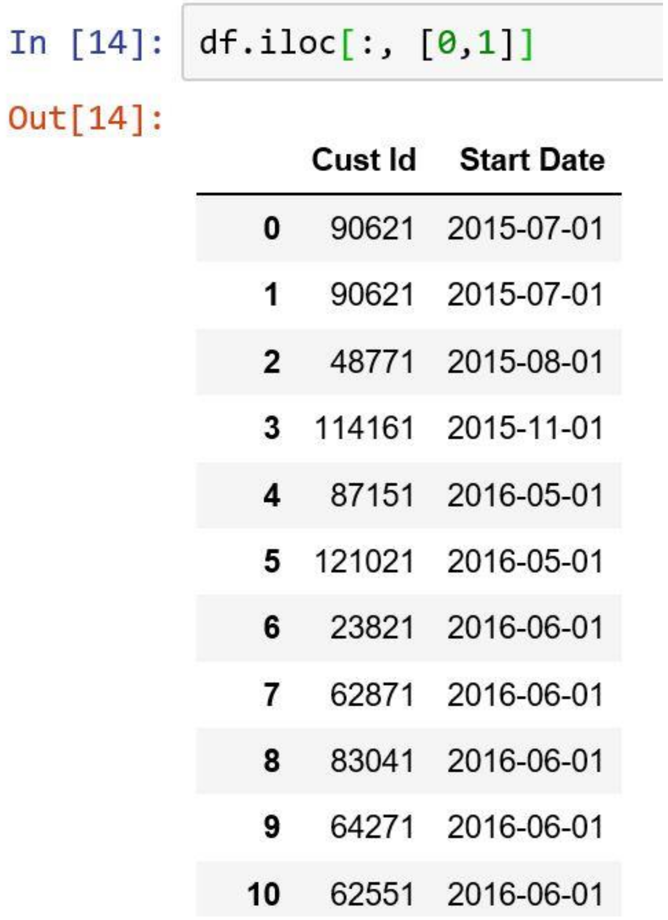
Extracting, Merging, Grouping

Creating and Manipulating Data in Pandas

Python For Data

Pandas: Creating Data

- Variables are manipulated using vectorized code
- Means you basically treat columns and datasets as a single variable
- A bit different from regular Python: no loops!



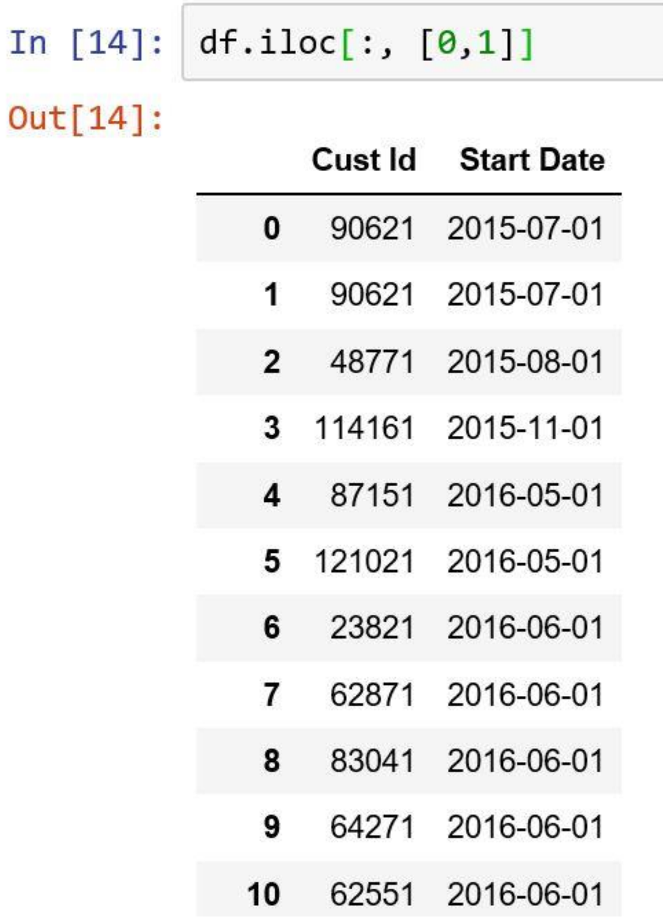
```
In [14]: df.iloc[:, [0,1]]
Out[14]:
```

	Cust Id	Start Date
0	90621	2015-07-01
1	90621	2015-07-01
2	48771	2015-08-01
3	114161	2015-11-01
4	87151	2016-05-01
5	121021	2016-05-01
6	23821	2016-06-01
7	62871	2016-06-01
8	83041	2016-06-01
9	64271	2016-06-01
10	62551	2016-06-01

Pandas: Creating Data

- You can think of pandas methods for creating data falling into two categories: general and particular.

- 3 general methods:
    - np.where()
    - np.select()
    - df.apply()

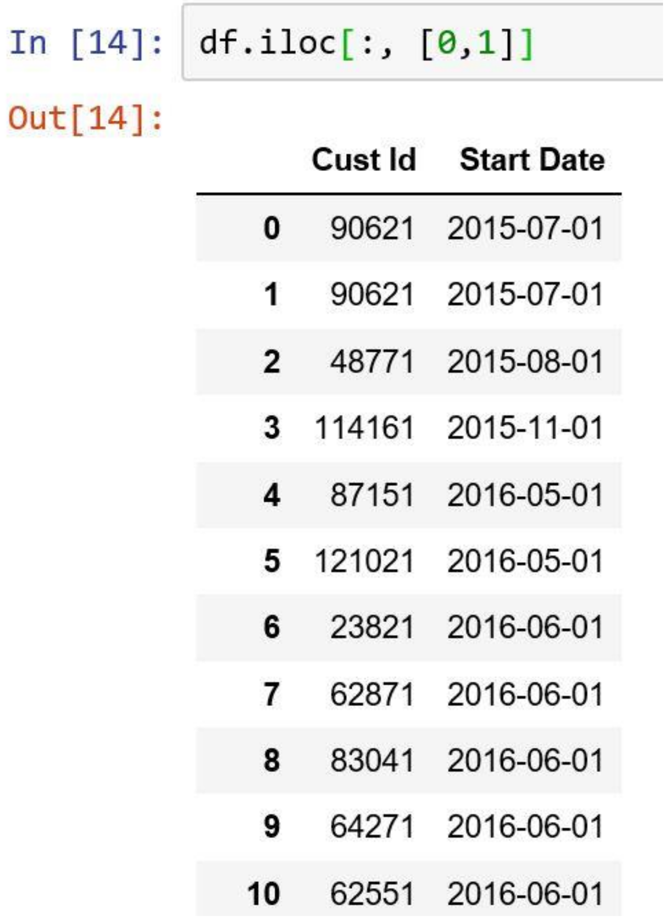


```
In [14]: df.iloc[:, [0,1]]
Out[14]:
```

	Cust Id	Start Date
0	90621	2015-07-01
1	90621	2015-07-01
2	48771	2015-08-01
3	114161	2015-11-01
4	87151	2016-05-01
5	121021	2016-05-01
6	23821	2016-06-01
7	62871	2016-06-01
8	83041	2016-06-01
9	64271	2016-06-01
10	62551	2016-06-01

Pandas: Creating Data

- np.where():
  - Simple way to create if/else statement to create new variables
- np.select():
  - Advanced version of np.where() - can create multiple conditions



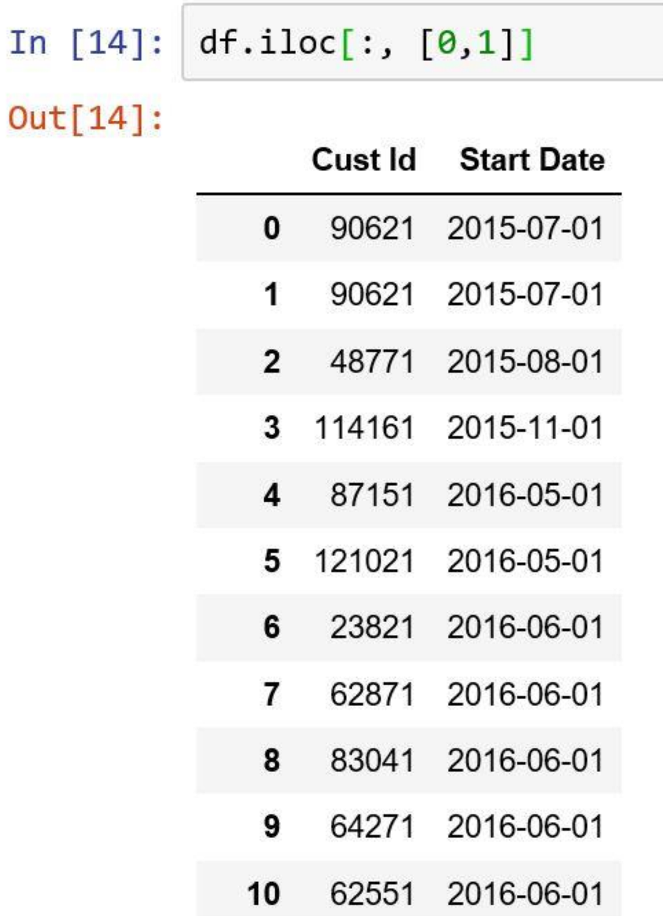
```
In [14]: df.iloc[:, [0,1]]

Out[14]:
```

	Cust Id	Start Date
0	90621	2015-07-01
1	90621	2015-07-01
2	48771	2015-08-01
3	114161	2015-11-01
4	87151	2016-05-01
5	121021	2016-05-01
6	23821	2016-06-01
7	62871	2016-06-01
8	83041	2016-06-01
9	64271	2016-06-01
10	62551	2016-06-01

Pandas: Creating Data

Methods for specific ways of changing data:

- pd.cut() – turn a numeric category into predefined bins
- df.map() – change values in a column from one to another
- df.replace() – replace specific values with something else



```
In [14]: df.iloc[:, [0,1]]
Out[14]:
```

	Cust Id	Start Date
0	90621	2015-07-01
1	90621	2015-07-01
2	48771	2015-08-01
3	114161	2015-11-01
4	87151	2016-05-01
5	121021	2016-05-01
6	23821	2016-06-01
7	62871	2016-06-01
8	83041	2016-06-01
9	64271	2016-06-01
10	62551	2016-06-01

Group Exercise: Section 1 of Lab

Take 10-12 minutes and complete section I of the accompanying lab.

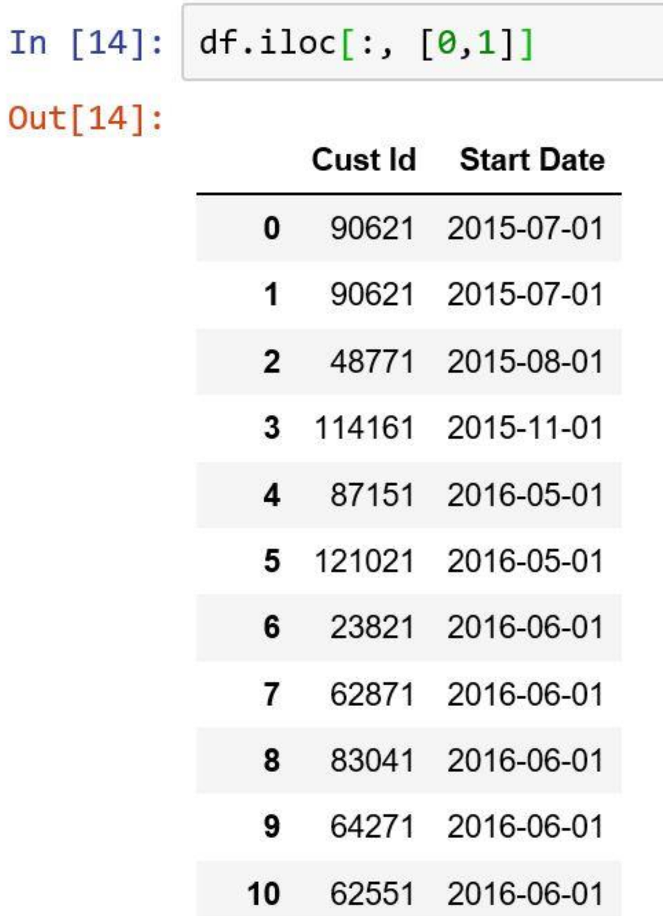
Merging Data

Python For Data

Pandas: Merging Data

Pandas allows you to combine multiple dataframes together

- Very similar to UNION and JOIN in SQL
- Corresponding methods are:
    - df.merge
    - pd.concat



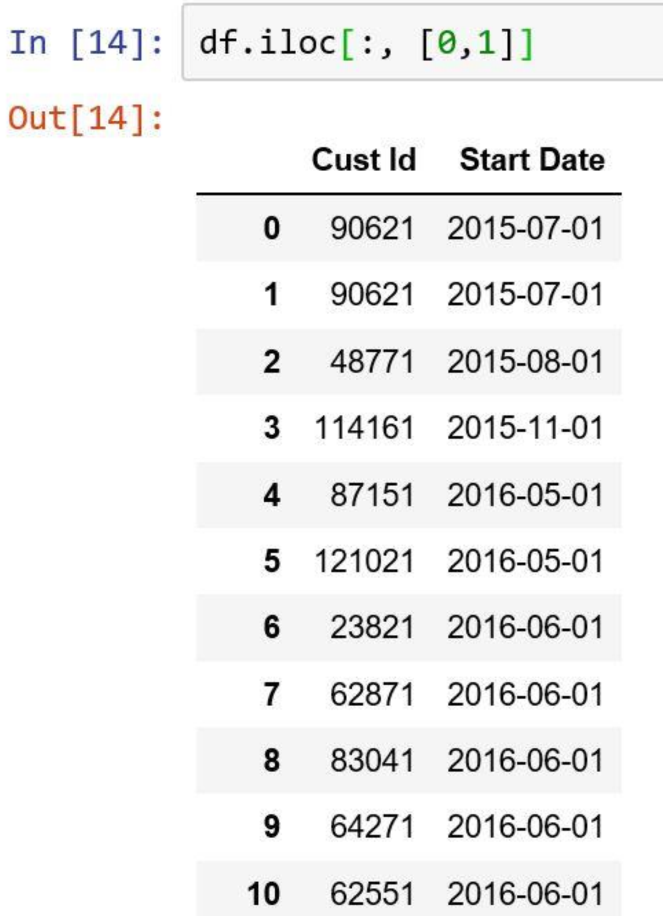
```
In [14]: df.iloc[:, [0,1]]

Out[14]:
```

	Cust Id	Start Date
0	90621	2015-07-01
1	90621	2015-07-01
2	48771	2015-08-01
3	114161	2015-11-01
4	87151	2016-05-01
5	121021	2016-05-01
6	23821	2016-06-01
7	62871	2016-06-01
8	83041	2016-06-01
9	64271	2016-06-01
10	62551	2016-06-01

Pandas: Merging Data

df.merge():

- similar to JOIN
- important arguments:
- **on:** column to join on
  defaults to joining on **every** column
  can also choose subsets
- **how:** how to do the JOIN/merge:
  inner, outer, left, right
- **left_on/right_on:** which columns to use for each dataframe
- **left_index/right_index:** set to True if you want to merge on index



```
In [14]: df.iloc[:, [0,1]]
Out[14]:
```

	Cust Id	Start Date
0	90621	2015-07-01
1	90621	2015-07-01
2	48771	2015-08-01
3	114161	2015-11-01
4	87151	2016-05-01
5	121021	2016-05-01
6	23821	2016-06-01
7	62871	2016-06-01
8	83041	2016-06-01
9	64271	2016-06-01
10	62551	2016-06-01

Group Exercise: Section 2 of Lab

Take 10-12 minutes and complete section II of the accompanying lab.

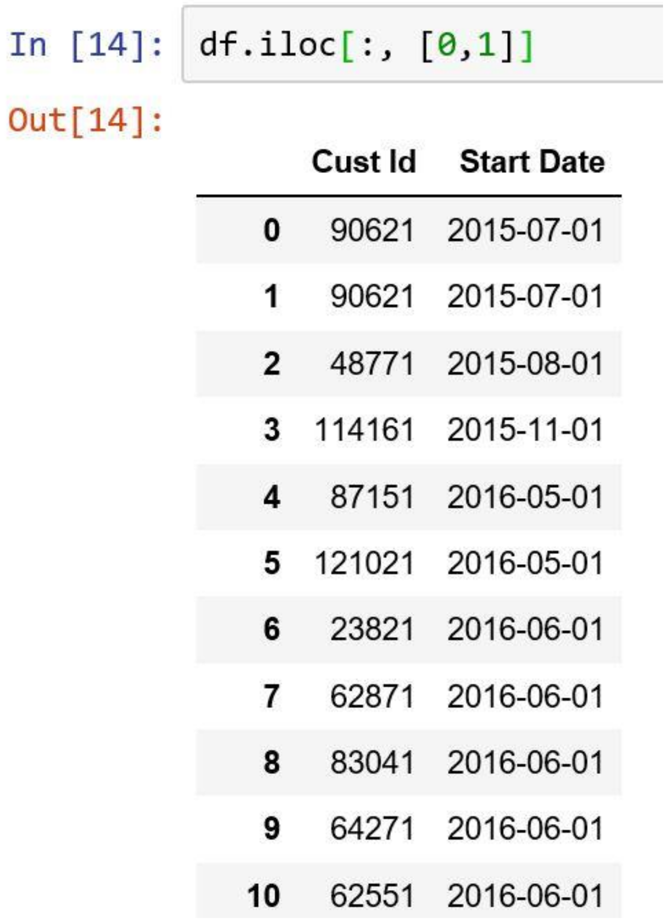
Grouping Data

Python For Data

Pandas: Grouping Data

You can *squash* data into discrete categorie
to get summary statistics for different levels
within a particular column.

- Similar to GROUPBY in SQL or pivot tables in Excel
- Two primary ways you do this in pandas:
    - Groupby(): the name says it all
    - Resample: Allows you to group using continuous time



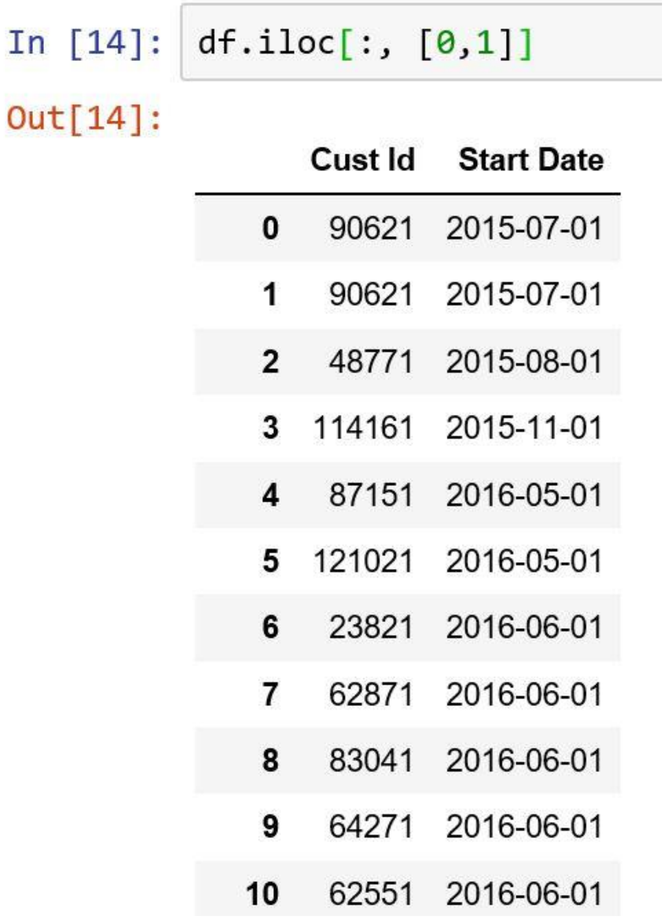
```
In [14]: df.iloc[:, [0,1]]
Out[14]:
```

	Cust Id	Start Date
0	90621	2015-07-01
1	90621	2015-07-01
2	48771	2015-08-01
3	114161	2015-11-01
4	87151	2016-05-01
5	121021	2016-05-01
6	23821	2016-06-01
7	62871	2016-06-01
8	83041	2016-06-01
9	64271	2016-06-01
10	62551	2016-06-01

Pandas: Grouping Data

You can *squash* data into discrete categories to get summary statistics for different levels within a particular column.

- Similar to GROUPBY in SQL or pivot tables in Excel
- Two primary ways you do this in pandas:
    - Groupby(): the name says it all
    - Resample: Allows you to group using continuous time



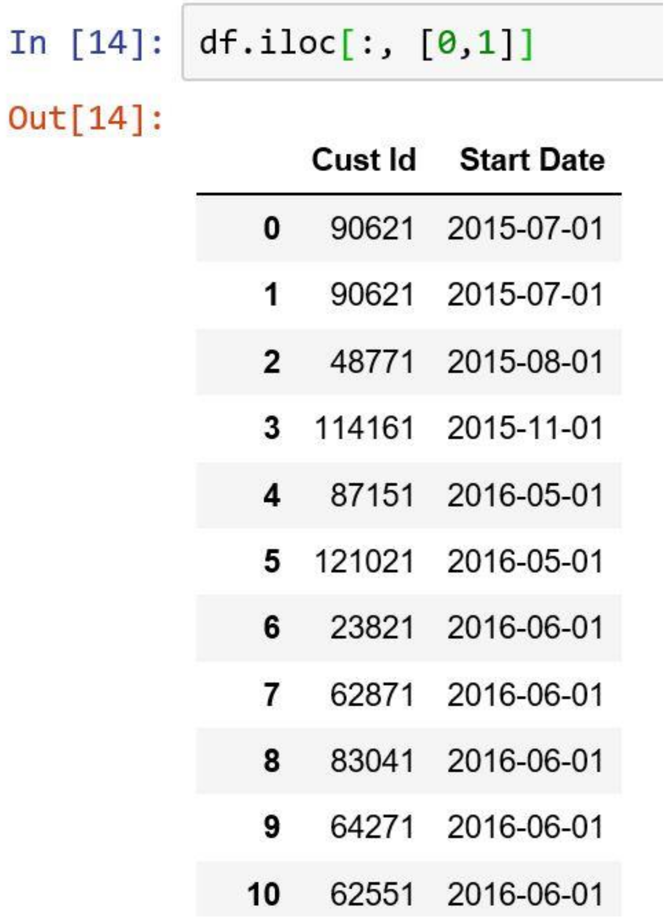
```
In [14]: df.iloc[:, [0,1]]

Out[14]:
```

	Cust Id	Start Date
0	90621	2015-07-01
1	90621	2015-07-01
2	48771	2015-08-01
3	114161	2015-11-01
4	87151	2016-05-01
5	121021	2016-05-01
6	23821	2016-06-01
7	62871	2016-06-01
8	83041	2016-06-01
9	64271	2016-06-01
10	62551	2016-06-01

Pandas: Grouping Data

A pandas groupby statement has the following characteristics:

- Columns that you are grouping on:
  - Can be either single values or a list
- Columns that you are selecting from your groupby object
- Aggregators that you are returning
  - Can either be pre-specified or custom
  - Can be single or multiple



```
In [14]: df.iloc[:, [0,1]]
```

Out[14]:

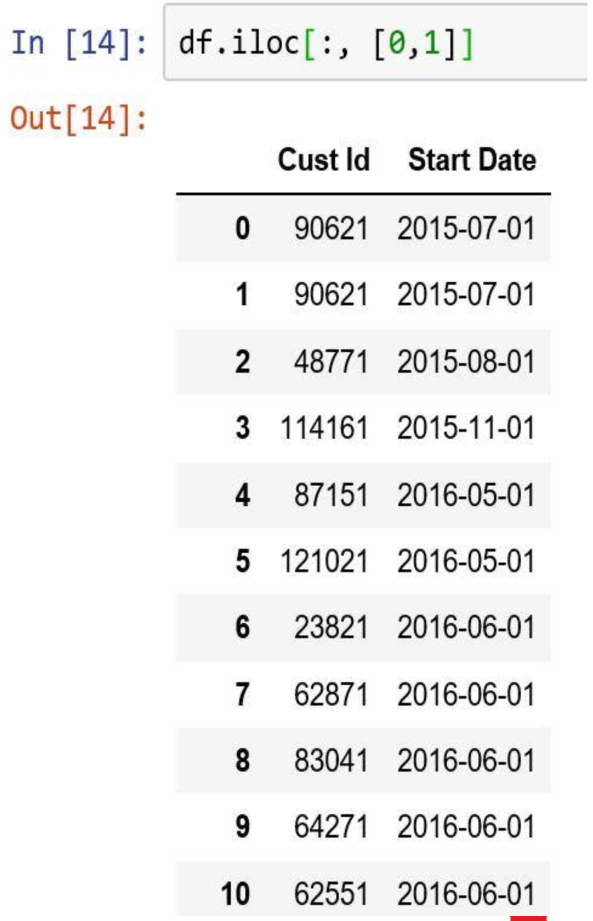
	Cust Id	Start Date
0	90621	2015-07-01
1	90621	2015-07-01
2	48771	2015-08-01
3	114161	2015-11-01
4	87151	2016-05-01
5	121021	2016-05-01
6	23821	2016-06-01
7	62871	2016-06-01
8	83041	2016-06-01
9	64271	2016-06-01
10	62551	2016-06-01

Pandas: Grouping Data

Resampling:

Similar to groupby, except it allows you to apply it to continuous time.

Gives you the ability to define a fairly particular time sequence to view data on. Complete list is here: http://bit.ly/date-offsets



```
In [14]: df.iloc[:, [0,1]]
Out[14]:
```

	Cust Id	Start Date
0	90621	2015-07-01
1	90621	2015-07-01
2	48771	2015-08-01
3	114161	2015-11-01
4	87151	2016-05-01
5	121021	2016-05-01
6	23821	2016-06-01
7	62871	2016-06-01
8	83041	2016-06-01
9	64271	2016-06-01
10	62551	2016-06-01

Group Exercise: Section 1 of Lab

Take 10-12 minutes and complete section III of the accompanying lab.