

AI Bootcamp

---

# Advanced Data Reshaping with Pandas

Module 5 Day 3

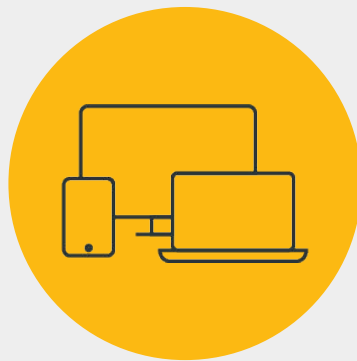


# Class Objectives

By the end of class, you will be able to:

---

- 1 Understand the concept of pivoting data and explain its role in data analysis.
- 2 Differentiate between single and multiple aggregations when reshaping data.
- 3 Apply one or more aggregation functions to reshaped data.
- 4 Use the **aggfunc** function effectively to perform various aggregations on reshaped data.
- 5 Use custom Python functions to transform reshaped data.
- 6 Reshape data into multi-index and apply aggregations.
- 7 Reshape data using `resample` and `melt`.

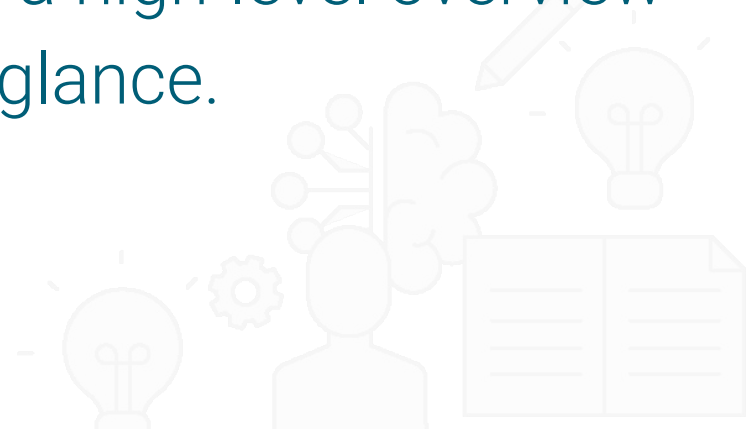


# Instructor **Demonstration**

Pivoting



Pivot tables allow the aggregation and summarization of data based on your specific requirements, providing the potential for a high-level overview of the data at a glance.



## Parameters of the pivot Function:

```
pd.pivot(data, columns, index=<a column>, values=<a column>,
```

**or**

```
df.pivot(columns, index=<a column>, values=<a column>
```

# The pivot Function

```
# Pivot on the date_ending with the book_name as the index, and pass the "total_sales" as the values.
pivot_date_short_form = pd.pivot(book_sales_df,
                                  columns="date_ending",
                                  index="book_name",
                                  values="total_sales" )

# Show the table.
pivot_date_short_form
```

date_ending	10/31/23	11/30/23	12/31/23	8/31/23	9/30/23
book_name					
Foundation	75	50	125	60	100
Foundation and Earth	25	30	20	0	20
Foundation and Empire	75	60	75	40	50
Foundation's Edge	25	30	30	20	30
Second Foundation	50	100	50	35	40
The Fellowship of the Ring (The Lord of the Rings, Part 1)	125	100	175	80	150
The Hobbit	200	200	250	150	100
The Return of the King (The Lord of the Rings, Part 3)	200	50	200	100	125
The Two Towers (The Lord of the Rings, Part 2)	225	100	150	100	75

# The pivot Function

```
# Pivot on the book_name with the date_ending as the index, and pass the "total_sales" as the values.
pivot_books_long_form = pd.pivot(book_sales_df,
                                  columns="book_name",
                                  index="date_ending",
                                  values="total_sales" )
pivot_books_long_form
```

book_name	Foundation	Foundation and Earth	Foundation and Empire	Foundation's Edge	Second Foundation	The Fellowship of the Ring (The Lord of the Rings, Part 1)	The Hobbit	The Return of the King (The Lord of the Rings, Part 3)	The Two Towers (The Lord of the Rings, Part 2)
date_ending									
10/31/23	75	25	75	25	50	125	200	200	225
11/30/23	50	30	60	30	100	100	200	50	100
12/31/23	125	20	75	30	50	175	250	200	150
8/31/23	60	0	40	20	35	80	150	100	100
9/30/23	100	20	50	30	40	150	100	125	75

## **Parameters of the pivot\_table() Function:**

```
pd.pivot_table(data, values=None, index=None,  
columns=None, aggfunc='mean', fill_value=None,  
margins=False, dropna=True, margins_name='All',  
observed=False, sort=True)
```



# The pivot\_table Function

```
# Using the `pivot_table()` function, get the total book sales for each book.
pivot_table_books_sum = pd.pivot_table(book_sales_df,
                                       values='total_sales',
                                       columns='book_name',
                                       aggfunc='sum')

# Show the table.
pivot_table_books_sum
```

book_name	Foundation	Foundation and Earth	Foundation and Empire	Foundation's Edge	Second Foundation	The Fellowship of the Ring (The Lord of the Rings, Part 1)	The Hobbit	The Return of the King (The Lord of the Rings, Part 3)	The Two Towers (The Lord of the Rings, Part 2)
total_sales	410	95	300	135	275	630	900	675	650

# The pivot\_table Function

```
# Get the total and average book sales for each book.
# Make the books the columns, and the mean and sum of the total sales under each book.
avg_sum_books = pd.pivot_table(book_sales_df,
                                values='total_sales',
                                columns='book_name',
                                aggfunc=('sum', 'mean'))

# Show the table.
avg_sum_books
```

book_name	Foundation	Foundation and Earth	Foundation and Empire	Foundation's Edge	Second Foundation	The Fellowship of the Ring (The Lord of the Rings, Part 1)	The Hobbit	The Return of the King (The Lord of the Rings, Part 3)	The Two Towers (The Lord of the Rings, Part 2)
mean	82.0	19.0	60.0	27.0	55.0	126.0	180.0	135.0	130.0
sum	410.0	95.0	300.0	135.0	275.0	630.0	900.0	675.0	650.0

# The pivot\_table Function

```
# Using the pivot_table function get the average and the total of the book sales
# for each date. Make the date the index and round to one decimal place.
date_ending_pivot_table = book_sales_df.pivot_table(index="date_ending",
                                                    values="total_sales",
                                                    aggfunc=('mean', 'sum')).round(1)

# Show the table.
date_ending_pivot_table
```

	mean	sum
date_ending		
10/31/23	111.1	1000
11/30/23	80.0	720
12/31/23	119.4	1075
8/31/23	65.0	585
9/30/23	76.7	690



## Activity:

### Pivoting Student Exam Scores

---

In this activity, you will use the `pivot_table()` function on a DataFrame to reshape exam score data.

**Suggested Time:**

15 Minutes



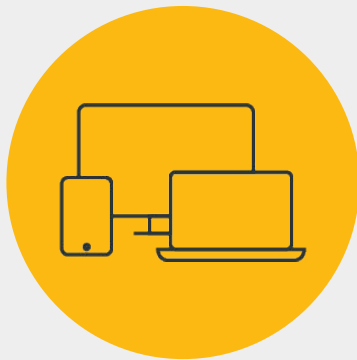


**Time's up!**  
Let's review



**Questions?**





# Instructor **Demonstration**

Pivoting with Multi-Index and Multi-Aggregations

# Pivoting with Multi-Index

```
# Show the average seconds for each country and state and
# round to one decimal place.
ufo_country_state = pd.pivot_table(converted_ufo_df,
                                    index=['country', 'state'],
                                    values='duration (seconds)',
                                    aggfunc='mean').round(1)

# Show the table.
ufo_country_state.head(20)
```

		duration (seconds)
country	state	
au	al	900.0
	dc	300.0
	nt	180.0
	oh	180.0
	sa	152.5
	wa	225.0
	yt	30.0
ca	ab	1869.7
	bc	948.2
	mb	1291.4
	nb	668.3
	nf	1250.7



# Pivoting with Multi-Index

```
# Show the number of UFOs for each country, state, and city.
ufo_country_state_city_cnt =
pd.pivot_table(converted_ufo_df,
                index=['country','state','city'],
                values='shape',
                aggfunc='count')

# Show the table.
ufo_country_state_city_cnt.head(20)
```

			shape
country	state	city	
au	al	melbourne (australia)	1
	dc	maroochydore (queensland) (australia)	1
	nt	darwin (nt&#44 australia)	2
	oh	adelaide (south australia)	1
	sa	adelaide (south australia)	1
		port adelaide (south australia)	1
	wa	cue (western australia) (australia)	1
		perth (western australia)	1
	yt	port macquarie (australia)	1
ca	ab	airdrie (canada)	10

# Pivoting with Multi-Index and Applying Multi-Aggregations

```
# Show the minimum and maximum seconds for each country and state.  
ufo_country_state_mean_sum = pd.pivot_table(converted_ufo_df,  
      index=['country','state'],  
      values='duration (seconds)',  
      aggfunc=('min', 'max'))  
  
# Show the table.  
ufo_country_state_mean_sum.head(10)
```

		max	min
country	state		
au	al	900.0	900.00
	dc	300.0	300.00
	nt	300.0	60.00
	oh	180.0	180.00
	sa	300.0	5.00
	wa	420.0	30.00
	yt	30.0	30.00
ca	ab	259200.0	1.00
	bc	37800.0	0.02
	mb	36000.0	1.00



## Activity:

### Pivoting with Multi-Index and Multi-Aggregations on Car Sales

---

In this activity, you will practice reshaping data on multiple indices and gain insights on car sales using aggregations.

**Suggested Time:**

15 Minutes





**Time's up!**  
Let's review



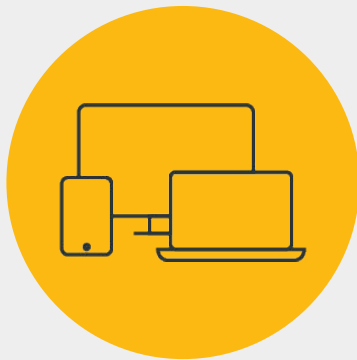
**Questions?**





**Break**

15 mins



# Instructor **Demonstration**

Pivoting with Custom Aggregations

# Pivoting and Applying a Custom Aggregation

```
# Create a custom function that will calculate the average
# of the DataFrame column.
def custom_avg(x):
    return x.mean()

# Use the custom_avg function to show the average seconds
# for each country and state and round to one decimal place.
ufo_country_state = pd.pivot_table(converted_ufo_df,
    index=['country', 'state'],
    values='duration (seconds)',
    aggfunc=custom_avg).round(1)

ufo_country_state.head(20)
```

		duration (seconds)
country	state	
au	al	900.0
	dc	300.0
	nt	180.0
	oh	180.0
	sa	152.5
	wa	225.0
	yt	30.0
ca	ab	1869.7
	bc	948.2
	mb	1291.4
	nb	668.3
	nf	1250.7



# Pivoting and Applying Multiple Custom Aggregations

```
# Create a function that checks if the number of sightings in the "duration (seconds)"
# column are greater than 20.
def custom_count(x):
    if x.count()>20:
        return x.count()

# Show the number of UFOs for each city, state, and country.
# Sort in ascending order.
ufo_country_state_metrics = pd.pivot_table(converted_ufo_df,
    index=['country','state'],
    values='duration (seconds)',
    aggfunc=(custom_count, custom_avg, custom_sum)).round(1)

# Display the results.
ufo_country_state_metrics

# Drop the null values.
ufo_country_state_metrics.dropna(how="any")
```

# Pivoting and Applying Multiple Custom Aggregations

		custom_avg	custom_count	custom_sum
country	state			
ca	ab	1869.7	284.0	530994.0
	bc	948.2	677.0	641955.8
	mb	1291.4	124.0	160132.0
	nb	668.3	86.0	57473.0
	ns	1383.3	101.0	139710.0
...	...	...	...	...
us	vt	1042.5	254.0	264785.5
	wa	15273.5	3707.0	56618769.4
	wi	1928.4	1205.0	2323749.3
	wv	6791.9	438.0	2974853.0
	wy	1487.8	169.0	251443.0



## Activity:

### Custom Aggregations Pivoting Delayed Flights

---

In this activity, you will reshape data and apply custom functions for aggregations to gain insights into airline flight delays.

**Suggested Time:**

15 Minutes



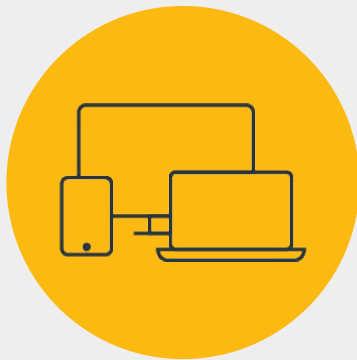


**Time's up!**  
Let's review



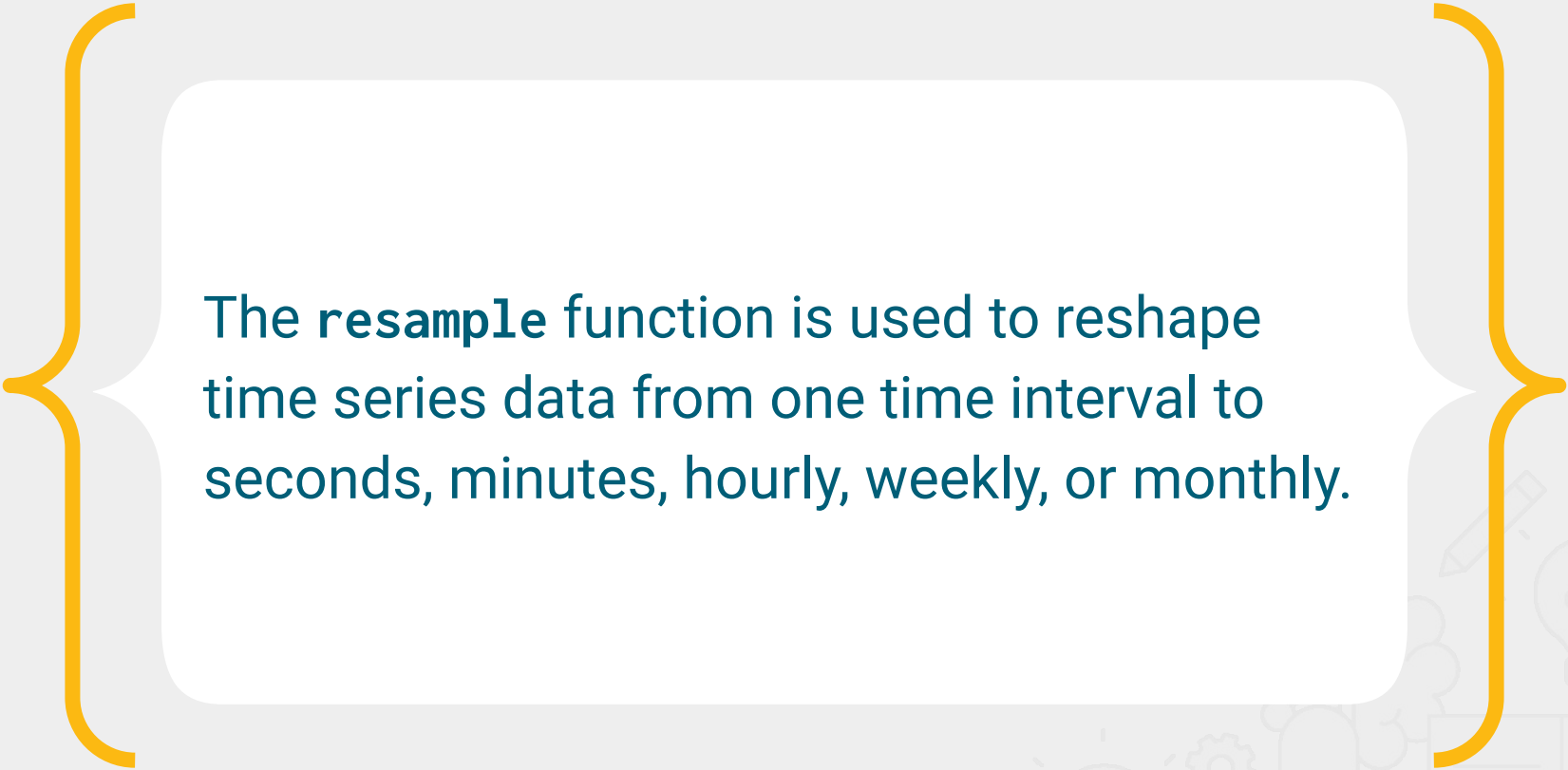
**Questions?**



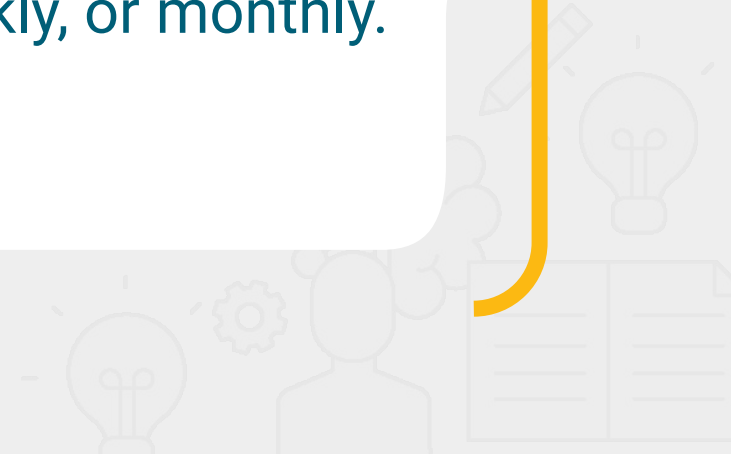


# Instructor **Demonstration**

Reshaping Data with Resample and Melt



The `resample` function is used to reshape time series data from one time interval to seconds, minutes, hourly, weekly, or monthly.



# Resampling Data into Weekly Bins

```
# Get the total visits for each week.  
sales_df.resample('W').sum()
```

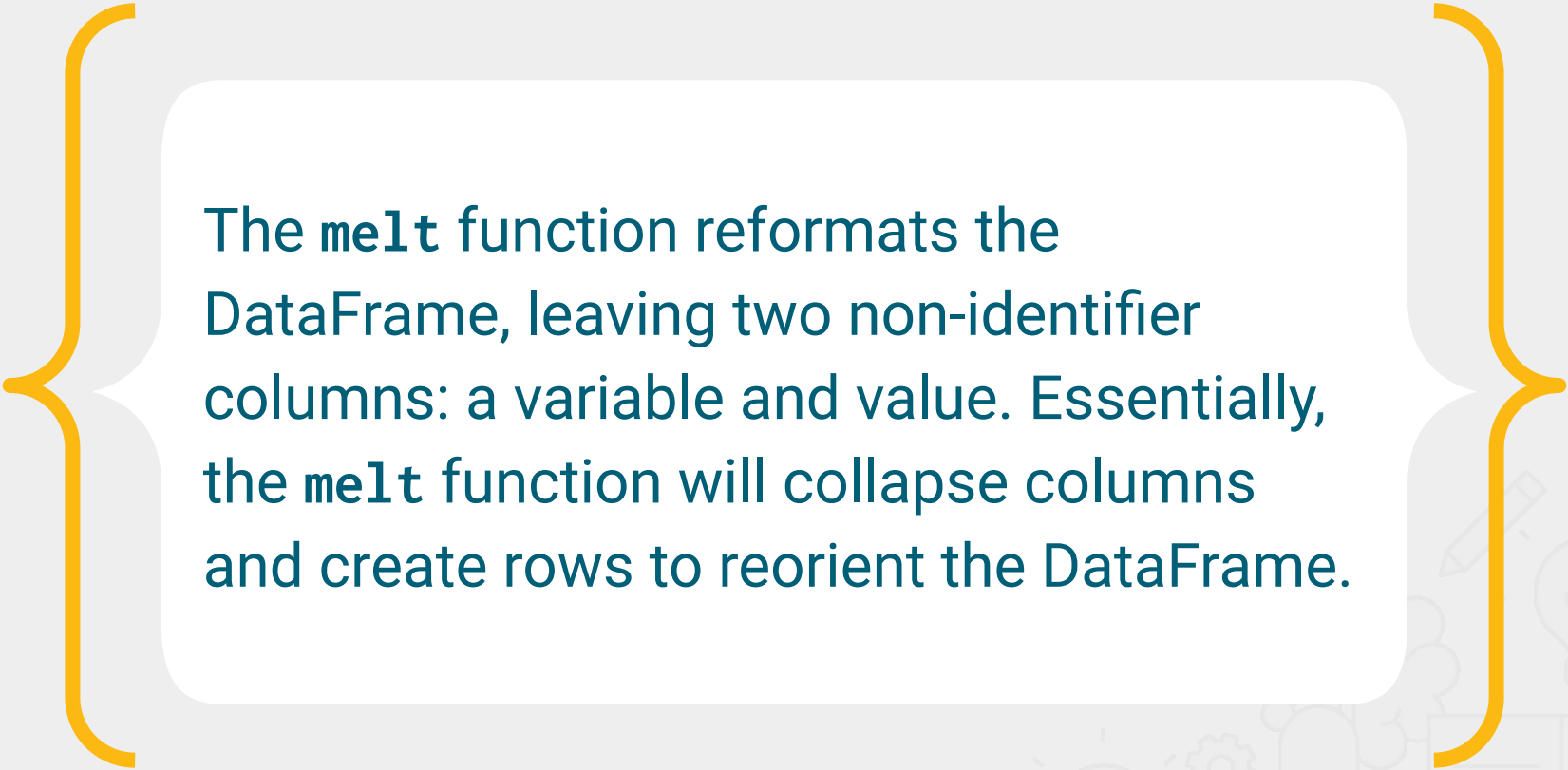
visits	
date	
2023-01-01	30
2023-01-08	219
2023-01-15	200
2023-01-22	211
2023-01-29	236
2023-02-05	187
2023-02-12	263
2023-02-19	193
2023-02-26	243
2023-03-05	211
2023-03-12	248
2023-03-19	204
2023-03-26	220
2023-04-02	170



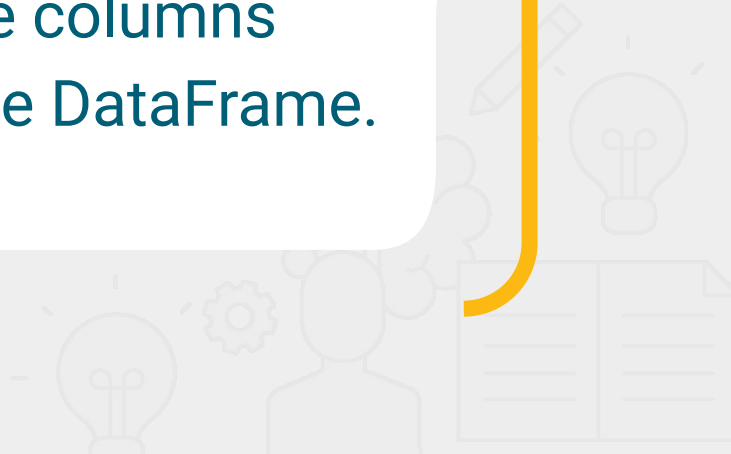
# Resampling Data into Monthly Bins

```
# Get the total visits for each month.  
sales_df.resample('M').sum()
```

visits	
date	
2023-01-31	962
2023-02-28	897
2023-03-31	976



The `melt` function reformats the DataFrame, leaving two non-identifier columns: a variable and value. Essentially, the `melt` function will collapse columns and create rows to reorient the DataFrame.






# Resampling Data into Monthly Bins

	Alcina	Clem	Frank	Kati	Loris	Marinella	Sloane	Uwe
subject								
Biology	85	85	96	76	69	72	90	80
Composition	74	70	96	95	85	69	84	87
Math	94	78	88	70	74	82	77	93
Speech	73	90	95	82	77	93	81	97



Reshaping a DataFrame with the `melt` function is best used on a numerical index.



# Reshaping Data with the `melt` Function

```
# Reset the index so "subject" is a column.  
exam_scores_reindexed = exam_scores_subject.reset_index()  
exam_scores_reindexed
```

	subject	Alcina	Clem	Frank	Kati	Loris	Marinella	Sloane	Uwe
0	Biology	85	85	96	76	69	72	90	80
1	Composition	74	70	96	95	85	69	84	87
2	Math	94	78	88	70	74	82	77	93
3	Speech	73	90	95	82	77	93	81	97

# Reshaping Data with the `melt` Function

```
# Convert the DataFrame from short form to long form.  
# Melt the DataFrame.  
exam_scores_reindexed.melt()
```

	variable	value
0	subject	Biology
1	subject	Composition
2	subject	Math
3	subject	Speech
4	Alcina	85
5	Alcina	74
6	Alcina	94
7	Alcina	73
8	Clem	85
9	Clem	70
10	Clem	78

# Reshaping Data with the `melt` Function

```
# Melt the DataFrame and rename the columns.  
melted_df = exam_scores_reindexed.melt(id_vars="subject",  
                                         var_name="student_name",  
                                         value_name="exam_score")
```

```
melted_df
```

	subject	student_name	exam_score
0	Biology	Alcina	85
1	Composition	Alcina	74
2	Math	Alcina	94
3	Speech	Alcina	73
4	Biology	Clem	85
5	Composition	Clem	70
6	Math	Clem	78
7	Speech	Clem	90
8	Biology	Frank	96
9	Composition	Frank	96

# Applying groupby to a Melted DataFrame

```
# Group the melted DataFrame on the subject to get the average exam score rounded to one decimal place.  
subject_exam_scores = melted_df.groupby("subject")[["exam_score"]].mean().round(1)  
subject_exam_scores
```

exam_score	
subject	
Biology	81.6
Composition	82.5
Math	82.0
Speech	86.0





## Activity:

### Resampling and Melting DataFrames

---

In this activity, you will use the `resample` function to down-sample time series data and use the `melt` function to reshape a DataFrame from a wider form to longer form.

**Suggested Time:**

15 Minutes



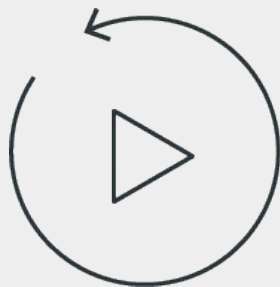


**Time's up!**  
Let's review



**Questions?**





Let's **recap**



# Recap

After today's lesson you are able to:

---

- 1 Understand the concept of pivoting data and explain its role in data analysis.
- 2 Differentiate between single and multiple aggregations when reshaping data.
- 3 Apply one or more aggregation functions to reshaped data.
- 4 Use the **aggfunc** function effectively to perform various aggregations on reshaped data.
- 5 Use custom Python functions to transform reshaped data.
- 6 Reshape data into multi-index and apply aggregations.
- 7 Reshape data using `resample` and `melt`.



# Challenge

You'll be analyzing electronics sales data to gain insights to which zip codes have the most products ordered, generated the most sales, and had the greatest average sales price. Then, you'll determine which zip codes had the most iPhone sales, and which day and week had the most iPhone sales.



**Questions?**





**The End**