# Databricks

Data Boot Camp

Lesson 22.4

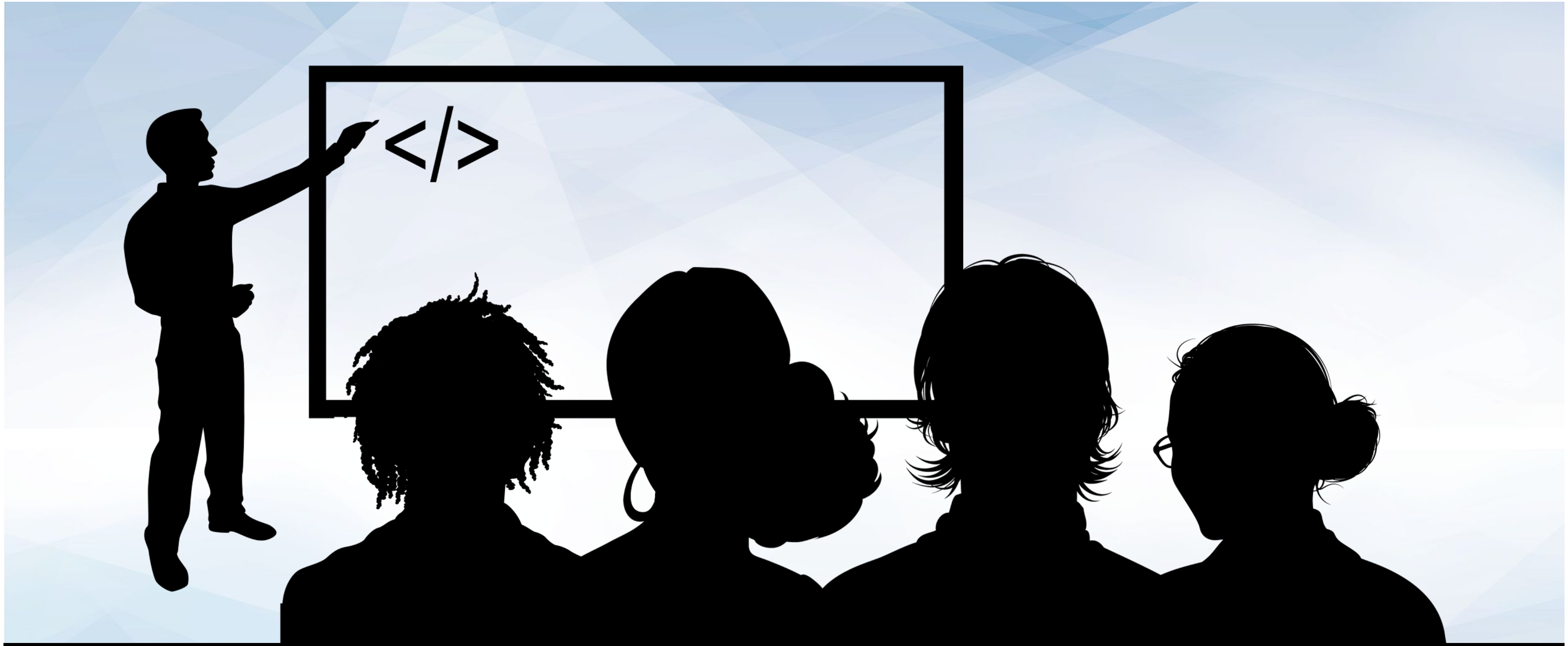
# Congratulations!

- You’ve (almost) made it to the end of the course.
- You have learned many technologies used in data analytics, like SQL and Python/Pandas.
- You have acquired quantitative skills, including statistical analysis.
- You have learned the crucial skills of data visualization and data storytelling.
- The final project is an opportunity to integrate and showcase all of these skills.

# Class Objectives

By the end of this lesson, you will be able to:

- Describe the purpose of Databricks, and identify two of its key features and a use case.
- Set up a Databricks environment and identify its key components.
- Navigate the Databricks workspace using dbutils.
- Import data into a new notebook using the following sources: Parquet, CSV, and S3.
- Explain the advantage of Parquet as a big-data storage format.
- Perform complex data analysis using Python and SQL interfaces.
- Identify two advantages of using Databricks over PySpark in data analysis.



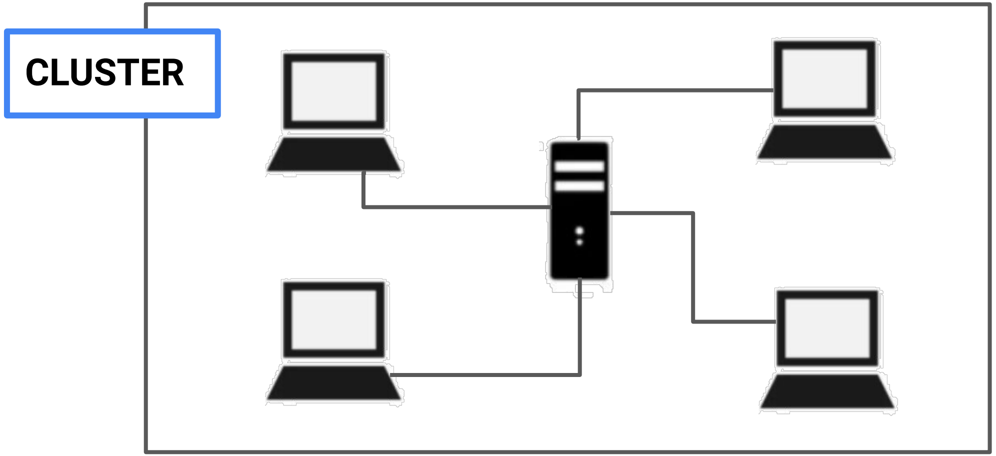
# Instructor Demonstration

## Introduction to Databricks

What is Databricks?

# Introduction to Databricks

- A **cloud** platform for running Apache Spark for big data analysis.
- Provides a robust system to manage and optimize clusters of computers for data analysis.
- Remember, a cluster is a network of machines that coordinate and divide up data-related tasks.



databricks

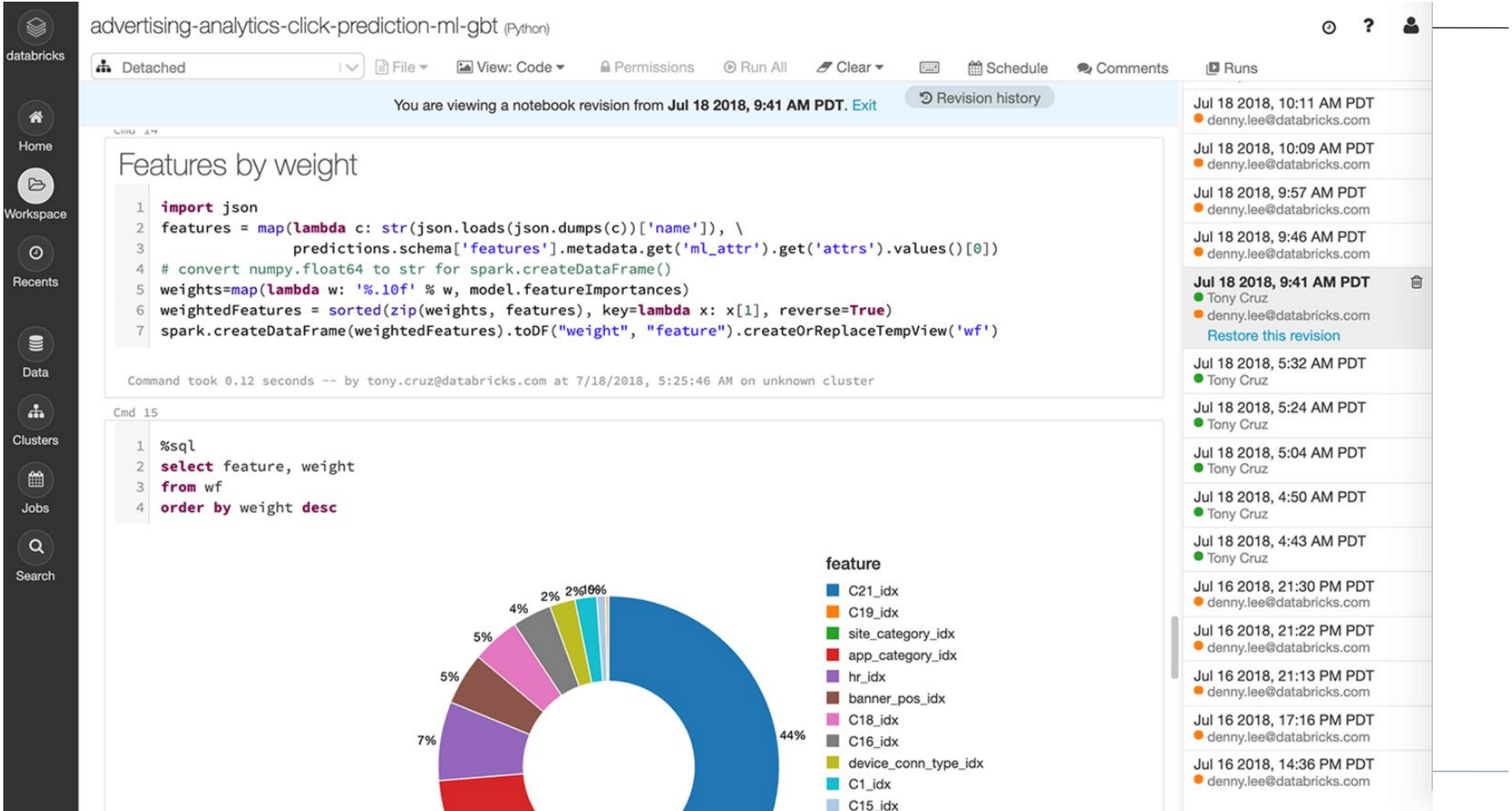
## Databricks Advantages

# Introduction to Databricks

- **Ease of use.** Databricks can scale activity depending on need.
- **Real-time collaboration.** Multiple people can work on the same notebook simultaneously.
- **Potential savings in time and cost.** Pay for what you use. Eliminates need for a separate administrator.
- **Flexible use of multiple languages:** Python, SQL, R, Scala.

Large-scale data analytics is moving toward cloud platforms like Databricks.

# Example of shared Databricks notebook



advertising-analytics-click-prediction-ml-gbt (Python)

You are viewing a notebook revision from **Jul 18 2018, 9:41 AM PDT**. Exit

Cmd 14

## Features by weight

```
import json
features = map(lambda c: str(json.loads(json.dumps(c))['name']), \
               predictions.schema['features'].metadata.get('ml_attr').get('attrs').values()[0])
# convert numpy.float64 to str for spark.createDataFrame()
weights=map(lambda w: '%.10f' % w, model.featureImportances)
weightedFeatures = sorted(zip(weights, features), key=lambda x: x[1], reverse=True)
spark.createDataFrame(weightedFeatures).toDF("weight", "feature").createOrReplaceTempView('wf')
```

```
Command took 0.12 seconds -- by tony.cruz@databricks.com at 7/18/2018, 5:25:46 AM on unknown cluster
```

Cmd 15

```
%sql
select feature, weight
from wf
order by weight desc
```

Jul 18 2018, 10:11 AM PDT
denny.lee@databricks.com

Jul 18 2018, 10:09 AM PDT
denny.lee@databricks.com

Jul 18 2018, 9:57 AM PDT
denny.lee@databricks.com

Jul 18 2018, 9:46 AM PDT
denny.lee@databricks.com

**Jul 18 2018, 9:41 AM PDT**
Tony Cruz
denny.lee@databricks.com

Jul 18 2018, 5:32 AM PDT
Tony Cruz

Jul 18 2018, 5:24 AM PDT
Tony Cruz

Jul 18 2018, 5:04 AM PDT
Tony Cruz

Jul 18 2018, 4:50 AM PDT
Tony Cruz

Jul 18 2018, 4:43 AM PDT
Tony Cruz

Jul 16 2018, 21:30 PM PDT
denny.lee@databricks.com

Jul 16 2018, 21:22 PM PDT
denny.lee@databricks.com

Jul 16 2018, 21:13 PM PDT
denny.lee@databricks.com

Jul 16 2018, 17:16 PM PDT
denny.lee@databricks.com

Jul 16 2018, 14:36 PM PDT
denny.lee@databricks.com

### Parquet

# Introduction to Databricks

- Like CSV or JSON, Parquet is a data storage format.
- Parquet is commonly used with Spark.
- Unlike CSVs, where rows are read into a Pandas DataFrame, Parquet allows selective loading of columns.
- **Question: What’s a potential advantage of using Parquet?**

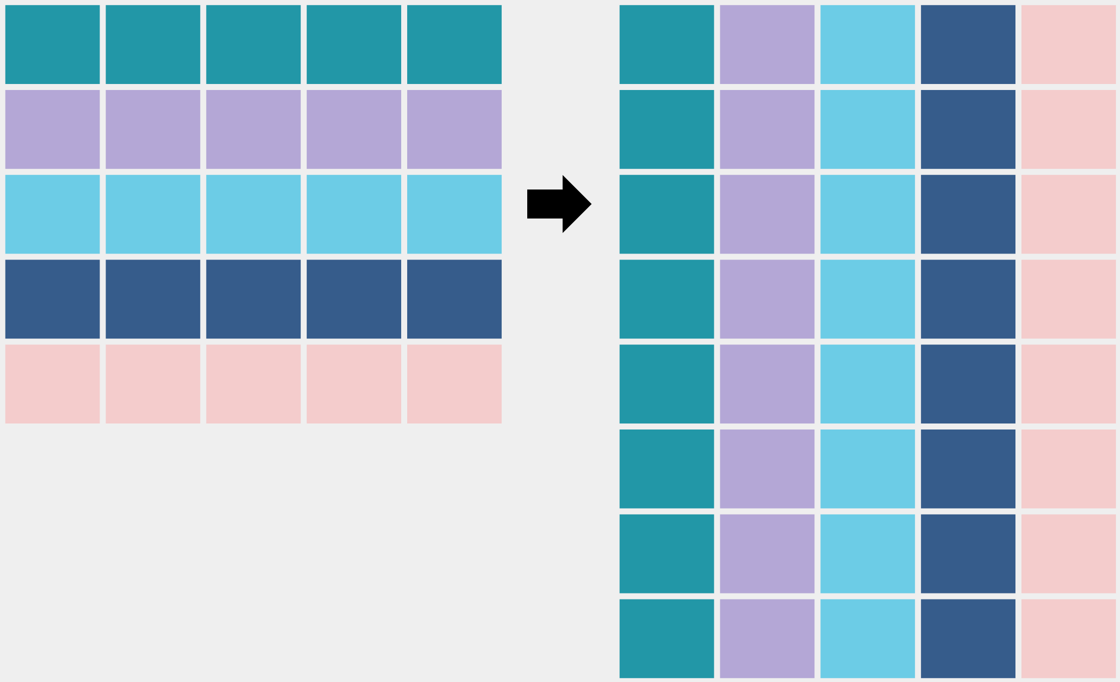
## Parquet

# Introduction to Databricks

- Traditional data formats store data by row. Therefore, if we use multiple nodes to perform Spark queries, each node would need to load a copy of all rows in a dataset. Making complete copies of our data is both slow and storage-intensive.
- Instead of storing data row-by-row, Spark can use Parquet format, which stores data in columnar format.
- Parquet allows us to store and retrieve only selected columns in the data. Loading only the specified columns can lead to savings in time and computing resources.

# Optimizing Spark — Data Storage

“Columnar” refers to how the data is stored.



- In a columnar format, each column of a row is stored separately, with a reference to all of its columns.
- This allows you to query and filter a single column and return only the selected columns in your query with great efficiency.
- This also greatly reduces the amount of reading Spark needs to do.

# Activity: Sign Up for Databricks

In this activity, you will sign up for a free Databricks Community Edition account.

**Suggested Time:**
10 Minutes

Instructions:

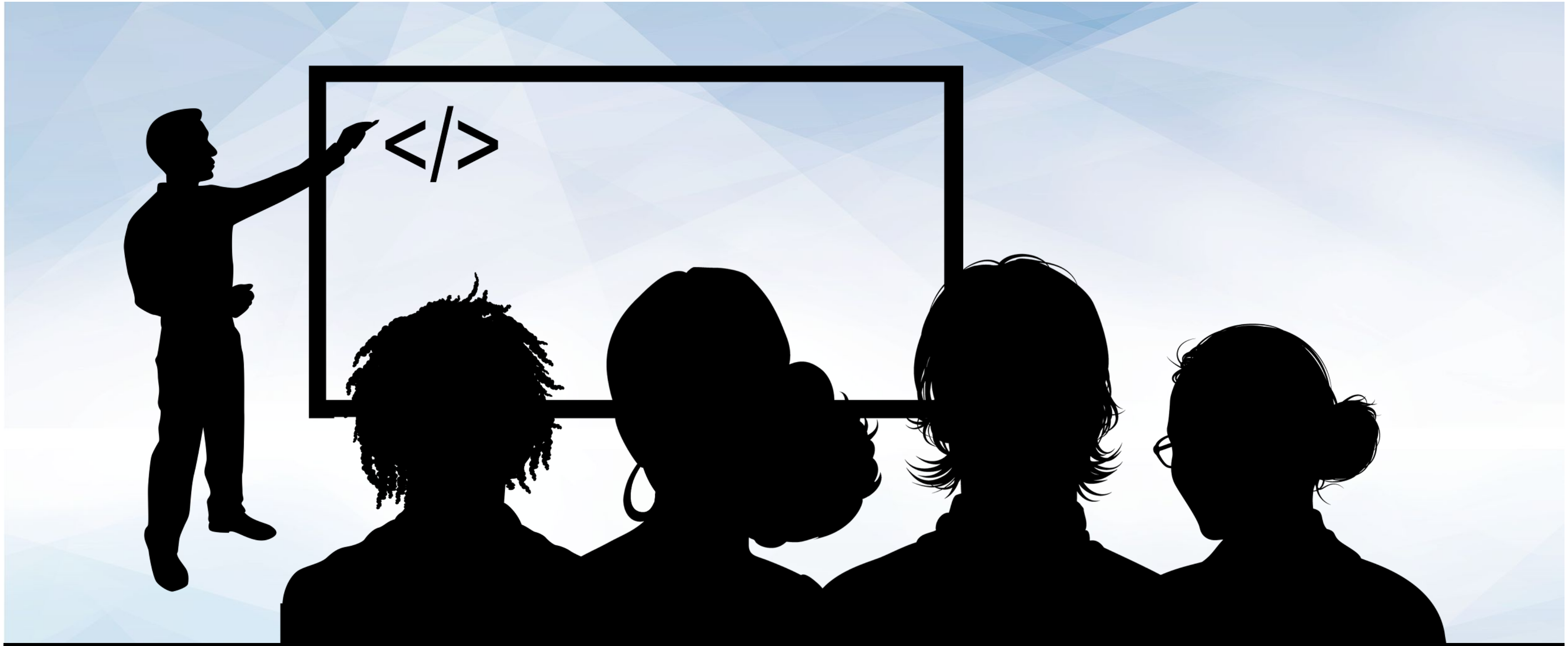
# Activity: Sign Up for Databricks

- Sign up for a Databricks Community Edition account.

- If you finish early, continue to explore the Databricks interface.

- **Bonus**:
  - Upload the included data files to Databricks. Create a Spark DataFrame for each.

# Let's Review



# Instructor Demonstration

## Databricks Demo

# Activity: Databricks Basics

In this activity, you will create a Databricks notebook and perform basic data analysis using Python and SQL interfaces.

**Suggested Time:**
15 Minutes

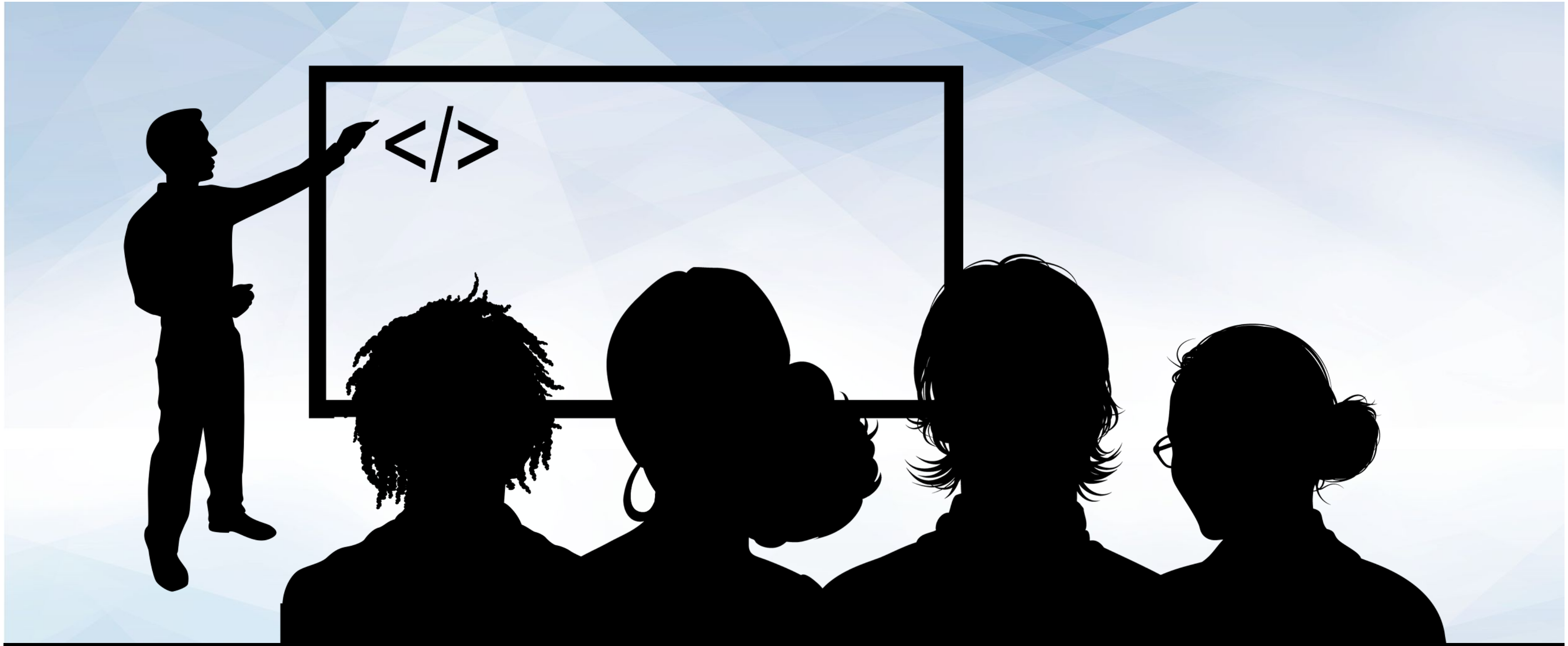
Instructions:

# Activity: Databricks Basics

- Upload **vehicles.csv** to Databricks.
- Create a blank Databricks notebook and use **dbutils** to note the location of the CSV.
- Create a Spark DataFrame of the dataset and preview the DataFrame.
- Create a PySpark query to obtain the number of vehicles for sale by type of transmission.
- Create a bar chart of the results.
- Perform the same query and visualization using SQL. You will need to create a temporary table in order to do this.

- **Bonus**:
  - The same dataset is available in Parquet format.
  - Load only the following columns of the dataset into a Spark DataFrame: year, manufacturer, and transmission.
  - Using PySpark, obtain the number of vehicles per sale by manufacturer.

# Let's Review



# Instructor Demonstration

## Joins

# Activity: Joins in Databricks

In this activity, you will perform joins on datasets, using both PySpark and SQL interfaces in Databricks.

**Suggested Time:**
15 Minutes

Instructions:

# Activity: Joins in Databricks

- Use the S3 links to create a Spark DataFrame for each.
- Use **display** to preview the DataFrames. Count the number of rows in each DataFrame.
- Join the two DataFrames in order to answer the following questions using PySpark: How many birds are there in the dataset? How many rodents were recorded in 1978?
- Create a temporary table of each DataFrame. Preview the first 5 rows and perform the same queries above, this time in SQL.

# Let’s Review

# Activity: Group Activity

In this activity, you will work in groups to perform data analysis using a database of a fictional company. You'll put together all the skills you learned today and in this course, such as loading multiple data sources, analyzing data, visualizing data, and presenting findings.

**Suggested Time:**
60 Minutes

## Instructions:

# Activity: Group Activity

- For each data file, create a Spark DataFrame and a temporary view.
- Run **spark.catalog.listTables()** to verify that the tables have been created.
- Create requested queries using SQL. Feel free to create additional queries of your own.
- Use the results of your data analysis to create a brief report (about 3 to 5 slides).
    - Make three actionable recommendations. Support each recommendation with a data finding.
    - Use visualizations where appropriate.
- Send the link of your presentation slides to your instructor.

# Activity: Group Presentations

In this activity, you will present your results.

**Suggested Time:**
20 Minutes

# Let's Review