

# Develop Like A Pro In Databricks Notebooks

---

Weston Hutchins – Product Manager, Databricks

Neha Sharma – Engineering Manager, Databricks



# Product safe harbor statement

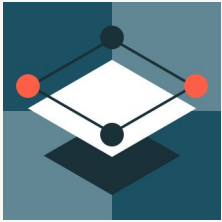
This information is provided to outline Databricks' general product direction and is **for informational purposes only**. Customers who purchase Databricks services should make their purchase decisions relying solely upon services, features, and functions that are currently available. Unreleased features or functionality described in forward-looking statements are subject to change at Databricks discretion and may not be delivered as planned or at all.



**Why is becoming data-driven  
such a challenge for companies?**



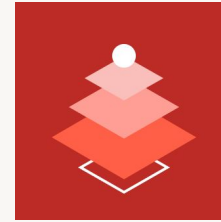
# There are major hurdles in the way..



## Data is spread out across many sources

Making data easily available is complex

- Can have many data types: tabular data, text blobs, images, streaming logs, etc.
- Can be stored across clouds
- Each source has different semantics, access controls, and governance patterns



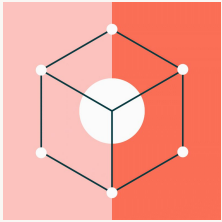
## Disparate tools limit efficiency, reproducibility, and scale

Data is just the starting point; getting value out is its own challenge

- Users can be siloed, have different skill sets, and use different tools
- Results aren't reproducible and don't reach the right audience
- Getting to production takes forever



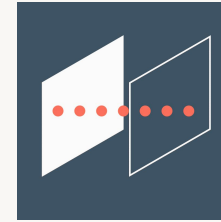
# But—hurdles are meant to be jumped!



## The Databricks Lakehouse unifies all data in one place, regardless of type or source

Companies can leave data where it is and still..

- Make existing data lakes and warehouses accessible, governable, and secure
- Leverage an open ecosystem
- Unify all use cases (DS, ML, BI, etc.) on top



## The Databricks Lakehouse developer experience accelerates the journey to insights

Users get a powerful developer platform in which to derive insights and deliver value

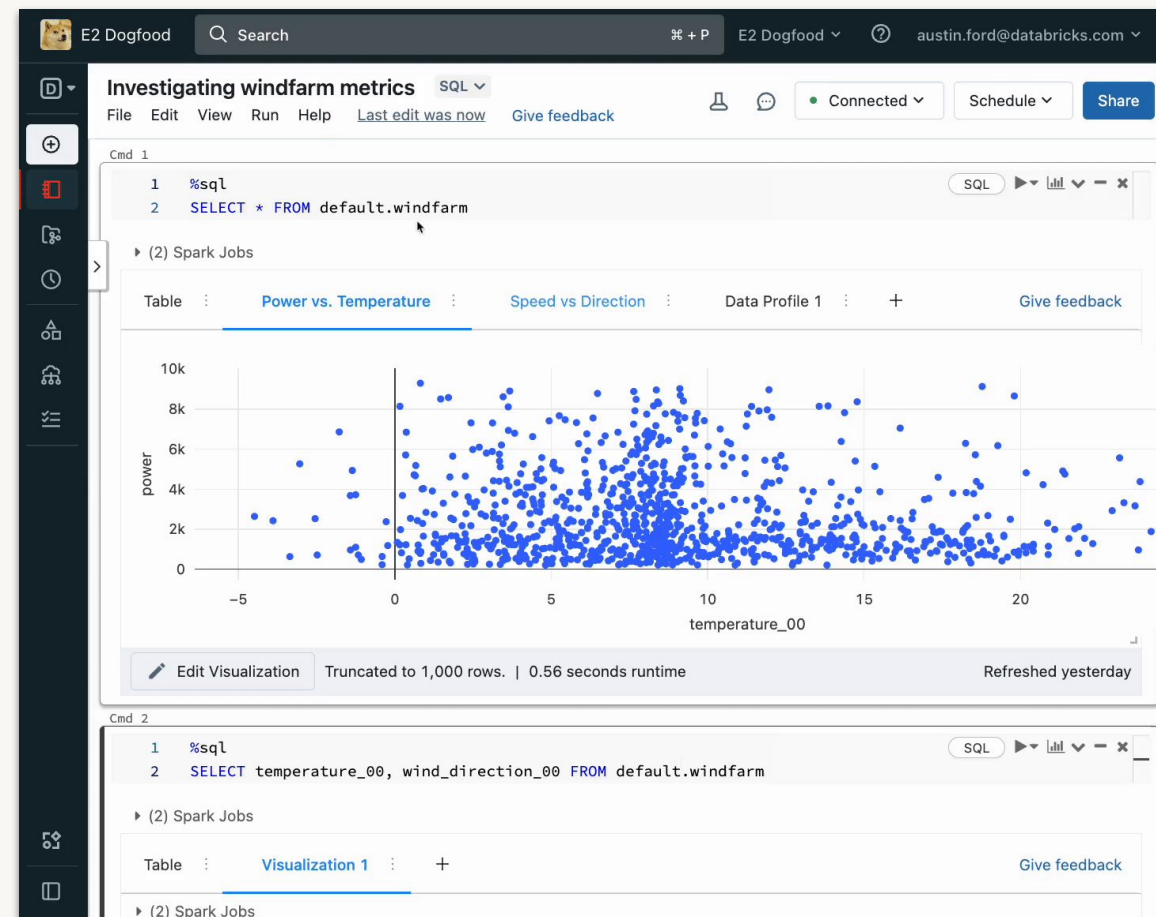
- A collaborative **Notebook** that enables efficient data analysis, insights sharing, and faster paths to production
- Support for all popular developer tools



# Developing in the Lakehouse

## Bringing the Lakehouse platform to users' fingertips

- But—*what is it and how do I use it?*
- The developer experience is made up of all the authoring surfaces and tools:
  - The **Workspace**, the **Notebook**, the **SQL Editor**, **Repos**, the **CLI** and **external APIs**.
- They are the face of Databricks and the **front door** to the Lakehouse



Let's talk about the **Notebook**



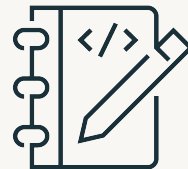
# Our **vision** for the Notebook

The best authoring experience for the lakehouse

Lakehouse-aware through  
LakehouseIQ



Bringing your favorite IDE  
features to the notebook



Modern, intuitive UX





# The Databricks Notebook

## The front door of the Lakehouse

### Multi-language

Use Python, SQL, Scala, and R, all in one Notebook

### Collaborative

Real-time co-presence, co-editing, and commenting

### Jupyter-compatible

Use the power of the Jupyter ecosystem in the Notebook

### Ideal for exploration

Explore, visualize, and summarize data with built-in charts and data profiles



### Adaptable

Install standard libraries and use local modules

### Reproducible

Automatically track version history, and use git version control with Repos

### Get to production faster

Quickly schedule notebooks as jobs or create dashboards from their results, all in the Notebook

### Enterprise-ready

Enterprise-grade access controls, identity management, and auditability





# Databricks Assistant

*Powered by LakehouseIQ* 

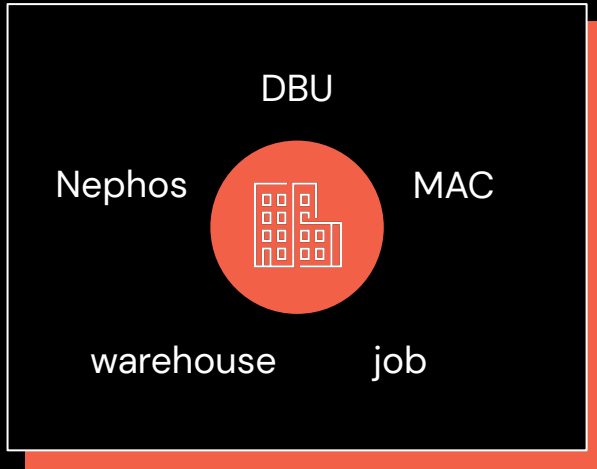


# Problem:

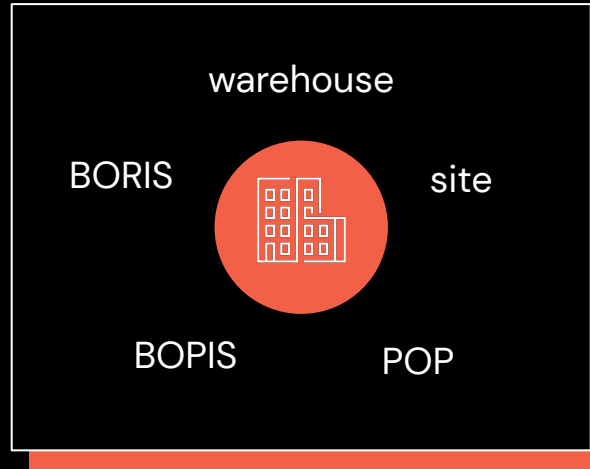
Naively adding an LLM assistant doesn't work

# Problem:

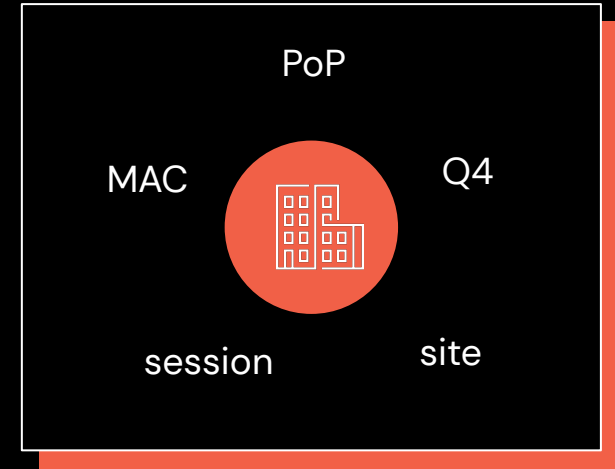
every organization has its own jargon, data, and org structure



Software Company



Retailer

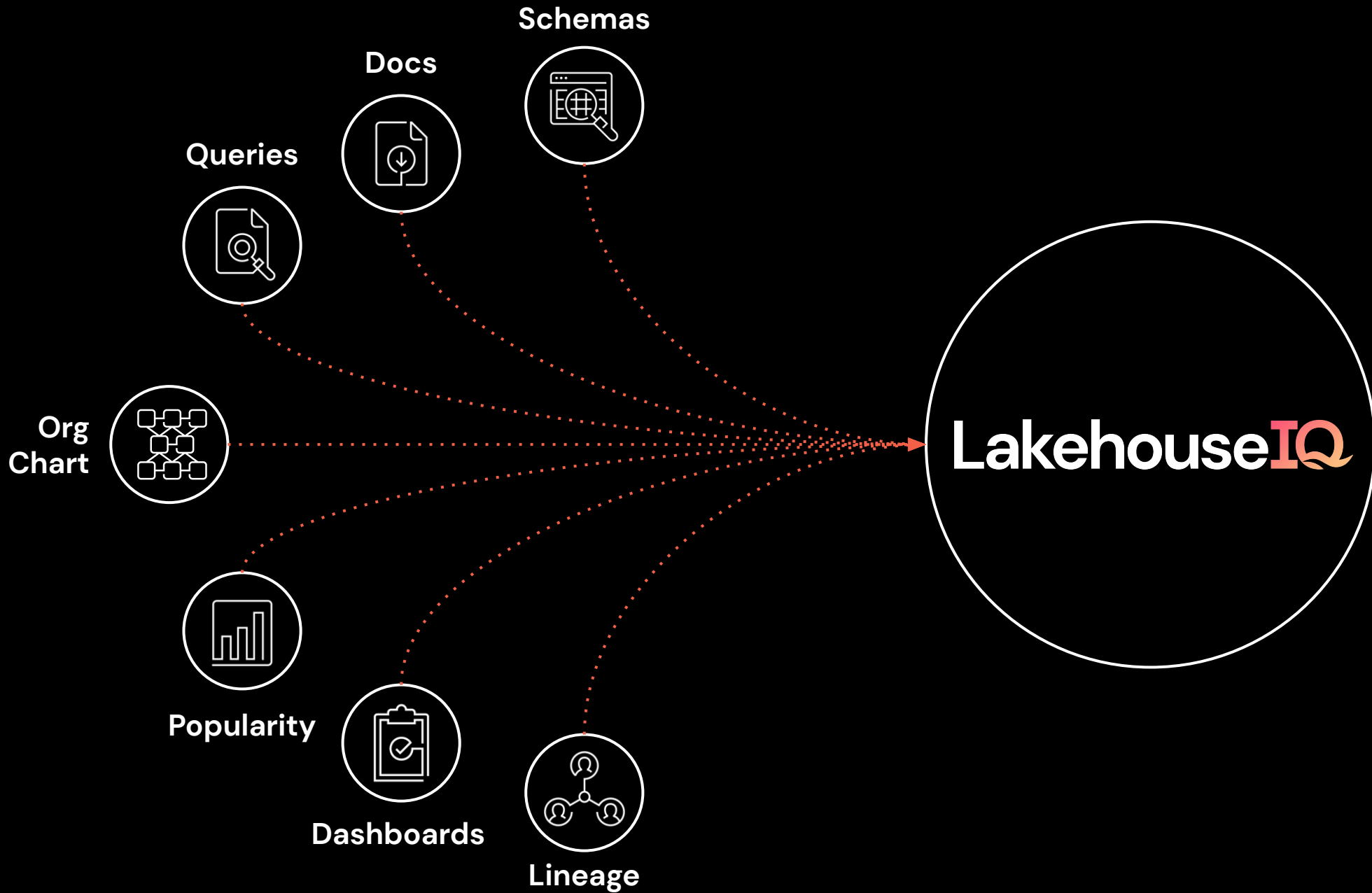


Telco

*“How many **DBUs** were there in Europe last quarter?”*

**DBU** may refer to:

- [1,8-Diazabicyclo\[5.4.0\]undec-7-ene](#), an organic chemical, amidine base
- [Desh Bhagat University](#), Mandi Gobindgarh, Punjab, India
- [Dallas Baptist University](#), a Christian liberal arts university in Dallas, Texas
- [Dansk Boldspil-Union](#), the [Danish Football Association](#)
- dBu, abbreviation of [decibel  \$\mu\$ V](#), a logarithmic measurement of voltage
- DBU, ISO 639-3 code for the [Bondum Dogon](#) language
- [Duluth Business University](#), a private university in Duluth, Minnesota



# Authoring Assistant

- LLM-powered Authoring Assistant integrated into the notebook, file editor, and SQL editor.
- Uses **context** to return more accurate responses
  - Code cells
  - Libraries
  - Databricks Runtime version
  - Documentation
  - Table schema
  - Recent and favorite tables
  - Popular tables
  - Popular joins
  - UC Descriptions and Tags
  - Lineage

The screenshot displays the 'Assistant playground' interface in Databricks. The top bar includes 'Assistant playground', 'Python' language selection, and buttons for 'Run all', 'Schedule', and 'Share'. Below the top bar, the 'Assistant' panel shows the context 'Command 1' and the user's query: 'Using df, plot the number of trips per day.' The assistant's response provides a list of steps: 1. Convert the pickup date-time column to a date-only column using the pandas dt accessor. 2. Group the data by date and count the number of trips in each group. 3. Plot the resulting counts using any plotting library of your choice. Below the steps, the assistant provides an example implementation using pandas and matplotlib, with the following code:

```
python
import matplotlib.pyplot as plt

# Convert datetime column to date-only column
df['date'] = df['pickup_datetime'].dt.date

# Group trips by date and count number of trips in each group
trips_per_day = df.groupby('date').size()

# Plot trips per day
trips_per_day.plot()

# Set plot title and axis labels
```

The right panel shows the execution of the code, displaying the command 'df = spark.sql('select \* from piatek.nyctaxi limit 10000').toPandas()' and the resulting Spark Jobs output. The output indicates that the command took 2.42 seconds and was executed on 4/22/2023, 12:52:31 PM on michael.piatek@databricks.com's Personal Compute (Fleet) Cluster. Below the output, there are instructions: 'Shift+Enter to run' and 'Shift+Ctrl+Enter to run selected text'.



# Text-to-SQL

- **Personalized for you:** Use natural language to create and refine queries using company jargon
- **Inline-suggestions:** Write a comment and have the Assistant suggest a commonly used query snippet.
- **Find relevant tables:** Enhanced search makes it easy to discover data assets.

The screenshot displays the Databricks web interface. At the top, there's a search bar and user information. The main area is divided into three sections:

- Query Editor:** Contains a SQL query to calculate weekly revenue for European sales territories. The query is as follows:

```
1 SELECT
2   date_trunc('week', date) as week,
3   salesTerritory,
4   sum(sales_amount) as revenue
5 FROM
6   dbdemos.sales.orders_gold
7 WHERE
8   -- European sales territories
9   salesTerritory in (
10    'EMEA Central',
11    'EMEA Northern',
12    'EMEA MEA',
13    'EMEA Southern'
```
- Results Table:** Shows the output of the query with 84 rows. The first six rows are:

#	week	salesTerritory	revenue
1	2023-06-12 00:00:00.000	EMEA Northern	71578.45
2	2023-06-12 00:00:00.000	EMEA Central	18456.53
3	2023-06-12 00:00:00.000	EMEA MEA	11770.41
4	2023-06-05 00:00:00.000	EMEA MEA	13018.25
5	2023-06-05 00:00:00.000	EMEA Northern	70453.75
6	2023-06-05 00:00:00.000	EMEA Central	11189.24
- Assistant Chat:** A chat window on the right titled "Assistant" shows a conversation. The user asks: "Calculate the weekly revenue in Europe". The Assistant responds: "Here is a SQL query for the weekly revenue in European sales territories:" and provides a SQL query snippet that is identical to the one in the editor.



# Complete

Databricks Assistant Demo

Python

File Edit View Run Help [Last edit was 2 minutes ago](#) [Provide feedback](#)

▶ Run all

● Weston Hutchins's Int... ▼

📅 Schedule

Share



Cmd 1

Python



```
1 #Write code to reverse a string. Just the code; no explanatory text  
2
```

Shift+Enter to run

Shift+Ctrl+Enter to run selected text



# Generate from Text

The screenshot shows the Databricks Assistant playground interface. At the top, it says "Assistant playground" with a "Python" dropdown menu. Below this are menu items: "File", "Edit", "View", "Run", "Help", and a link "Last edit was 1 minute ago". There are also buttons for "Run all", a user dropdown "michael.piatek@data...", "Schedule", and "Share".

The interface is split into two main panels. The left panel is titled "Assistant" and contains a "Context: Command 1" section. Below that, the "User:" section contains the text: "Using df, plot the number of trips per day." The "Assistant:" section provides instructions: "To plot the number of trips per day, you could:" followed by a numbered list: 1. Convert the pickup date-time column to a date-only column using the pandas dt accessor. 2. Group the data by date and count the number of trips in each group. 3. Plot the resulting counts using any plotting library of your choice. Below the list, it says "Here's an example implementation using pandas and matplotlib:" and shows a code block with the following Python code:

```
python
import matplotlib.pyplot as plt

# Convert datetime column to date-only column
df['date'] = df['pickup_datetime'].dt.date

# Group trips by date and count number of trips in each group
trips_per_day = df.groupby('date').size()

# Plot trips per day
trips_per_day.plot()

# Set plot title and axis labels
```

At the bottom of the code block is an input field with the placeholder text "Enter your message" and a send button.

The right panel is titled "Cmd 1" and shows the execution of the code. It displays the code: `df = spark.sql('select * from piatek.nyctaxi limit 10000').toPandas()`. Below the code, it shows "(1) Spark Jobs" and a status message: "Command took 2.42 seconds -- by michael.piatek@databricks.com at 4/22/2023, 12:52:31 PM on michael.piatek@databricks.com's Personal Compute (Fleet) Cluster". At the bottom of the panel, there are instructions: "Shift+Enter to run" and "Shift+Ctrl+Enter to run selected text".

# Explain

**Demo Draft: Notebooks** Python

File Edit View Run Help Last edit was 5 minutes ago Provide feedback Run all Weston Hutchins's Int... Schedule Share

**Assistant**

User  
Explain this code

Assistant

This code reads a dataframe, filters it for movies and TV shows and plots a bar chart to show the count of movies versus TV shows.

- `import matplotlib.pyplot as plt` is importing the matplotlib library for plotting.
- `titles_df` is a pandas dataframe object containing data about movie and TV show titles.
- `movies_df` is another dataframe that is created by filtering the `titles_df` dataframe to only include rows where type is "MOVIE".
- `shows_df` is created similarly, for rows where type is "SHOW"
- `movies_count` and `shows_count` are each calculated by taking the length of each dataframe object created in the previous step.
- `fig, ax = plt.subplots(figsize=(6, 4))` is creating a figure with a size of 6 x 4 inches.
- `ax.bar(["Movies", "TV Shows"], [movies_count, shows_count], color=["red", "blue"])` is creating a bar chart with two bars, one for

Cmd 4

## Movies vs Shows

Cmd 5

```
1 import matplotlib.pyplot as plt
2
3 movies_df = titles_df[titles_df["type"] == "MOVIE"]
4 shows_df = titles_df[titles_df["type"] == "SHOW"]
5 movies_count = len(movies_df)
6 shows_count = len(shows_df)
7 fig, ax = plt.subplots(figsize=(6, 4))
8 ax.bar(["Movies", "TV Shows"], [movies_count,
9 shows_count], color=["red", "blue"])
9 ax.set_title("Total number of Movies and TV Shows")
10 ax.set_xlabel("Type")
11 ax.set_ylabel("Count")
12 plt.show()
```

The bar chart displays the total number of movies and TV shows. The x-axis is labeled 'Type' and the y-axis is labeled 'Count'. There are two bars: a red bar for 'Movies' and a blue bar for 'TV Shows'. The red bar is significantly taller than the blue bar.

Type	Count
Movies	~3500
TV Shows	~1500



# Fix

### Databricks Assistant Demo

Python  
File Edit View Run Help [Last edit was 1 minute ago](#) [Provide feedback](#) ▶ Run all ● Weston Hutchins's Int... 📅 Schedule Share

#### Assistant

**User**  
How do I fix my code?

**Assistant**  
The zeros function takes the shape of the array as a single argument which should be a tuple. In your code, you are providing two integers as separate arguments instead of a single tuple.  
Here is the corrected code:

```
python  
  
import numpy as np  
np.zeros((2, 3))
```

Note that the argument passed to zeros is now a tuple (2, 3).

#### Cmd 2

Python

```
1 import numpy as np  
2 np.zeros(2,3)
```

**TypeError: Cannot interpret '3' as a data type**

-----  
-----  
TypeError Traceback (most recent c  
all last)  
File <command-4252588689653531>:2  
 1 import numpy as np  
----> 2 np.zeros(2,3)

**TypeError: Cannot interpret '3' as a data type**

💡 1

Command took 1.01 seconds -- by weston.hutchins@databricks.com at 6/28/2023, 3:36:16 PM on Weston Hutchins's Interactive Cluster Policy Cluster

Shift+Enter to run  
Shift+Ctrl+Enter to run selected text

# Transform/Optimize

**Demo Draft: Notebooks** Python ▼

File Edit View Run Help [Last edit was 8 minutes ago](#) [Provide feedback](#) ▶ Run all ● Weston Hutchins's Int... ▼ 📅 Schedule Share ^

**Assistant** ⓧ ⓘ ✕

**User**

Convert this code to Pyspark

**Assistant**

```
python
```

```
# Load netflix_titles into a pandas df
titles_df = spark.sql("select * from users.neha_sharma.netf
display(titles_df)
```

The equivalent PySpark code for this would be:

```
python
```

```
# Load netflix_titles into a PySpark df
titles_df = spark.sql("select * from users.neha_sharma.netf
display(titles_df)
```

Note that the `toPandas()` method has been removed because it converts the Spark DataFrame to a pandas DataFrame, which would be inappropriate if you want to work with Spark DataFrames. If you need to work with pandas DataFrames, you could use the `toPandas()` method

Cmd 2

## Load into Pandas

Cmd 3

```
Python ▶ 📊 ▼ - ✕
```

```
1 # Load netflix_titles into a pandas df
2 titles_df = spark.sql("select * from users.
   neha_sharma.netflix_titles").toPandas()
3
4 display(titles_df)
```

Table ▼ +

	id	title
1	ts300399	Five Came Back: The Reference Films
2	tm84618	Taxi Driver
3	tm154986	Deliverance
4	tm127384	Monty Python and the Holy Grail



# Other LakehouseIQ-Powered Features

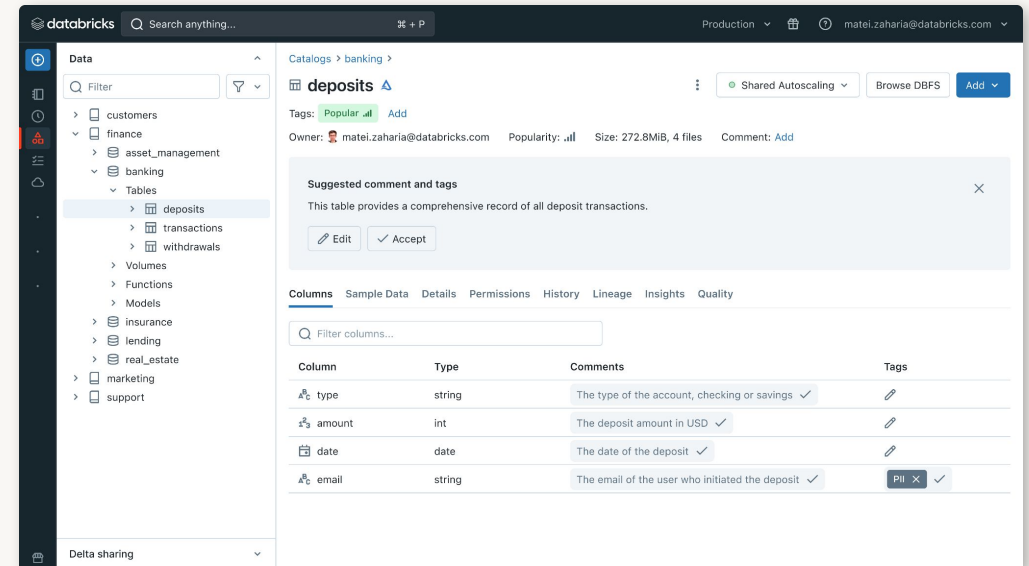
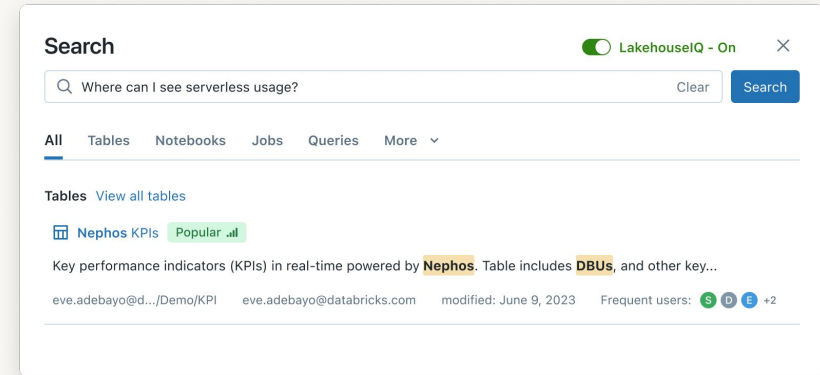
Search

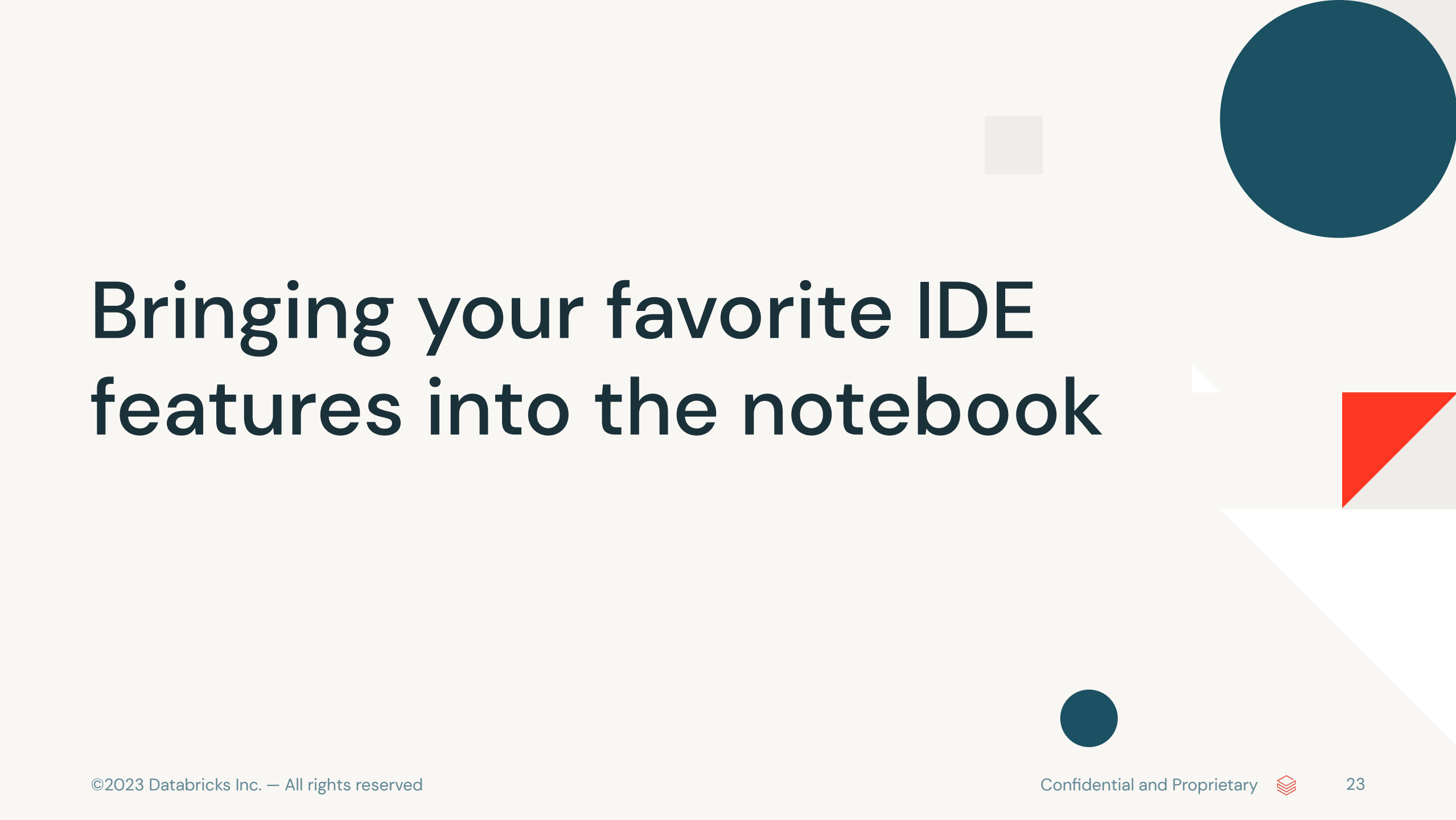
Metadata suggestions

Workflow troubleshooting

Spark NL API

docs.databricks.com





# Bringing your favorite IDE features into the notebook



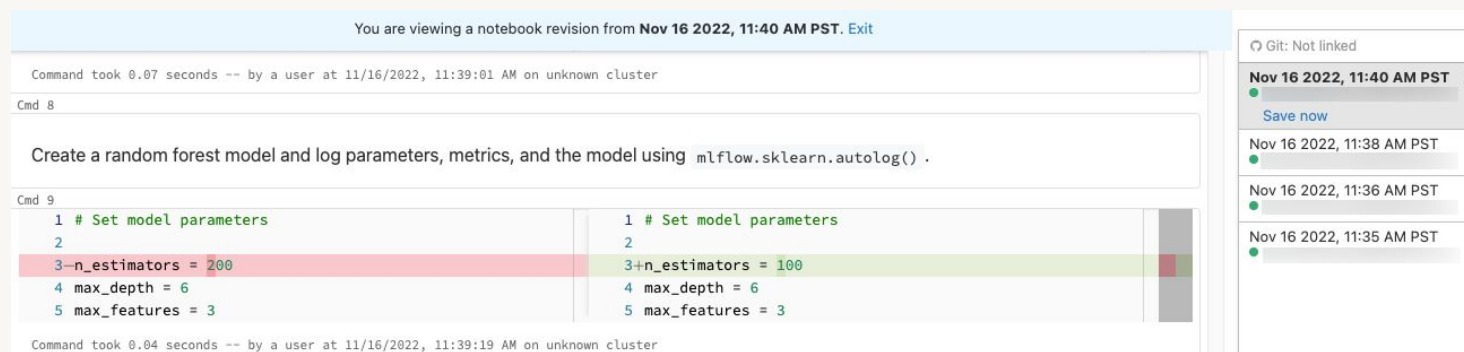
# Unified Editor Everywhere

## Consistent editing experience across notebooks, files, and SQL

- New Editor: the same editor that powers VS Code.
- Integrated into the Notebook, File Editor, SQL Editor
- What you now get...
  - Auto-complete as-you-type
  - Code folding
  - Multi-cursor and box selection
  - Side-by-side diff
  - Run selected text



A screenshot of a code editor window. The title bar shows 'Python' and standard window controls. The code area contains three lines: '1 import numpy as np', '2', and '3 |'. Below the code area, it says 'Shift+Enter to run'.



A screenshot of a notebook revision interface. The top bar says 'You are viewing a notebook revision from Nov 16 2022, 11:40 AM PST. Exit'. Below that, it shows 'Command took 0.07 seconds -- by a user at 11/16/2022, 11:39:01 AM on unknown cluster'. The main area shows 'Cmd 8' with the command 'Create a random forest model and log parameters, metrics, and the model using mlflow.sklearn.autolog()'. Below that, 'Cmd 9' shows a side-by-side diff of code. The left side has '1 # Set model parameters', '2', '3-n\_estimators = 200', '4 max\_depth = 6', and '5 max\_features = 3'. The right side has '1 # Set model parameters', '2', '3+n\_estimators = 100', '4 max\_depth = 6', and '5 max\_features = 3'. The bottom bar says 'Command took 0.04 seconds -- by a user at 11/16/2022, 11:39:19 AM on unknown cluster'. On the right, there's a sidebar with 'Git: Not linked' and a list of revisions: 'Nov 16 2022, 11:40 AM PST' (selected), 'Nov 16 2022, 11:38 AM PST', 'Nov 16 2022, 11:36 AM PST', and 'Nov 16 2022, 11:35 AM PST'.

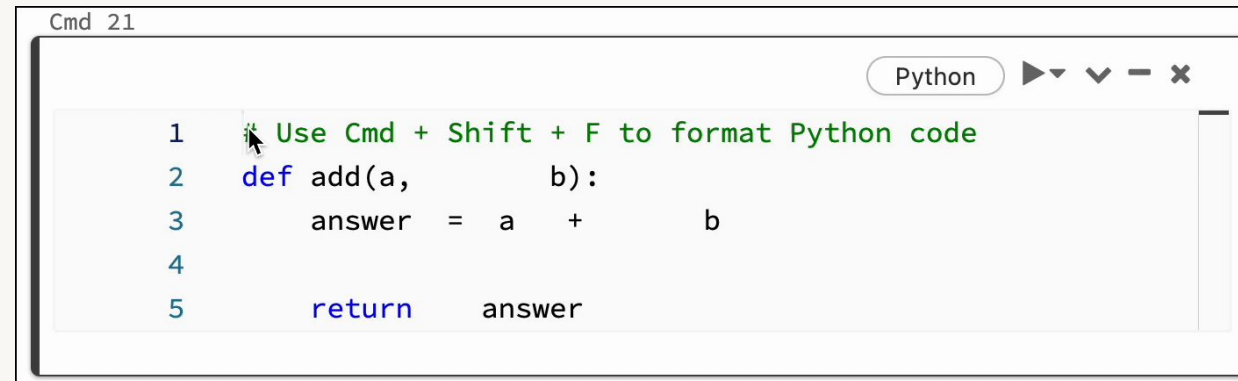




# Language Features

## Language-specific smarts

- We've built a **Python Language Server** that interacts with our Notebook.
- Adding **language features** such as go-to-definition, go-to error, syntax highlighting, inline errors, quick fixes, auto-import libraries and more.
- **Format** your Python code using Black.



The screenshot shows a code editor window titled "Cmd 21" with a "Python" language selector. A tooltip is displayed over the code, stating "Use Cmd + Shift + F to format Python code". The code in the editor is:

```
1  Use Cmd + Shift + F to format Python code
2  def add(a,      b):
3      answer = a  +      b
4
5      return  answer
```

# Debugging Features

Make it easy to find and fix bugs

- The new **Variable Explorer** makes it easy to view variable types and values.
- **Inspect a DataFrame** in your notebook with a single click.
- Set breakpoints and **step-through** your code with pdb.

The screenshot displays a Databricks notebook interface. On the left, a code editor window titled 'Cmd 1' shows Python code for creating and manipulating DataFrames and arrays. Below the code, the execution output shows the creation of a Spark DataFrame from a pandas DataFrame. On the right, the 'Variables 4' panel is open, displaying a list of variables and their types and values. The variables listed are my\_koalas\_df (DataFrame), my\_np\_arr (ndarray), my\_pandas\_df (DataFrame), and my\_spark\_df (DataFrame). The my\_np\_arr variable is expanded to show its values: [[0.65029338, 0.99607378], [0.89789179, 0.18326482], [0.95789813, 0.92213363]].

```
Cmd 1
Python ▶ ▾ - ×
1 import numpy as np
2 import pandas as pd
3 import random
4 import pyspark.pandas as ps
5
6 my_np_arr = np.random.rand(3, 2)
7 my_pandas_df = pd.DataFrame({"Column1": my_np_arr[:, 0], "Column2": my_np_arr[:, 1]})
8 my_spark_df = spark.createDataFrame(my_pandas_df)
9
10 my_koalas_df = ps.from_pandas(my_pandas_df)

▶ my_spark_df: pyspark.sql.dataframe.DataFrame = [Column1: double, Column2: double]
Command took 0.33 seconds -- by jaipreet.singh@databricks.com at 5/3/2023, 5:43:52 PM on 13.x Shared Autoscaling

Shift+Enter to run
Shift+Ctrl+Enter to run selected text
```

Variables 4 Give feedback ⓘ ×

Filter by name or type

my_koalas_df	DataFrame (? , 2)
['Column1', 'Column2']	
my_np_arr	ndarray (3, 2)
[[0.65029338, 0.99607378], [0.89789179, 0.18326482], [0.95789813, 0.92213363]]	
my_pandas_df	DataFrame (3, 2)
['Column1', 'Column2']	
my_spark_df	DataFrame (? , 2)
['Column1', 'Column2']	



# Repos

## Using git for production

- Sparse checkout for large monorepos
- Reset workflow
- Conflict resolution, rebase/merge
- Remote Git references in the job

## Future

- Git for Queries, Dashboards, and Workflows
- OAuth and SSH support
- Private Git server access

The image shows two overlapping dialog boxes from the Databricks interface. The top dialog, titled "Add Repo", is for creating a new repository. It includes fields for "Location" (set to "/Repos/vaibhav.sethi@databricks.com"), "Git repository URL" (https://github.com/sethi/big\_mono\_repo.git), "Git provider" (GitHub), and "Repository name" (big\_mono\_repo). It also has a checked option for "Create repo by cloning a Git repository" and an "Advanced" section with "Sparse checkout mode" checked and "Cone patterns" set to "project1/churn\_predictor Utils".

The bottom dialog, titled "Resolve conflict for DE\_repo branch\_1", shows a conflict resolution screen for the file "02\_data\_process.py". It lists two conflicting versions: "Use file from branch\_1" and "Use file from main". The right side of the dialog displays a diff view of the code, with line numbers 1 through 23. The code includes a Databricks notebook source, a command section, and Python code for using a CSV file and writing it out in Parquet format. The diff highlights changes between the current HEAD and the incoming change from branch\_1.

# Modern, Intuitive UX

# UX Challenges

## What we've heard from you

- Navigation is not unified today
  - Three separate product areas (DS&E, SQL, ML)
  - Different homepages
  - Two user settings and two admin settings pages
- Hard to remember where to find things
- UX interactions are different across the product

29

**One, consistent experience  
across all of Databricks**



# Notebook ❤️ Databricks SQL

## Making the Notebook an awesome SQL tool

- Unified SQL editing across both products
- Unified visualizations across the Notebook and DBSQL
- Bring SQL warehouses into the Notebook
- Schedule notebooks using SQL warehouses

The screenshot shows a Databricks Notebook interface titled "Investigating windfarm metrics". The SQL query is: `SELECT * FROM .windfarm`. The results are displayed in a table with columns: `temperature_00`, `wind_direction_00`, `wind_speed_00`, `temperature_08`, `wind_direction_08`, and `wind_speed_08`. The table contains 15 rows of data, with a note at the bottom indicating "Truncated results, show all".

	temperature_00	wind_direction_00	wind_speed_00	temperature_08	wind_direction_08	wind_speed_08
1	17.836880366007488	103.73823	5.6045	23.739837328592934	108.83645	9.227882000000001
2	23.82335535685221	116.91123999999999	9.608578	26.456441561381027	105.0664	8.891249
3	21.85329373677571	234.82682999999997	1.2878020000000001	27.67105801900228	188.00433	1.9217148000000002
4	21.094342867533367	283.488	5.087506299999999	25.718126932779953	296.25412	3.757636
5	18.42347844441732	283.8347	4.8929300000000001	23.14645353953044	301.06735	3.2225952
6	15.51370334625244	70.591630000000001	3.6681495	22.78459231058757	157.31951999999998	2.9299054
7	13.96827824910482	279.4585	6.7601743	18.54185676574707	227.68999	4.4574943
8	13.640029271443687	194.35085	4.149229	20.540881474812824	188.68436	3.2073842999999997
9	13.985969861348472	282.7044	5.28328	19.01255448659261	301.545960000000004	2.7801354
10	14.925614992777504	291.88422	2.7405147999999997	21.359126408894856	179.8469	1.3697847
11	15.915756861368813	283.3805	2.421521	21.872821489969894	160.34723	1.011566
12	20.17403284708659	275.85974	5.1469727	22.871932347615555	239.57953999999998	2.205038
13	16.045138994852707	299.18652000000003	1.6818657	20.86613114674886	175.04363999999998	1.5664022
14	13.810901959737144	87.82553	4.4720283	20.749955495198567	108.47401	4.9307027
15	16.90955130259196					

The dialog box "Attach to an existing compute resource" shows options for "General cluster" and "SQL Warehouse". The "SQL Warehouse" option is selected. A "Data Team Endpoint" is entered in the text field. A "Summary" box displays "Size: Medium" and "Type: Serverless". At the bottom, there is a note: "Notebooks can only be attached to Pro or Serverless SQL Warehouses". Buttons for "Cancel" and "Attach" are visible.



# Unified Results Grid

## Sort, Filter, and Search with ease

**Demo Draft: Notebooks** Python ▾

File Edit View Run Help Last edit was 30 minutes ago Provide feedback

▶ Run all ● Weston Hutchins's Int... ▾ 📅 Schedule Share ^

Cmd 3 Python ▶ ▾ 📊 ▾ - ✕

```

1 # Load netflix_titles into a pandas df
2 titles_df = spark.sql("select * from users.neha_sharma.netflix_titles").toPandas()
3
4 display(titles_df)

```

Table ▾ + Preview 📄

#	id	title	type	description
1	ts300399	Five Came Back: The Reference Films		ection includes 12 World War II-era propagar
2	tm84618	Taxi Driver		y unstable Vietnam War veteran works as a i
3	tm154986	Deliverance		seeing the Cahulawassee River before it's tl
4	tm127384	Monty Python and the Holy Grail	MOVIE	King Arthur, accompanied by his squire, recruits his K
5	tm120801	The Dirty Dozen	MOVIE	12 American military prisoners in World War II are ord
6	ts22164	Monty Python's Flying Circus	SHOW	A British sketch comedy series with the shows being
7	tm70993	Life of Brian	MOVIE	Brian Cohen is an average young Jewish man, but thr
8	tm14873	Dirty Harry	MOVIE	When a madman dubbed 'Scorpio' terrorizes San Frai
9	tm119281	Bonnie and Clyde	MOVIE	In the 1930s, bored waitress Bonnie Parker falls in lov
10	tm98978	The Blue Lagoon	MOVIE	Two small children and a ship's cook survive a shipwr
11	tm44204	The Guns of Navarone	MOVIE	A team of allied saboteurs are assigned an impossible

Sorting  
Sort A-Z  
Sort Z-A





# Unified Navigation UI

The screenshot displays the Databricks Unified Navigation UI. At the top, there is a search bar with the text "Search data, notebooks, recents, and more..." and a user profile icon for "justin.kim@databricks.com". The left sidebar contains navigation options: "New", "Workspace", "Recents", "Data", "Workflows", "Compute", "SQL", "SQL Editor", "Queries", "Dashboards", "Alerts", "Query History", "SQL Warehouses", "Data Engineering", "Machine Learning", "Marketplace", "Partner Connect", "Disable new UI", "Provide feedback", and "Collapse menu".

The main content area is divided into three sections:

- Get started:** Features four cards: "Import and transform data" (with sub-links "Upload data" and "Delta Live Tables"), "Notebook" (with sub-link "Create a notebook"), "SQL query editor" (with sub-link "Create a query"), and "AutoML" (with sub-link "Start AutoML").
- Recents:** A list of recently accessed items, including "NYC Taxi Trip Analysis" (Dashboard, 1 minute ago), "bb dashboard1" (Dashboard, 2 minutes ago), "(Clone) fb dash 1" (Dashboard, 2 minutes ago), "test" (Notebook, 2 minutes ago), "sql notebook" (Notebook, 2 minutes ago), "fb1 create - dsml personal (test4)" (Notebook, 2 minutes ago), "Quick Dashboard for popularity\_bugbash.willard.menu" (Dashboard, 2 minutes ago), "Retail Revenue & Supply Chain" (Dashboard, 2 minutes ago), "math dash" (Dashboard, 2 minutes ago), and "rxjava-1.3.0.jar" (Notebook, 6 days ago). A "View more" link is at the bottom.
- Popular:** A list of popular items, including "ml.sentiment\_ab.joined\_requests\_v2 Monitoring" (Dashboard), "abc" (Notebook), "fine-tune-notebook" (Notebook), "Example image and language model usage" (Notebook), "Notebook" (Notebook), "Data Monitoring Pricing Benchmark Setup" (Notebook), "Docfooding" (Notebook), "feature\_store.online\_feature\_store\_example.wine\_static\_features" (Table), "feature\_store.fse\_bugbash.test\_table\_617b" (Table), and "main.default.own\_by\_all\_users" (Table). A "Provide feedback" link is at the top right.

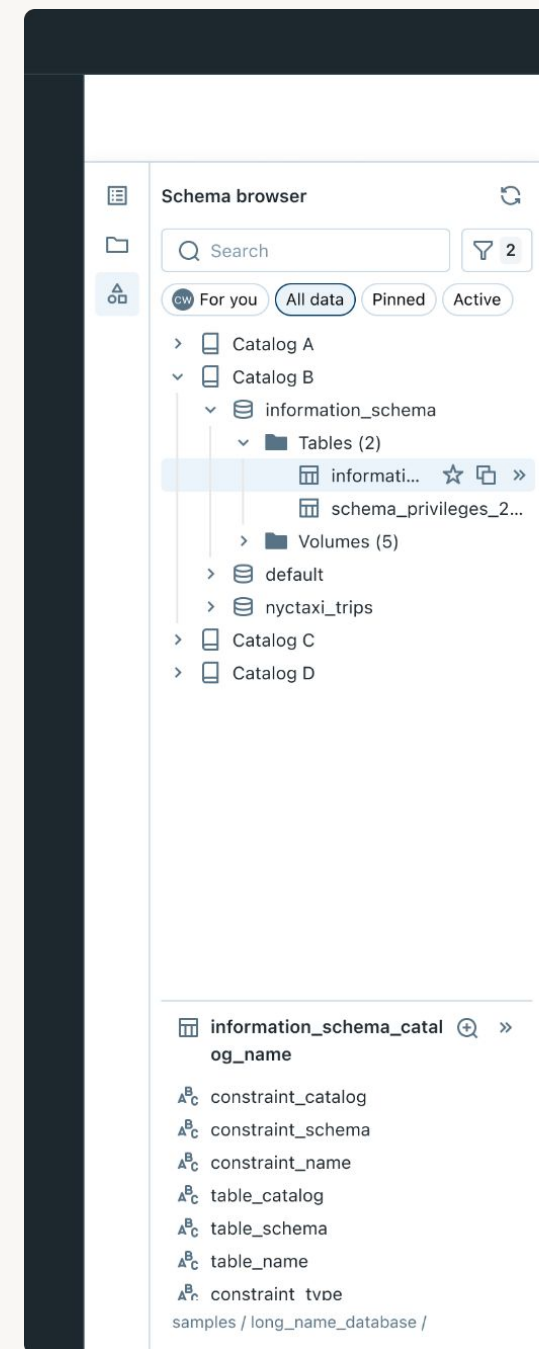
# Lakehouse, but everywhere!

## Get context without leaving the Notebook

- Schema Browser
  - Filter on active tables
  - Jump-to-table from a notebook cell
- File Browser
- Delta-Live-Tables Output

## Exploring ways to surface:

- Data Lineage
- Lakehouse Monitoring
- Expectations



# DEMO

# New cell look and feel

## Revisiting the main canvas of the Notebook

9/10/2021, 8:19:44 AM  
ML Shared

⋮

▶
✓
01:29

Cell 1

02
⌂
⋮

1 `SELECT * from fire_department_calls_for_service_sampled_1_percent_csv`

Result.df
+

	i <sub>2</sub> <sup>3</sup> _c0	i <sub>2</sub> <sup>3</sup> Call_Number	A <sub>C</sub> <sup>B</sup> Unit_ID	i <sub>2</sub> <sup>3</sup> Incident_Number	A <sub>C</sub> <sup>B</sup> Call_Type	A <sub>C</sub> <sup>B</sup> Call_Date	A <sub>C</sub> <sup>B</sup> Watch
1	3017367	90030257	E32	9000909	Medical Incident	null	null
2	4888798	10110234	M03	103393	Medical Incident	null	null
3	1083945	160612215	AM16	16024366	Medical Incident	null	null
4	2891475	90570326	T11	9016983	Medical Incident	null	null
5	1910186	131010278	E13	13034023	Structure Fire	null	null
6	2992423	90400347	87	9012092	Medical Incident	null	null
7	3996046	43300301	E03	4091239	Alarms	null	null
8	3458083	70710434	B02	7020972	Alarms	null	null
9	1417789	150153442	E43	15006088	Medical Incident	null	null
10	1891284	131260041	KM07	13042434	Medical Incident	null	null

↓
550 rows
↻
Refreshed 28 sec ago

# Our **vision** for the Notebook

The best authoring experience for the lakehouse

Lakehouse-aware through  
LakehouseIQ



Bringing your favorite IDE  
features to the notebook



Modern, intuitive UX



# Learn more at the summit!



Databricks  
Events App



## Tells us what you think

- We kindly request your valuable feedback on this session.
- Please take a moment to rate and share your thoughts about it.
- You can conveniently provide your feedback and rating through the **Mobile App**.



## What to do next?

- Discover more related sessions in the mobile app!
- Visit the Demo Booth: Experience innovation firsthand!
- More Activities: Engage and connect further at the Databricks Zone!



## Get trained and certified

- Visit the Learning Hub at the Databricks Zone!
- Take complimentary certification at the event; come by the Certified Lounge
- Visit our Databricks Learning website for more training, courses and workshops! [databricks.com/learn](https://databricks.com/learn)





# <SKIP> Outline

## Develop Like A Pro In Databricks Notebooks ×

Join us to learn about the latest enhancements to our most popular feature, Databricks Notebooks, and how they can streamline your data science and machine learning tasks.

In this session, we'll cover the new and upcoming Notebook features including:

- How we're bringing Lakehouse and Unity Catalog metadata into your authoring flows
- A new editor that introduces programming ergonomic improvements from your favorite IDEs
- Simplified access to compute so you can quickly and efficiently run your notebooks
- A look at the modern UX changes we're making to improve the look and feel of notebooks
- A preview of how we're integrating AI and LLMs into notebooks
- Streamlined search and navigation across Databricks

Experience: In Person

Track: DSML: Production ML / MLOps, Databricks Experience (DBX)

Type: Breakout

Level: Intermediate

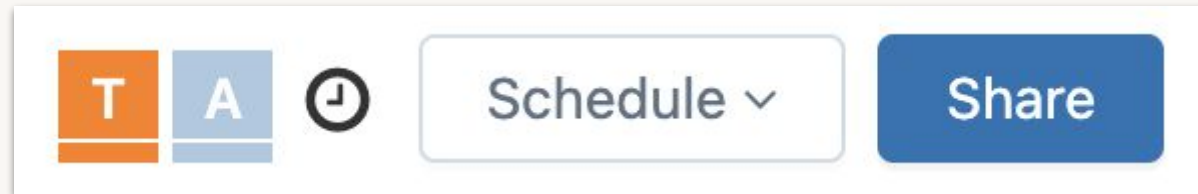
CLOSE



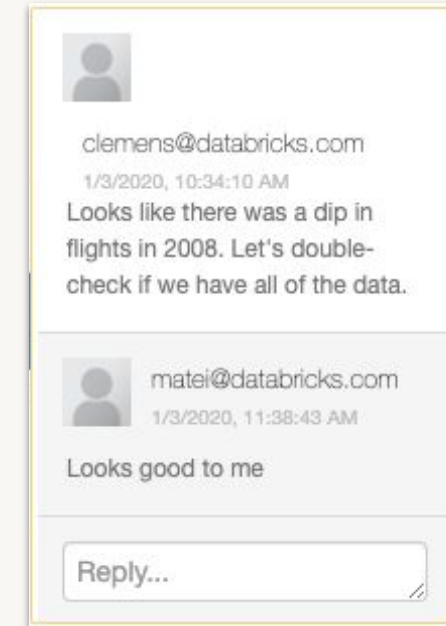


# Collaborate in real-time

Data science is a team sport



```
df_new = utils.scrub(df, drop="num_columns")
```



Users can share the same view of a notebook with **co-presence**

Users can **co-edit** in real-time with their colleagues to jointly iterate, debug, and more

**Comments** enable users to alert their colleagues to action items or interesting findings

# Jupyter-compatible

Bringing the power of the Jupyter ecosystem to the Databricks Notebook

*Newly GA!*

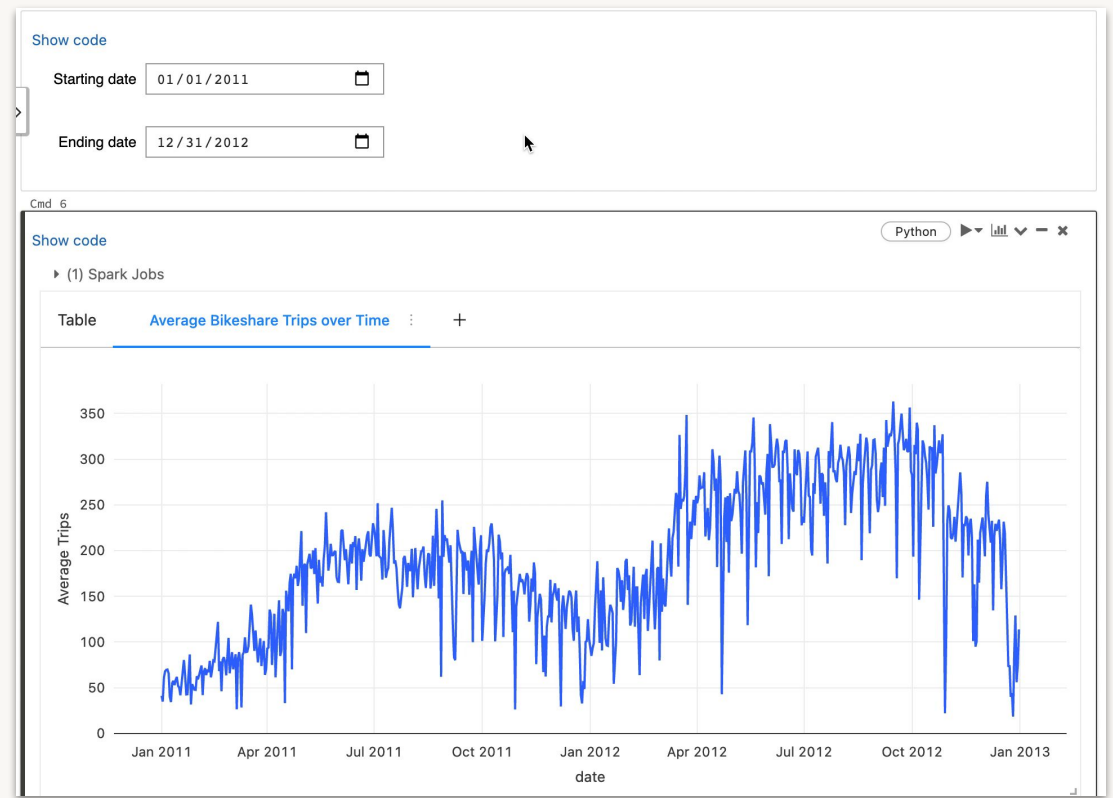
The Databricks Notebook uses the **IPython kernel** to power Python cell execution

*Public Preview!*

Use **ipywidgets** to turn notebooks into powerful, interactive apps

*Coming 2022*

Natively store Databricks notebooks as **.ipynb files** inside of Repos



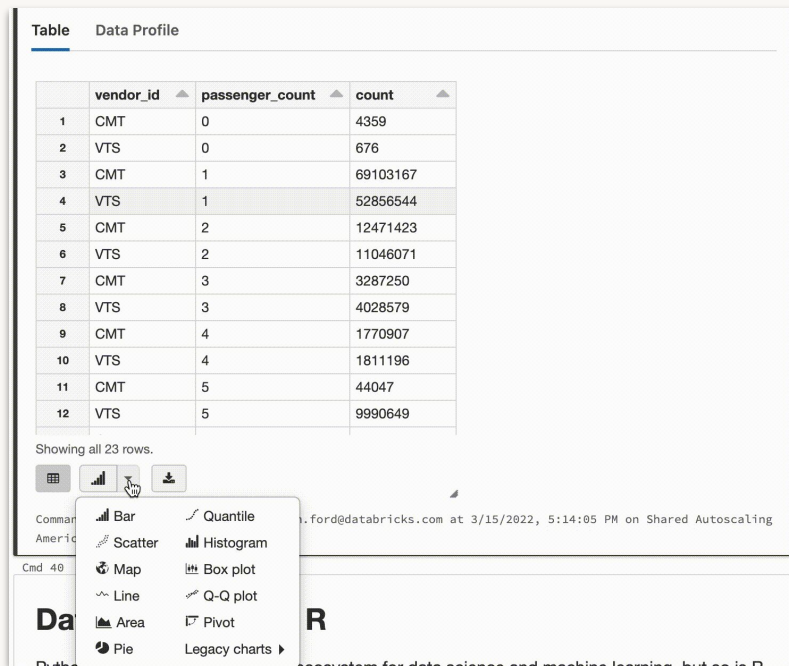
# Assistant Scenarios

## AI-Code Assist in the Notebook + SQL Editor

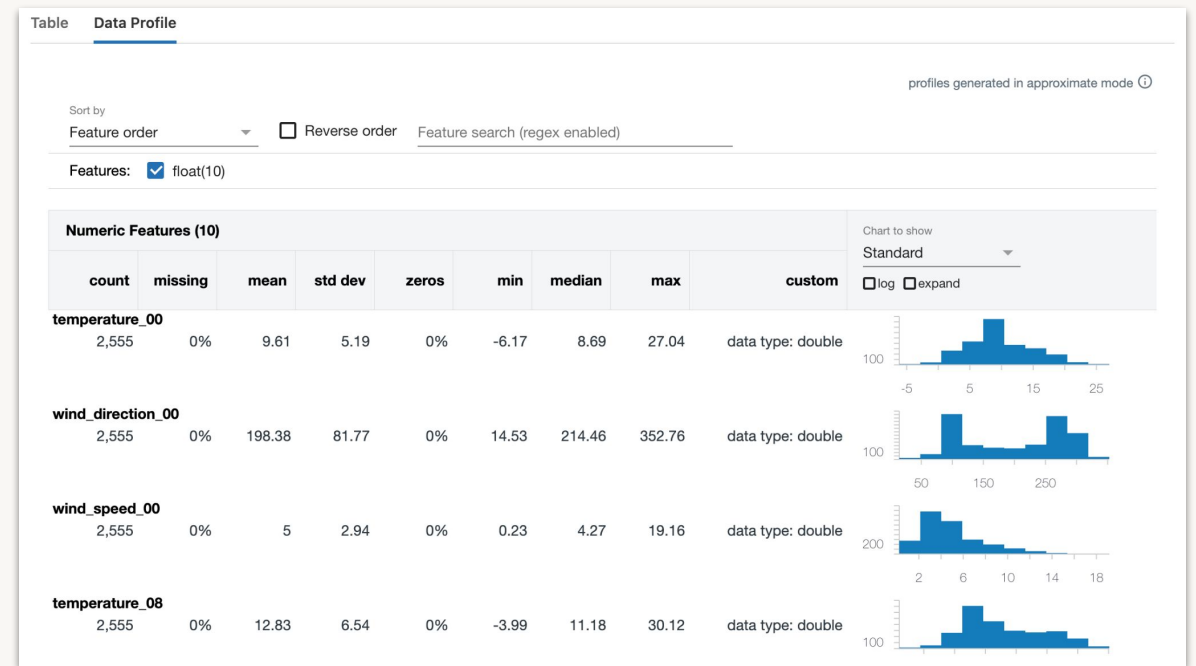
- **Complete Code:** suggestions-as-you-type
- **Generate Code:** NL Prompt → code/SQL
- **Explain Code:** Highlight a query and get an explanation in plain English
- **Debug Code:** Explain and fix syntax and runtime errors in your code with a single click.
- **Transform/Optimize Code:** Convert languages; RDD → DataFrame; Pandas → PySpark; Performance improvements

# Explore data efficiently

Native tools for visualizing and understanding data



Create **interactive charts** to visualize data in the Notebook with only two clicks



Summarize a data set's essential properties and statistics in a **data profile** with the push of a button

# Richer cell results in the Notebook

## Unified visualizations between Databricks SQL and the Notebook

Investigating windfarm metrics SQL Schedule Share

Shared Autoscaling

Cmd 1

```
1 SELECT * FROM clemens.windfarm
```

(2) Spark Jobs

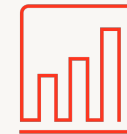
Table Power vs. Temperature Speed vs. Direction +

	temperature_00	wind_direction_00	wind_speed_00	temperature_08	wind_direction_08	wind_speed_08
1	17.836880366007488	103.73823	5.6045	23.739837328592934	108.83645	9.227882000000001
2	23.82335535685221	116.91123999999999	9.608578	26.456441561381027	105.0664	8.891249
3	21.85329373677571	234.82682999999997	1.2878020000000001	27.67105801900228	188.00433	1.9217148000000002
4	21.094342867533367	283.488	5.087506299999999	25.718126932779953	296.25412	3.757636
5	18.42347844441732	283.8347	4.892930000000001	23.14645353953044	301.06735	3.2225952
6	15.51370334625244	70.59163000000001	3.6681495	22.78459231058757	157.31951999999998	2.9299054
7	13.96827824910482	279.4585	6.7601743	18.54185676574707	227.68999	4.4574943
8	13.640029271443687	194.35085	4.149229	20.540881474812824	188.68436	3.2073842999999997
9	13.985969861348472	282.7044	5.28328	19.01255448659261	301.54596000000004	2.7801354
10	14.925614992777504	291.88422	2.7405147999999997	21.359126408894856	179.8469	1.3697847
11	15.915756861368813	283.3805	2.421521	21.872821489969894	160.34723	1.011566
12	20.17403284708659	275.85974	5.1469727	22.871932347615555	239.57953999999998	2.205038
13	16.045138994852707	299.18652000000003	1.6818657	20.86613114674886	175.04363999999998	1.5664022
14	13.810901959737144	87.82553	4.4720283	20.749955495198567	108.47401	4.9307027
15	16.90955130259196	282.4906	2.2472596	21.622171084086105	300.6086	2.0843556

Truncated results, showing first 1,000 rows. | 1.85 seconds runtime Refreshed 22 days ago



Create multiple charts and data profiles from a single cell data result



Craft a wider variety of visualizations with more visual appeal



Flexible chart configuration using the Databricks SQL chart builder

# UI-based data analysis and transformation

## Integrating bamboolib into the Notebook



Prepare, transform, visualize, and explore your data—all through a UI!



Be more efficient by spending less time writing boilerplate code



Operations in bamboolib generate code so users can see, customize, and learn from what happens via the UI



Enable citizen data scientists who know what they want to do to do it in Python

The screenshot shows a Databricks notebook titled 'Titanic Voyagers' in Python. The notebook content includes a cell named 'bam' with a 'Load data' button and a 'Data' tab. A dropdown menu is open, showing various transformation options: 'Search transformations', 'Select or drop columns', 'Filter rows', 'Sort rows', and 'Group by and aggregate (default)'. The 'Explore DataFrame' button is highlighted in green. Below the menu, a table of data is visible with columns 'i', 'Pclass', 'o Name', 'o Sex', 'f Age', and 'i'.

i	Pclass	o Name	o Sex	f Age	i		
3		Braund, Mr. Owe...	male	22.0	1		
1		Cummings, Mrs. J...	female	38.0	1		
3		Heikkinen, Miss. ...	female	26.0	0		
1		Futrelle, Mrs. Jac...	female	35.0	1		
4	5	0	3	Allen, Mr. William...	male	35.0	0
5	6	0	3	Moran, Mr. James	male	nan	0
6	7	0	1	McCarthy, Mr. Ti...	male	54.0	0
7	8	0	3	Palsson, Master. ...	male	2.0	3
8	9	1	3	Johnson, Mrs. O...	female	27.0	0





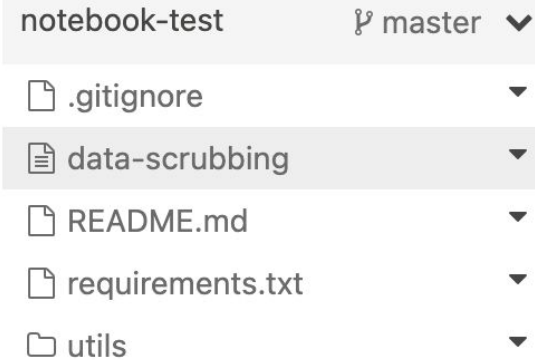
# Adaptable environments

Use standard libraries and custom modules in the Notebook

```
1 %pip install folium seaborn==0.11.1
```

```
1 import seaborn as sns
2
3 sns.violinplot(data=tips, x="day", y="total_bill", hue="smoker",
4               split=True, inner="quart", linewidth=1,
5               palette={"Yes": "b", "No": ".85"})
6 sns.despine(left=True)
```

Install Python libraries for a notebook  
without affecting other users with  
**%pip**



```
notebook-test      📄 master ▼
├── .gitignore      ▼
├── data-scrubbing  ▼
├── README.md       ▼
├── requirements.txt ▼
└── utils           ▼
```

```
1 import utils
2
3 df2 = utils.scrub(df1, drop="num_columns")
```

Import local modules using  
**arbitrary file support** when  
working in Repos

# Enterprise-ready

## Fulfilling essential governance requirements

Notebooks provide full access controls and identity management and can be shared for reading, execution, editing, or managing

All notebook access and user revisions logged with user identities

**New!**

All Notebook command executions tracked in **audit logs**

Permission Settings for: **Investigating windfarm metrics**

NAME	PERMISSION
tarek.madkour@databricks.com	Can Run
austin.ford@databricks.com	Can Manage (inherited)
ted.tomlinson@databricks.com	Can Read
admins	Can Manage (inherited)

Users dropdown: clem, clemens.mewald@databricks.com

Revision history dropdown:

- Jan 3 2020, 10:35 AM PST (clemens@databricks.com)
- Jan 3 2020, 10:34 AM PST (clemens@databricks.com)
- Jan 3 2020, 10:33 AM PST (clemens@databricks.com) [Restore this revision]
- Jan 3 2020, 10:25 AM PST (matei@databricks.com)
- Jan 3 2020, 10:24 AM PST (clemens@databricks.com)



# Developer Experience Themes

## Building an authoring experience for the Lakehouse

We believe the Lakehouse developer experience should...

- **Give users choice:** Use our native tools tailored to the Lakehouse, or use your favorite IDE — they're both great!
- **Be Lakehouse-aware:** Surface metadata, freshness, quality, popularity, tagging, and more — all in your dev environment
- **Feel modern and intuitive:** Bring powerful capabilities without getting in users' way