

DATA+AI
SUMMIT 2022

The Databricks Notebook

Front door to the
Lakehouse

ORGANIZED BY  databricks



Austin Ford

Senior Product Manager,
Databricks

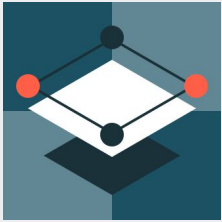


Rafi Kurlansik

Lead Product Specialist,
Databricks

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

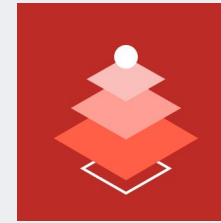
There are major hurdles in the way..



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

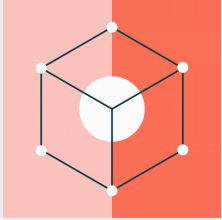


Disparate tools limit efficiency, reproducibility, and scale

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

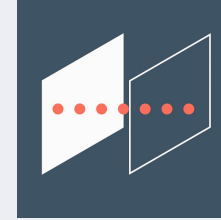
But—hurdles are meant to be jumped!



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

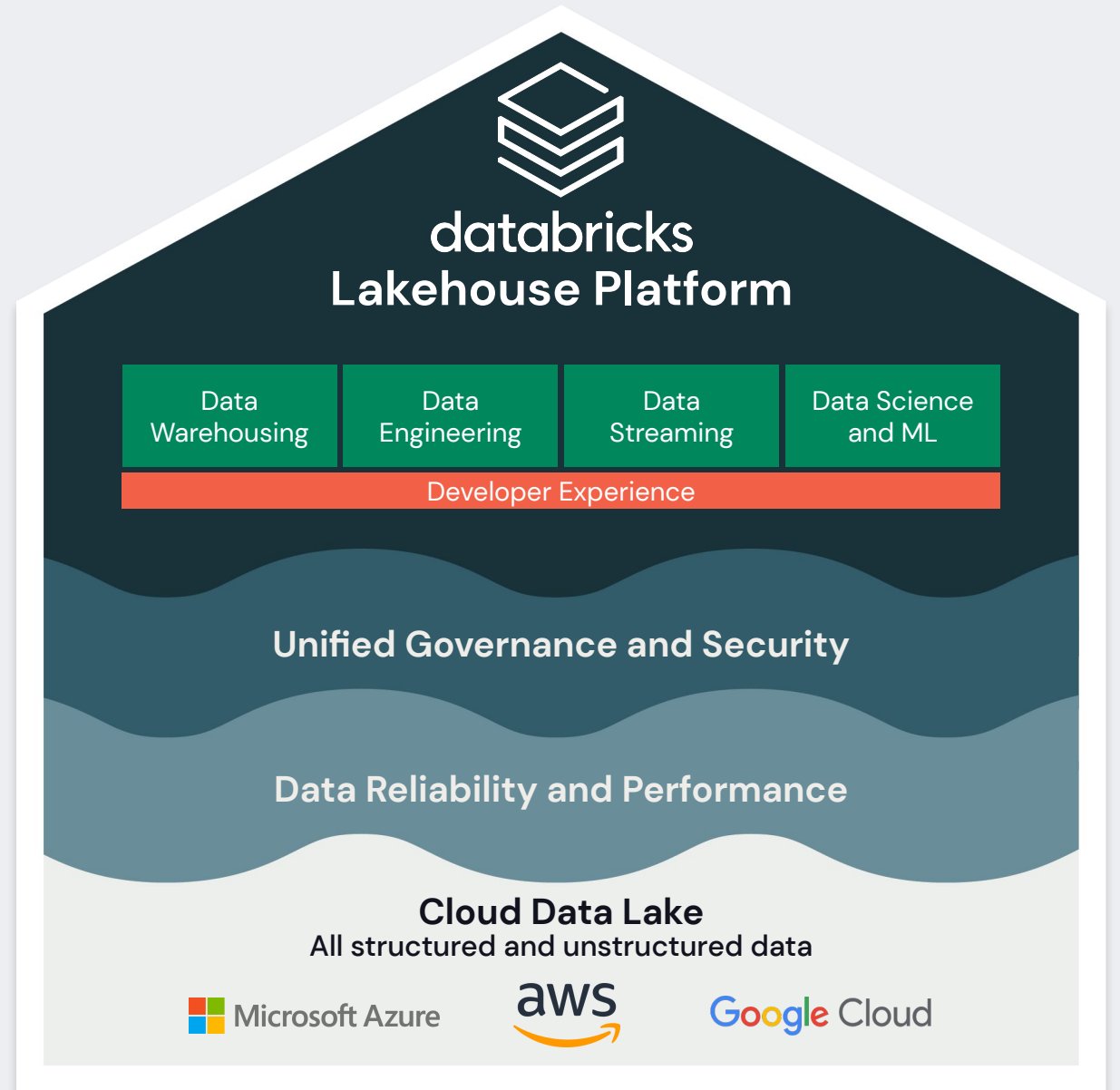


The Databricks Lakehouse developer experience accelerates the journey to insights

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

Databricks
Lakehouse is the
foundation for **all**
your data
analytics needs



Let's talk about the
Notebook—

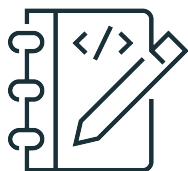
the front door to the
Lakehouse

Our **vision** for the Notebook

The front door to the Lakehouse

World-class data native developer experience

- Shortened distance to data insights, especially with Python and SQL
- Accelerated development
- Easy paths to production
- All the usual programming ergonomics



Foundation for sharing and consuming insights

- Effortless sharing of notebooks and results to anyone in your organization
- Optimized experiences for viewing and evaluating data assets



Portal to the entire Databricks Lakehouse

Connections to all the platform's powerful capabilities at the time and in the place users need, for..

- data engineering
- data science
- machine learning
- and more!



The Databricks Notebook

Through 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

The screenshot shows a Databricks Notebook interface. At the top, the title is "Learn to Use Databricks for Data Science" with a "Python" language selector. Below the title, there are icons for various actions like copy, paste, and refresh. The main content area shows a text block with the text: "mostly single-passenger trips, almost never more than 4. VTS (Verifone Transportation Systems) is the same, except 5+ passenger rides are not uncommon. They may specifically operate more, larger vans for pre-booking." Below this is a code cell with a SQL query:

```
1 %sql
2 SELECT vendor_id, passenger_count, COUNT(*) AS count FROM taxi_delta GROUP BY vendor_id, passenger_count ORDER BY passenger_count, vendor_id;
```

 Below the code cell, there are two tabs: "Chart" and "Data Profile". The "Chart" tab is active, showing a bar chart with "passenger_count" on the x-axis and "count" on the y-axis. The chart compares two vendor IDs: CMT (blue) and VTS (orange). The y-axis ranges from 0 to 60M. The x-axis ranges from 0 to 255. The chart shows that for 1 passenger, CMT has a count of approximately 65M and VTS has a count of approximately 55M. For 2 passengers, CMT has a count of approximately 15M and VTS has a count of approximately 10M. For 3 passengers, CMT has a count of approximately 5M and VTS has a count of approximately 3M. For 4 passengers, CMT has a count of approximately 2M and VTS has a count of approximately 1M. For 5 passengers, CMT has a count of approximately 10M and VTS has a count of approximately 5M. For 6 passengers, CMT has a count of approximately 5M and VTS has a count of approximately 2M. For 7 passengers, CMT has a count of approximately 2M and VTS has a count of approximately 1M. For 8, 9, 129, 208, and 255 passengers, the counts are very low.

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

Multi-language notebooks

Use the right tools for the job

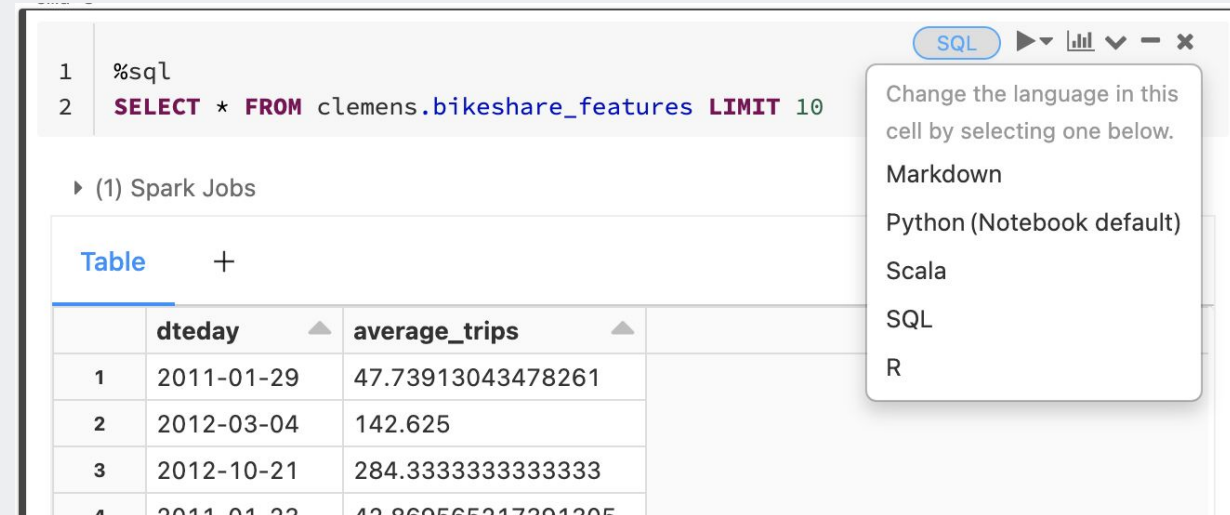
Users can mix and match languages based on their use case and preferred workflow, choosing from **Python, SQL, Scala, and R**

New!

Users can access the output of a SQL cell as a Python dataframe

Coming Soon

Users can re-style their Python code using the black formatter



The screenshot shows a notebook cell with the following content:

```
1 %sql
2 SELECT * FROM clemens.bikeshare_features LIMIT 10
```

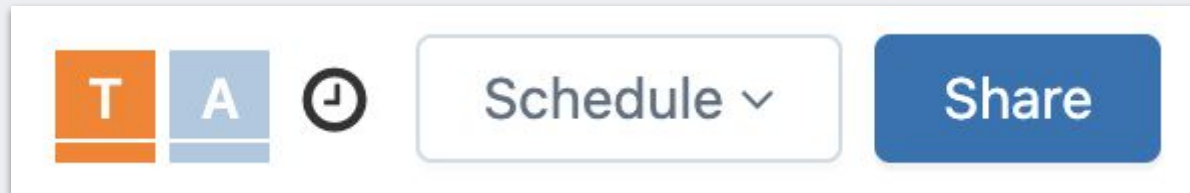
Below the code, there is a section for Spark Jobs and a table visualization. The table has the following data:

	dteday	average_trips
1	2011-01-29	47.73913043478261
2	2012-03-04	142.625
3	2012-10-21	284.3333333333333
4	2011-01-23	42.869565217391305

A language selection menu is open on the right, showing options: Markdown, Python (Notebook default), Scala, SQL, and R. The SQL option is currently selected.

Collaborate in real-time

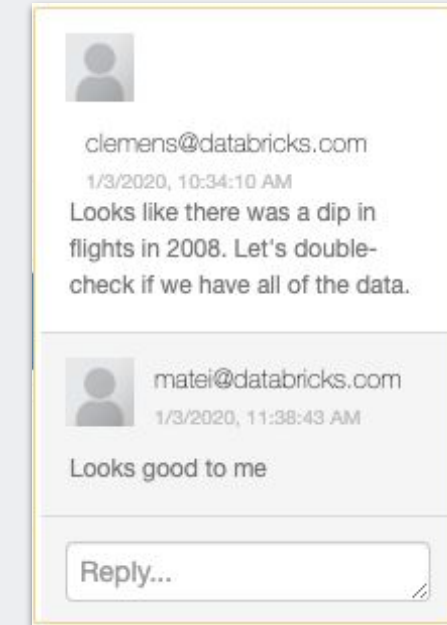
Data science is a team sport



```
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!

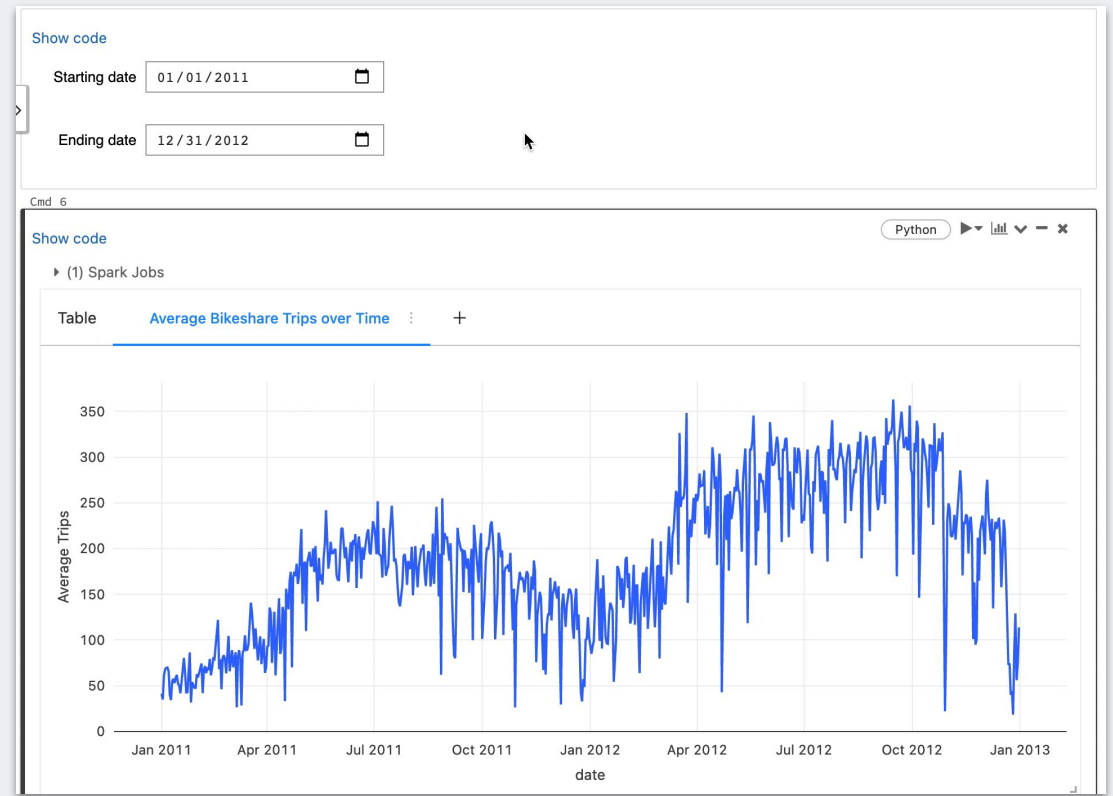
The Databricks Notebook uses the **IPython kernel** to power Python cell execution

Public Preview!

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

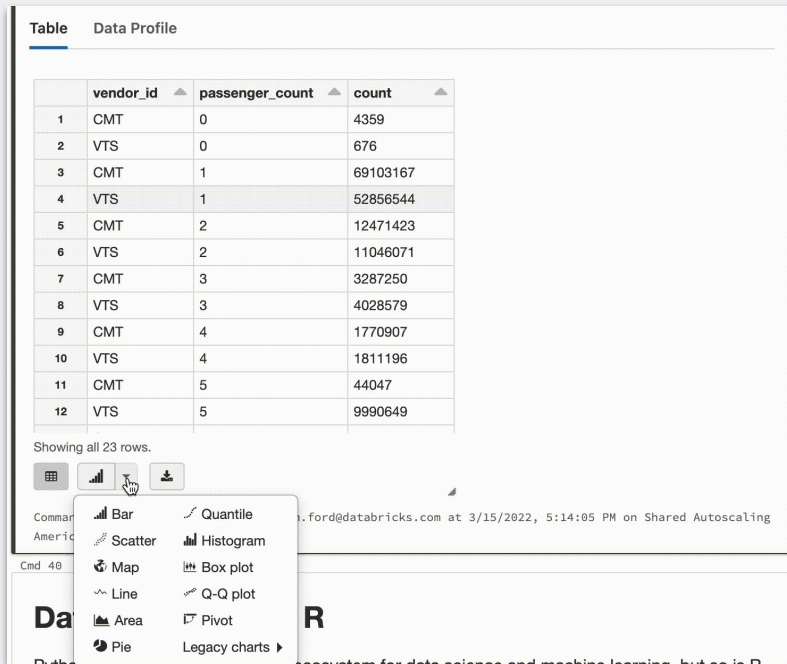
Coming 2022

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

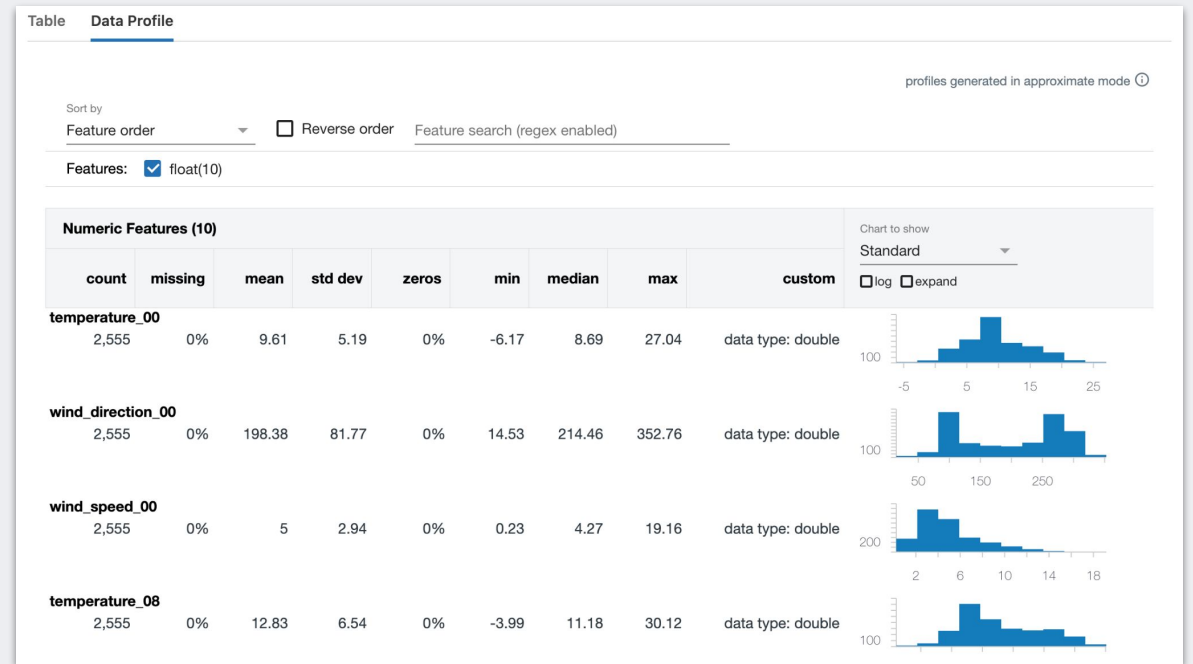


Explore data efficiently

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 displays the Bamboolib interface for a notebook titled "Titanic Voyagers" in Python. The interface includes a sidebar with navigation icons, a main workspace with a code cell containing "bam", and a data table. A dropdown menu is open, showing various transformation options such as "Select or drop columns", "Filter rows", "Sort rows", and "Group by and aggregate (default)". The data table shows columns for Pclass, Name, Sex, and Age, with rows of passenger data.

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		Allen, Mr. William...	male	35.0	0
5		Moran, Mr. James	male	nan	0
6		McCarthy, Mr. Ti...	male	54.0	0
7		Palsson, Master. ...	male	2.0	3
8		Johnson, Mrs. O...	female	27.0	0

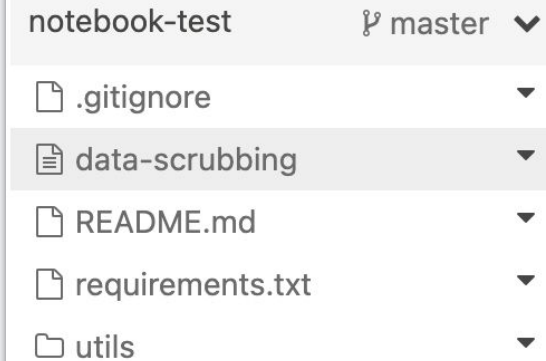
Adaptable environments

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**

The screenshot displays the 'Permission Settings' for a notebook titled 'Investigating windfarm metrics'. It shows a table of users and their permissions, along with a 'Revision history' panel.

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)

The 'Revision history' panel shows a list of revisions:

- 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)

Data exploration complete.

Now, how to get to
production?

From exploration to production

Enabling easy maintenance of assets over time

Take your notebook into production natively in Databricks

We've made it easy to take notebooks that power your simple ETL pipelines, reports, and dashboards and put them directly into production when they are ready



Use your favorite tool to productionize as you see fit

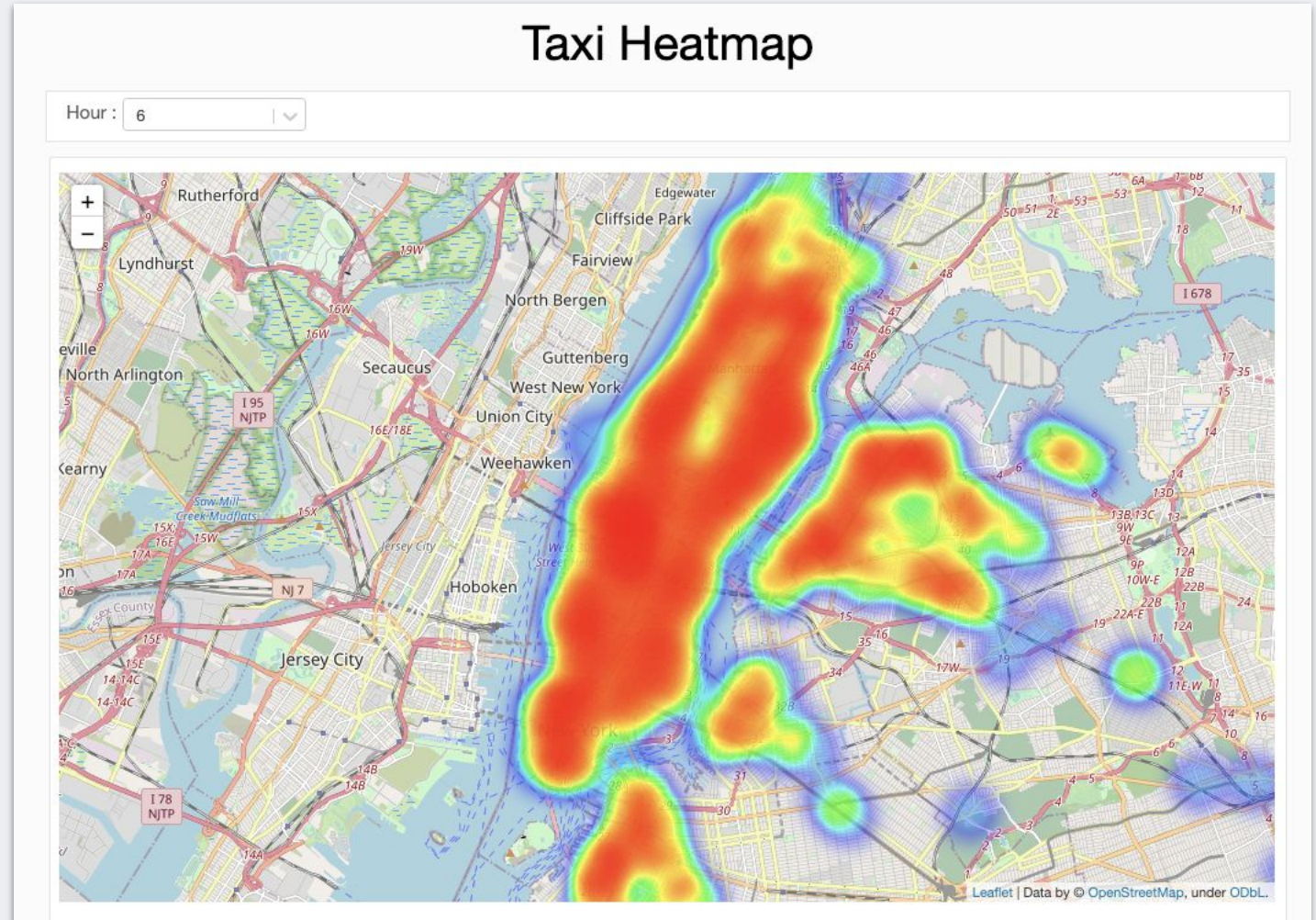
Sometimes you need more than a notebook, especially for complex projects—and Databricks enables this too



Faster paths to production

Quickly deliver value from work in the Notebook

Create interactive dashboards with parameters directly from the results of a notebook



Faster paths to production

Quickly deliver value from work in the Notebook

Investigating windfarm metrics (SQL) [Schedule] [Share]

Job name: Daily windfarm analytics

Schedule: Manual Scheduled

Every: Day at 21 : 30 (UTC-07:00) Pacific Ti...

Cluster: New Job Cluster (274.50 GB | 36 Cores | DBR 9.1 LTS | Spark 3.1.2 | Scala ...)

⚠ This cluster is started to run the job and deleted when the job completes

Parameters: Add

Alerts: austin.ford@databricks.cc Start Success Failure Add

[Cancel] [Create]

Schedule a notebook as a production job with the push of a button

Public Preview!

Add a name for your job...

Git Information

Git repo URL: https://github.com/austin-db/wind-analysis GitHub

Git ref (branch / tag / commit): main branch

[Cancel] [Confirm]

Type: Notebook Source: Git (main)

Path: notebooks/daily_wind_analysis

Cluster: New Job Cluster (274.50 GB | 36 Cores | DBR 9.1 LTS | Spark 3.1.2 | S...)

Parameters: Add UI | JSON

Pin jobs to git branches, tags, or commits using the Workflows integration with Repos

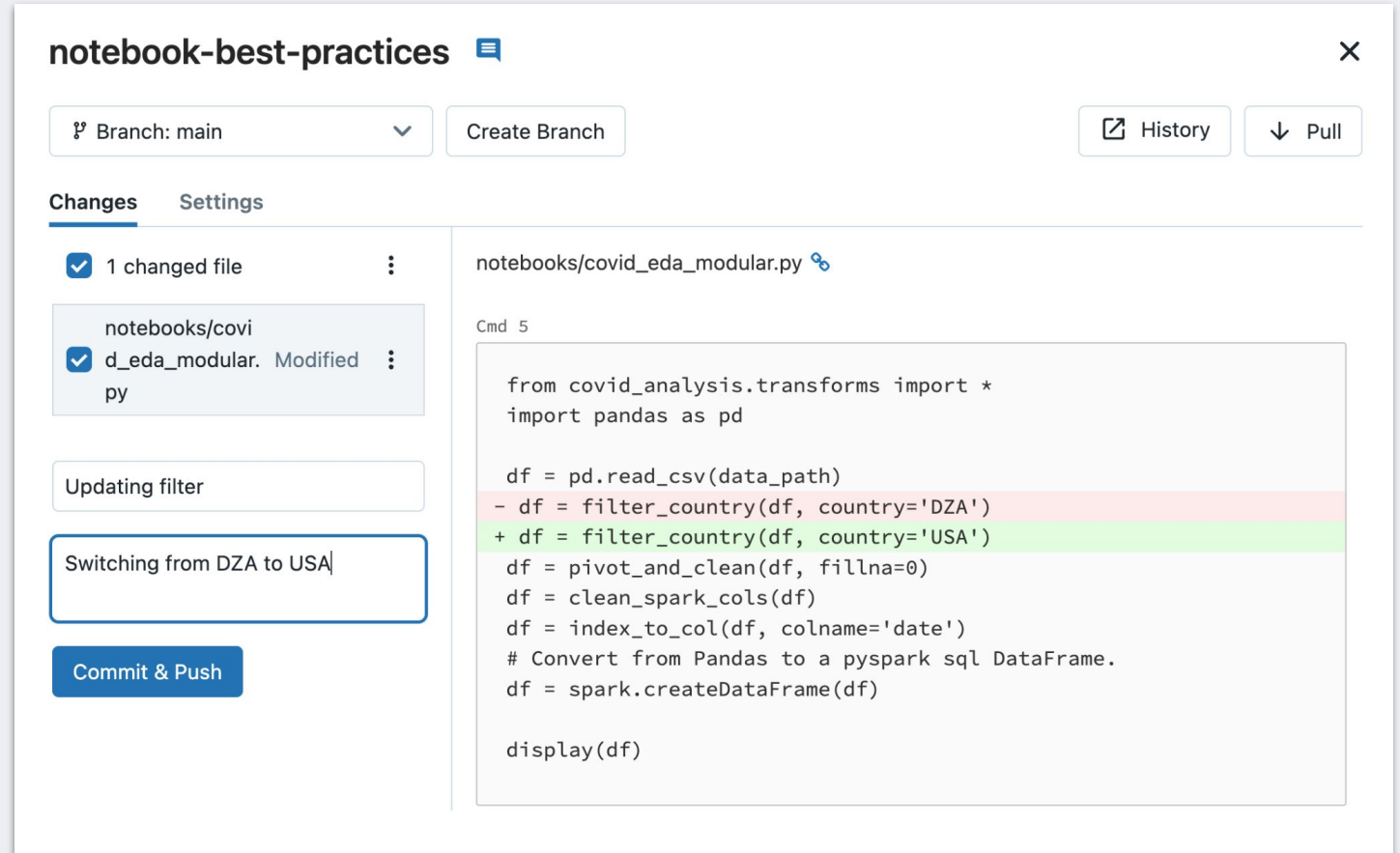
Best practices: Version control

Integrate with the most popular git providers

Databricks Repos support for git lets users collaborate on the same code without interfering with each others' work

Users' committed changes are logged in git history, making them fully **reproducible** and **auditable**

Protected branches and pull requests provide guardrails for what code moves to production



The screenshot shows the Databricks Repos interface for a repository named "notebook-best-practices". At the top, it indicates the current branch is "main" and provides a "Create Branch" button. There are also "History" and "Pull" buttons. Below this, the "Changes" section shows a list of files with checkboxes. One file, "notebooks/covid_eda_modular.py", is selected and marked as "Modified". Below the file list, there is a text input field for a commit message, which currently contains "Switching from DZA to USA". A "Commit & Push" button is visible below the message field. On the right side of the interface, the content of the selected file is displayed. The code is a Python script that filters data by country. The current commit shows a change from filtering by 'DZA' to 'USA', with the new line highlighted in green and the old line in red. The code includes imports for pandas and functions for reading CSV, pivoting, cleaning, and displaying data.

```
notebooks/covid_eda_modular.py  
Cmd 5  
from covid_analysis.transforms import *  
import pandas as pd  
  
df = pd.read_csv(data_path)  
- df = filter_country(df, country='DZA')  
+ df = filter_country(df, country='USA')  
df = pivot_and_clean(df, fillna=0)  
df = clean_spark_cols(df)  
df = index_to_col(df, colname='date')  
# Convert from Pandas to a pyspark sql DataFrame.  
df = spark.createDataFrame(df)  
  
display(df)
```


Best practices: Modular code

Simplify complex code projects for maintainability

The screenshot displays a Databricks workspace interface. On the left, a file editor shows a Python script named `transforms.py` with the following code:

```
1 import pandas as pd
2
3 # Filter by country code.
4 def filter_country(pdf, country="USA"):
5     pdf = pdf[pdf.iso_code == country]
6     return pdf
7
8 # Pivot by indicator, and fill missing values.
9 def pivot_and_clean(pdf, fillna):
10    pdf["value"] = pd.to_numeric(pdf["value"])
11    pdf = pdf.fillna(fillna).pivot_table(
12        values="value", columns="indicator", index="date"
13    )
14    return pdf
15
16 # Create column names that are compatible with Delta tables.
17 def clean_spark_cols(pdf):
18    pdf.columns = pdf.columns.str.replace(" ", "_")
19    return pdf
20
21
22 # Convert index to column (works with pandas API on Spark, too).
23 def index_to_col(df, colname):
24    df[colname] = df.index
```

On the right, a command terminal shows the execution of a command to automatically load modules:

```
Cmd 1
Automatically load modules

Cmd 2
1 %load_ext autoreload
2 %autoreload 2

Command took 0.38 seconds -- by rafi.kurlansik@databricks.com at 6/22/2022, 3:48:26 PM on 11.x
Shared Autoscaling

Cmd 3
1 | I

Cmd 4
Setup

Cmd 5
1 %pip install -r requirements.txt --quiet
```

File support in Repos and the File Editor enable users to refactor large notebooks into modules and common libraries

Modules can be imported and reloaded on demand with the **%autoreload** magic command

Best practices: Testing

Ensure code behaves correctly

Test your Python code with either notebooks or scripts using frameworks like pytest and unittest

Manually run test suites inside notebooks or at the command line with the **Web Terminal**

Automate testing by scheduling test notebooks with **Workflows**



Best practices: CI/CD

Automate code testing and deployment

Trigger continuous integration testing using the **Repos API** and support for git provider webhooks

Continuously deploy the latest production code using the **Workflows** integration with **Repos**

The screenshot displays a GitHub pull request interface. At the top, navigation links include Code, Pull requests (1), Actions, Security, Insights, and Settings. The main title is "An example change to be tested #3". Below the title, a green "Open" button is followed by the text "michaelp-db wants to merge 1 commit into main from example". A summary bar shows "Conversation 0", "Commits 1", "Checks 0", and "Files changed 1". A comment from "michaelp-db" is shown, stating "No description provided." Below the comment, a commit is listed: "An example change to be tested." by user "d4dba73". A section titled "Add more commits by pushing to the example branch on michaelp-db/terraform-playground." contains a "Checks" section. This section includes a warning "Some checks haven't completed yet" (1 in progress and 1 expected checks), a "Run pre-merge Databricks tests / unit-test-notebook (pull_request)" check (In progress, Required), and a "covid-eda-notebook" check (Expected, Required). A summary message states "Required statuses must pass before merging". At the bottom, there is a "Squash and merge" button and a note: "You can also open this in GitHub Desktop or view command line instructions." The bottom of the screenshot shows the start of a comment input area with "Write" and "Preview" tabs and a "Leave a comment" placeholder.

Demo

Recap

The Notebook today

What you can do right now

- Collaborate with your colleagues in our multi-language, interactive, and data-native Notebook that unlocks the power of the Databricks Lakehouse
 - Jump from pulling data with SQL to exploring it in Python without having to write any additional code
- Quickly and efficiently explore and visualize your data to derive shareable insights that deliver meaningful business value
 - Explore your data with less code using bamboolib
 - Use ipywidgets to turn insights into interactive applications for your stakeholders

The Notebook today

What you can do right now

- Take your work to production using Repos for version control and Databricks Workflows for scheduled, automated execution
- Alongside version control, employ other software development best practices like modular code, testing, and CI/CD

The Notebook tomorrow

New features coming soon

- Convey insights about your data using the new, easily configurable, and beautiful charts we are bringing to the Notebook from Databricks SQL
- Ipywidgets and bamboolib on GCP (with DBR 11.1)

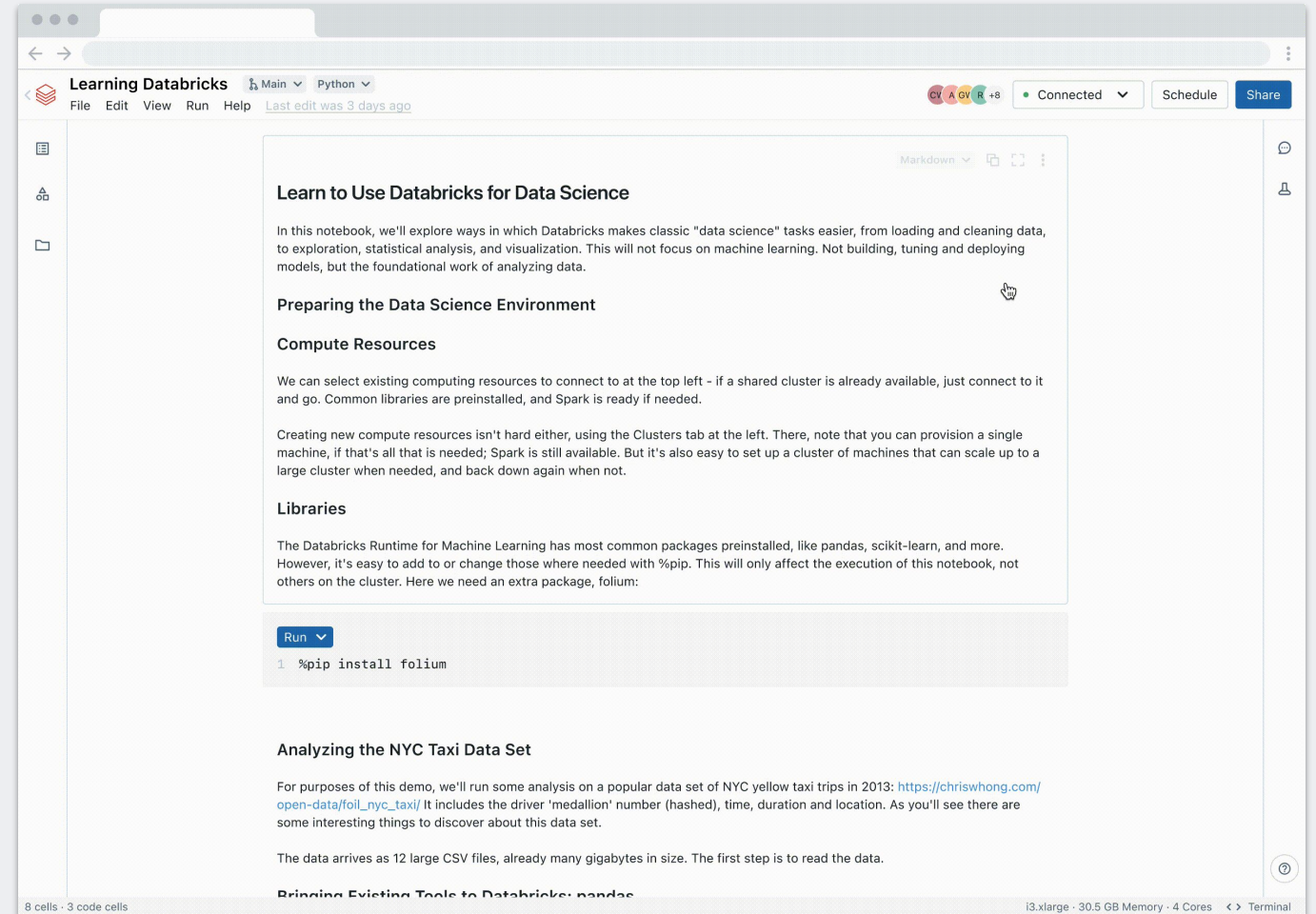
Where we're going

The future of the Notebook

Early ideas!

Focus areas

- EDA in Python and SQL
- Programming ergonomics
- Sharing and viewing
- Modern UI/UX
- Performance, reliability, and stability



DATA+AI
SUMMIT 2022

Thank you



Austin Ford

Senior Product Manager,
Databricks



Rafi Kurlansik

Lead Product Specialist,
Databricks