Department for Education

Introduction to



Databricks

EERC Workshop

Wojciech Klimowicz

Roadmap

01 **Workflow 1: work in the browser**

02 **Workflow 2: work in desktop IDE**

03 **Best practice**

Workflow 1: work in the browser

Microsoft Azure databricks

Search data, notebooks, recents, and more... CTRL + P

ADA Production s101p01-dbr-edap-02

New

Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Discover
Marketplace

SQL
SQL Editor
Queries
Dashboards
Genie Agents
Alerts
Query History
SQL Warehouses

Data Engineering
Runs
Data Ingestion
Visual Data Prep

AI/ML
Playground
Agents
AI Gateway
Experiments
Features
Models
Serving

LFS Labour Market Trends

Python Full

File Edit View Run Help Last edit was 7 minutes ago

Run all KLIMOWICZ, Wojciech'... Schedule Comments Share

Catalog Focus: ON

Catalog

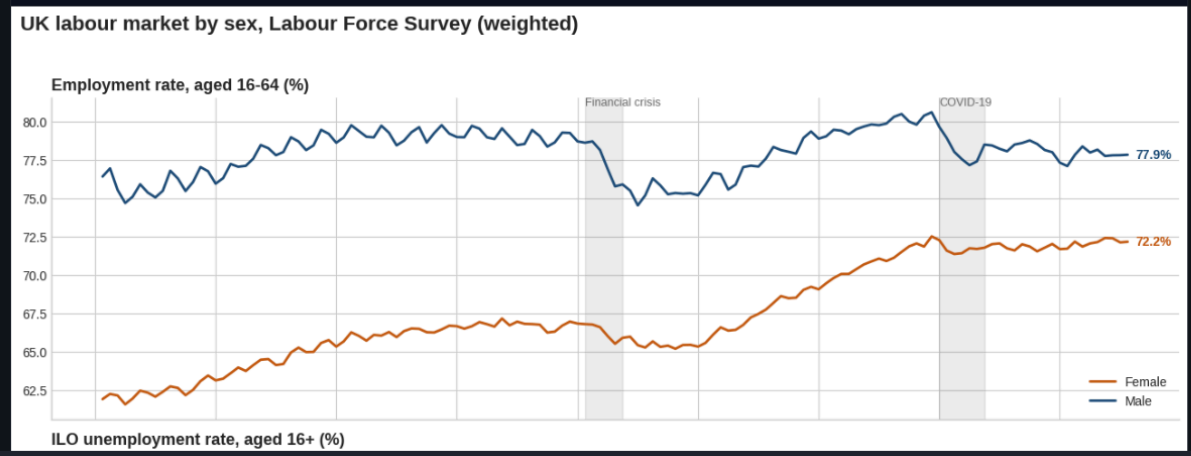
lfs.lfs

- My organization
  - catalog_40_copper_proj_tad_lfs
    - lfs
      - Tables
        - lfs_raw
        - lfs
          - lfs_sk
          - year
          - quarter
          - person_id
          - age
          - economic_activity
          - economic_activity_detailed
          - ethnicity
          - highest_qualification
          - highest_qualification_detailed
          - main_degree_subject_code
          - main_degree_subject_desc
          - main_degree_subject_group
          - main_occupation_code
          - main_occupation_desc
          - person_income_weight
          - person_weight
          - sex
        - aps_raw
        - aps
      - Volumes

03:56 PM (5s) 3

```
    (axes[0], "employment_rate", "Employment rate, aged 16-64 (%)"),
    (axes[1], "unemployment_rate", "ILO unemployment rate, aged 16+ (%)"),
]:
    for sex, g in pdf.groupby("sex"):
        g = g.sort_values("date")
        ax.plot(g["date"], g[col], label=sex, color=colours.get(sex), linewidth=2)
        last = g.iloc[-1]
        ax.annotate(f"{last[col]:.1f}%", (last["date"], last[col]), xytext=(6, 0),
                    textcoords="offset points", va="center", color=colours.get(sex), fontweight="bold")
    # Shade recessions / shocks for context
    for start, end, label in [("2008-04-01", "2009-07-01", "Financial crisis"), ("2020-01-01", "2021-07-01", "COVID-19")]:
        ax.axvspan(pd.Timestamp(start), pd.Timestamp(end), color="grey", alpha=0.15)
        if col == "employment_rate":
            ax.text(pd.Timestamp(start), ax.get_ylim()[1], label, fontsize=9, va="top", color="dimgrey")
    ax.set_title(title, loc="left", fontsize=13, fontweight="bold")
    ax.spines[["top", "right"]].set_visible(False)

axes[0].legend(frameon=False, loc="lower right")
fig.suptitle("UK labour market by sex, Labour Force Survey (weighted)", fontsize=16, fontweight="bold", x=0.01, ha="left")
fig.text(0.01, 0.005, "Source: LFS, catalog_40_copper_proj_tad_lfs.lfs.lfs. Weighted by person_weight; not seasonally adjusted.",
         fontsize=8, color="grey")
plt.tight_layout(rect=[0, 0.02, 1, 0.96])
plt.show()
```



Create a Python...

Auto-approve

E... New Notebook 2026-10-26... , upda...

E... New Notebook 2026-10-26... , adde...

E... New Notebook 2026-10-26... , upda...

Ran 2 cells

Renamed "New Notebook 2026-10-26 15:5...

Memory: Saved lfs-table

I've added a two-panel line chart to the notebook (now called "LFS Labour Market Trends"), built from lfs. Both cells ran without errors.

- **Top panel:** employment rate for people aged 16 to 64, split by sex.
- **Bottom panel:** ILO unemployment rate, which is unemployed people aged 16 and over as a share of everyone who is employed or unemployed.

The figures are weighted by `person_weight`. Each line is labelled with its latest value, and the 2008 to 2009 financial crisis and COVID-19 periods are shaded. The figures are not seasonally adjusted.

The table has all 137 quarters from 1992 Q2 to 2026 Q2 for both sexes. A few rows with no sex recorded are left out. I only checked the numbers for plausibility, not against published ONS figures:

1 asset

New Notebook 2026-10-26 15:54:55 +65 −2

@ for objects, / for commands, ↑↓ for history

Auto

Always review the accuracy of responses.

Browser workflow: pros and cons

Pros

Less setup

More powerful compute for R…

…depending on your laptop.

Large node on Databricks is 28GB RAM.

(You can still use Databricks SQL Warehouse from the IDE.)

Easy to share folders like SharePoint

Cons

You might not like notebooks

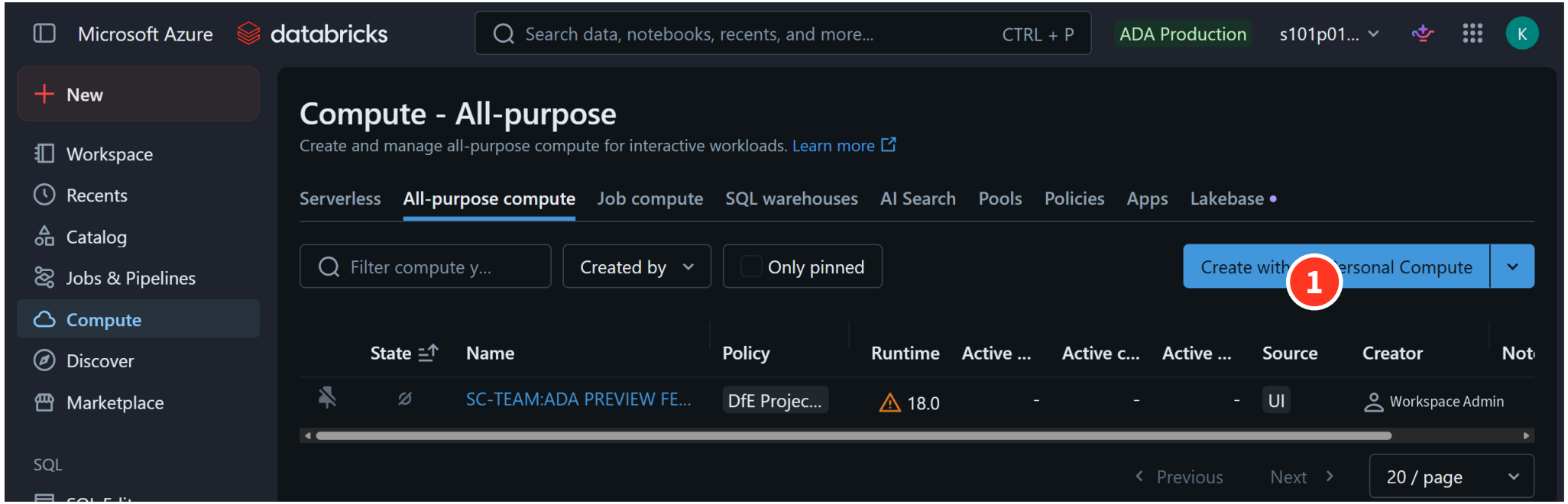
You might not like the browser

Some setup

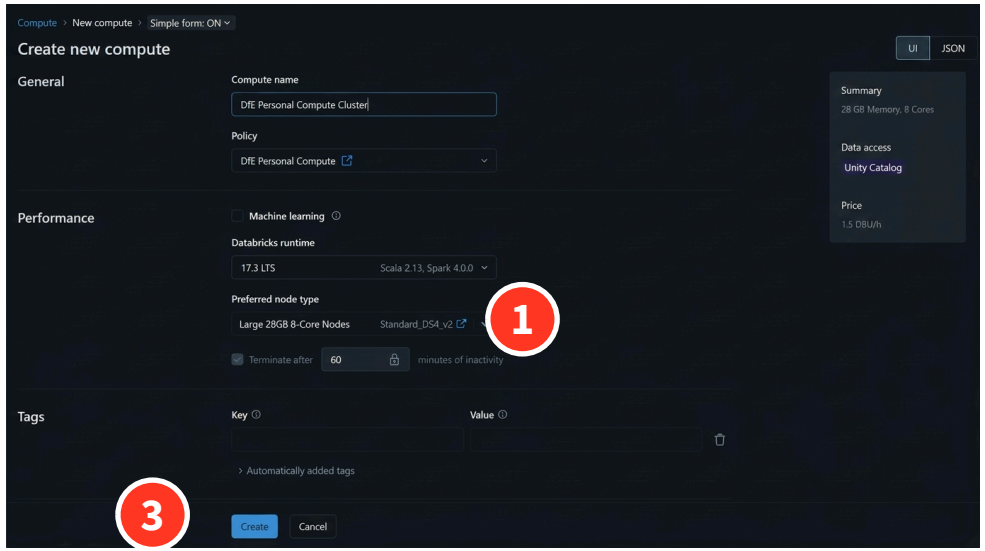
Python / SQL only: no setup.

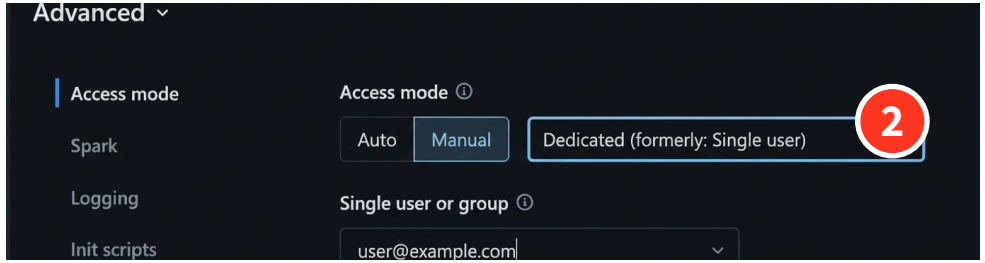
If working with R, create compute:

Compute → All-purpose compute → Create with **DfE Personal Compute**.



Setup: R compute





1. **Node type**
   Change to **Large** if you prefer.

2. **Advanced → Access mode**
   Manual → **Dedicated** → Your account.

3. **Create**
   Click **Create**. You only need to do this once.

   Takes about **5 minutes** to start.
   Switches off after **1 hour of inactivity**.
   Deleted after **1 month** without turning it on.

Languages on personal compute

1. SQL · summarise in Spark

```
%sql
SELECT AGE AS age,
       AVG(BUSHR) AS avg_hours, COUNT(*) AS n
FROM catalog_30_bronze.lfs.lfs_raw
WHERE BUSHR >= 0 AND BUSHR != 99
  AND AGE BETWEEN 21 AND 70
GROUP BY AGE
```

2. Python · keep larger groups

```
%python
hours = _sqldf.filter("n >= 30")
hours.createOrReplaceTempView("hours_by_age")
```

3. R · collect the summary and plot

```
%r
library(sparklyr)
library(tidyverse)
sc <- spark_connect(method = "databricks")
hours <- tbl(sc, "hours_by_age") |>
  collect()

ggplot(hours, aes(age, avg_hours)) +
  geom_line() +
  labs(x = "Age", y = "Mean weekly hours") +
  theme_minimal()
```

4. Markdown · explain the result

```
%md
Descriptive, unweighted means for ages 21–70.
Excludes negative hours and 99;
shows ages with at least 30 valid records.
```

Results next to code



Genie

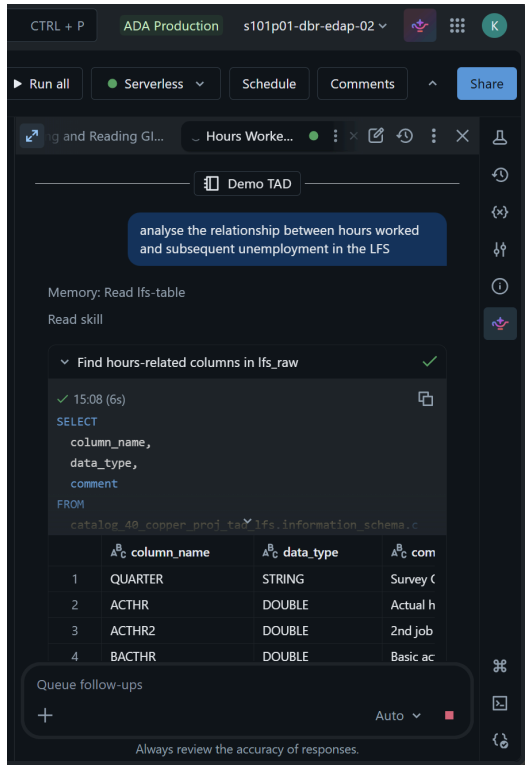
Ctrl+I: mini completion

Ask for a quick edit in the cell, then review and accept the suggestion.



Genie Code

Ask to help with a larger analysis in the side panel.



Connecting to data from notebooks in R

```
library(tidyverse)
library(sparklyr)

sc <- spark_connect(method = "databricks")

gias <- tbl(
  sc,
  I("catalog_30_bronze.gias_public.vw_gias_mains")
)
```

Genie connects SQL and Python as expected, but it struggles with R because the API is newer. (Don’t use SparkR, it’s old.)

Workflow 2: work on desktop IDE

File Edit Selection View Go Run Terminal Help

● test.R - example-eerc-project - Positron

New Open Search R 4.6.1 example-eerc-project

POSIT ASSISTANT

back to R.

- **1996 dropped:** it looked like a partial year, with a much smaller weighted total than later years and a visible artefact at the left edge.
- **Categories excluded:** "Does not apply", "No answer", "Don't know" and missing values. Shares are among people with a recorded qualification.
- **Possible coding changes:** the qualification categories may have been recoded over the period, so small steps in the series could be coding changes rather than real shifts. I didn't check this.
- **Not saved:** I ran the chart in the console only. It isn't in `test.R`, so say if you want it added.

✧ Split this chart by sex, or show degree-level share by age group.

✧ Add this plotting code to test.R.

3:29 PM

add it to the script

read: test.R lines 14-22 of 22

edit: test.R +38 -0

I added the qualification chart to the end of test.R. I haven't re-run it from the file. The code is the same as what I ran in the console, except that `levs` is now named `qual_levels`.

3:34 PM

Ask Posit Assistant...

R 45K Sonnet 5.5 (Medium)

test.R test.py

test.R > ...

```
41  filter(year >= 1997) |>
42  ggplot(aes(year, share, fill = factor(highest_qualification, l
43  geom_area() +
44  scale_y_continuous(labels = scales::percent, expand = c(0, 0))
45  scale_x_continuous(expand = c(0, 0)) +
46  scale_fill_brewer(palette = "RdYlBu") +
47  guides(fill = guide_legend(nrow = 3)) +
48  labs(
49    title = "Highest qualification of adults aged 25-64",
50    subtitle = "Weighted Labour Force Survey shares, 1997 onward
51    x = NULL, y = "Share of population", fill = NULL
52  ) +
53  theme_minimal(base_size = 12) +
54  theme(legend.position = "bottom")
55
```

CONSOLE TERMINAL PROBLEMS OUTPUT

~/Downloads/projects/example-eerc-p...

R 4.6.1 0% 519MB

Python...

```
+
  scale_y_continuous(labels = scales::percent, expan
d = c(0, 0)) +
+   scale_x_continuous(expand = c(0, 0)) +
+   scale_fill_brewer(palette = "RdYlBu") +
+   guides(fill = guide_legend(nrow = 3)) +
+
  labs(title = "Highest qualification of adults aged
 25-64",
+
       subtitle = "Weighted Labour Force Survey shar
es, 1997 onward",
+
       x = NULL, y = "Share of population", fill = N
ULL) +
+   theme_minimal(base_size = 12) +
+   theme(legend.position = "bottom")
>
```

SESSION HELP HISTORY VIEWER

VARIABLES 2.84GB Filter

DATA

qual	[186 rows x 4 columns] <tbl_df>

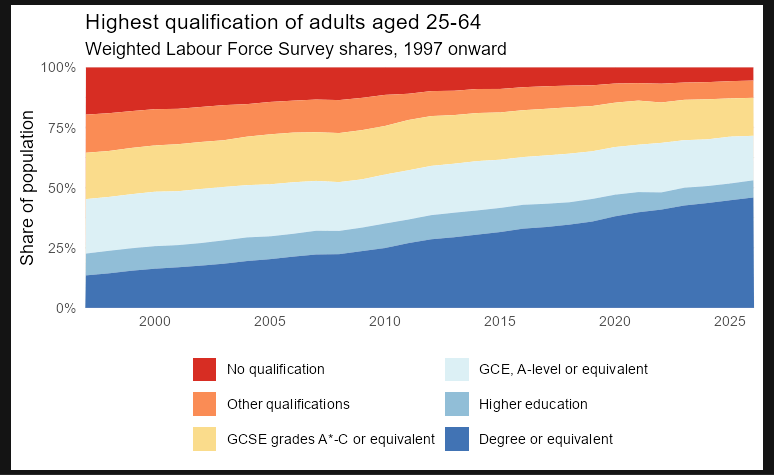
VALUES

con	<S4 class 'Spark SQL' [package "odbc"] with 5 slots>	Spark SQL
data	[con = <S4 class 'Spark SQL' [package "odbc"] with 5 slots>, src = [con = ...	tbl_Spark SQL
levs	"No qualification" "Other qualifications" "GCSE grades A*-C or equivalent" "GCE, ...	str [6]
lfs	[con = <S4 class 'Spark SQL' [package "odbc"] with 5 slots>, src = [con = ...	tbl_Spark SQL

PLOTS

Fit Auto

R 4.6.1 ggplot2 area chart 2



ggplot2 line cha...

ggplot2 area ch...

ggplot2 area ch...

0 0 Quarto: 1.10.18 Ln 16, Col 7 Spaces: 2 UTF-8 CRLF { } R Sign In

example-eerc-project - RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins | Posit Assistant | example-eerc-project

Posit Assistant

Fix databricks sql imp...

Append GIAS chart code to test.R

I made a stacked bar chart of GIAS establishments by opening year (2000–2026) and type, and added the code to the end of `test.R`. I ran it in the console, but I haven't re-run the script from the file.

The academy conversion programme shows up clearly. Opening records for local authority maintained schools run at about 150–250 a year until 2010. From 2010–11 they are almost all academies, at roughly 1,100–1,200 a year through 2011 and 2012. They are about 1,000–1,100 a year again in 2016–2018, then fall to about 400–500 a year in 2020–2022 before recovering. Free schools are a small slice, peaking around 120 in 2013.

- **What the bars count:** each bar counts establishment records by `OpenDate`. A school that converts to an academy gets a new record, so the 2010s academy bars are mostly conversions, not new schools.
- **Missing dates:** records with no `OpenDate` are dropped. That likely understates the older maintained-school bars, so I wouldn't compare the pre-2010 bars directly with the later ones.
- **2026:** it's a partial year.

Lines 21–27 of `test.R` are broken code that was already in the file: a `group_by` pipe with nothing feeding it, and a stray `mutate`. I left them alone, but they will cause errors if you source the whole file.

Chart the share of free school meals (PercentageFSM) by phase and region for open schools

Ask Posit Assistant...

31:00 | 85K | Sonnet 5.5 (Medium)

test.R

Source on Save | Run | Source

```
1  library(tidyverse)
2
3  con <- DBI::dbConnect(
4    odbc::databricks(),
5    httpPath = Sys.getenv("DATABRICKS_SQL_PATH"),
6    useNativeQuery = FALSE,
7  )
8
9  gias <- tbl(
10   con,
11   I("catalog_30_bronze.gias_public.vw_gias_mains")
12 )
13
14 gias |>
15   count() |>
16   show_query()
17
18 gias |>
19   filter(LAcode == 201)
20
21
22
23
```

14:8 | (Top Level) | Posit AI Pass: No completions available. | R Script

Console | Terminal | Background Jobs

R 4.6.1 · C:/Users/username /Downloads/projects/example-eerc-project/

```
+   geom_col(width = 0.85) +
+   scale_fill_manual(values = c("Local authority maintained schools" = "#8da0cb", "Academies" = "#e78ac3", "Free Schools" = "#fc8d62")) +
+   scale_x_continuous(breaks = seq(2000, 2026, 4)) +
+   scale_y_continuous(labels = scales::comma, expand = expansion(mult = c(0, 0.05))) +
+   labs(title = "Schools by opening year and type", subtitle = "Current GIAS establishment records (open and closed)",
+        x = NULL, y = "Establishments opened", fill = NULL) +
+   theme_minimal(base_size = 12) +
+   theme(legend.position = "bottom", panel.grid.minor = element_blank(), panel.grid.major.x = element_blank())
# A tibble: 3 × 2
  EstablishmentTypeGroupname            n
  <chr>                             <dbl>
1 Local authority maintained schools  1899
2 Academies                          11875
3 Free Schools                         951
```

Environment | History | Connections | Git | Tutorial

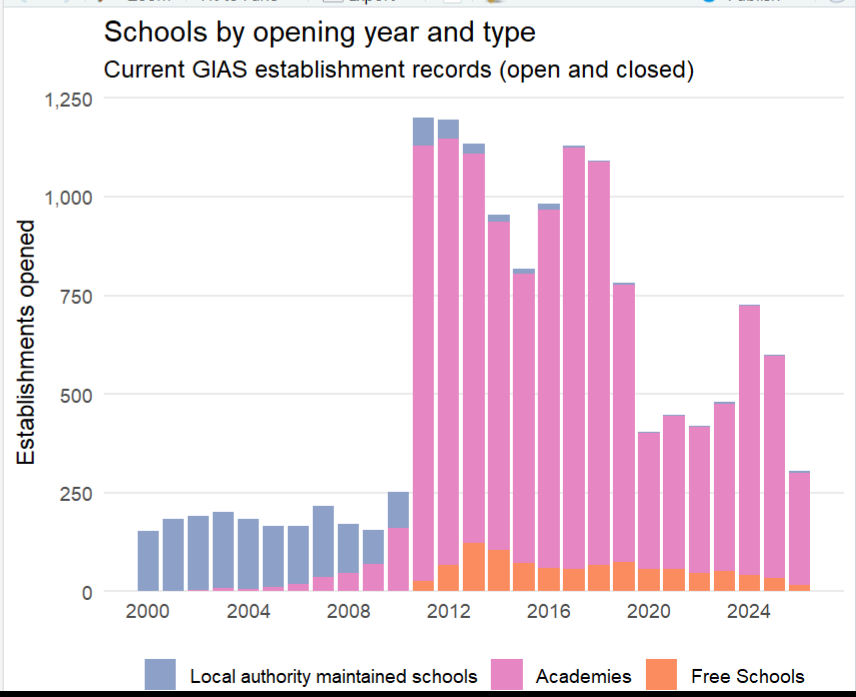
SQL | Help | Refresh Connection Data

catalog_10_gold - token@Spark SQL | Spark SQL

- catalog_10_gold
- catalog_20_silver
- catalog_30_bronze
- catalog_40_copper

Files | Plots | Packages | Help | Viewer | Presentation

Zoom | Fit to Pane | Export | Publish



IDE workflow: pros and cons

Pros

Use your favourite IDE…
…including Stata.

Better AI
Databricks endpoints (Sonnet/Opus 5.5) are better than Genie.

Cons

More (one-time) setup
Install tools, packages, and connection drivers.

Local memory is limited
Filter and aggregate in Databricks; download only what you need.

Laptop setup: overview

01 **Programming language**

02 **IDE**

03 **Databricks connection details + SQL warehouse**

What language?

R



R 4.6 with Rtools 4.5.

Installation instructions →

Python



Python managed with **uv**.

Installation instructions →

Stata



Search for **Stata** in MyDfE ↗ and select **Request**.

Choose your working environment

Positron



Install →

RStudio



Install →

VS Code



Install →

Claude Code / Terminal



Install →

Stata



MyDfE ↗

Requires a separate IT ticket.

Supporting tools

Git



Track changes to code and project files.

Databricks ODBC

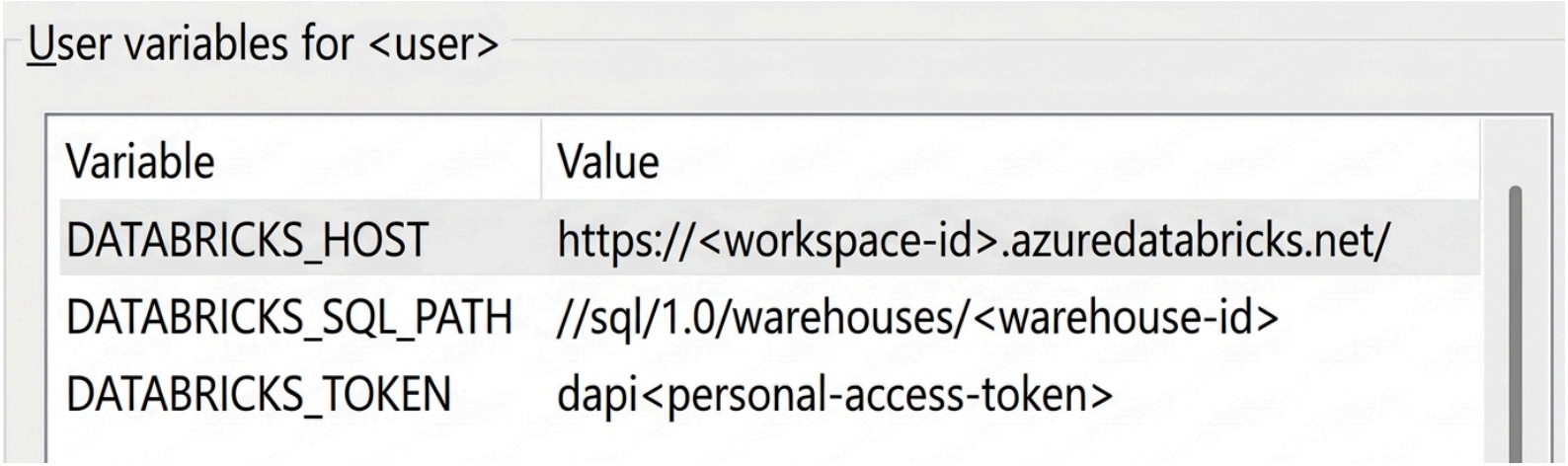


Driver to connect to Databricks SQL warehouse.

Installation instructions →

Connecting tools to Databricks

The end goal: three user variables

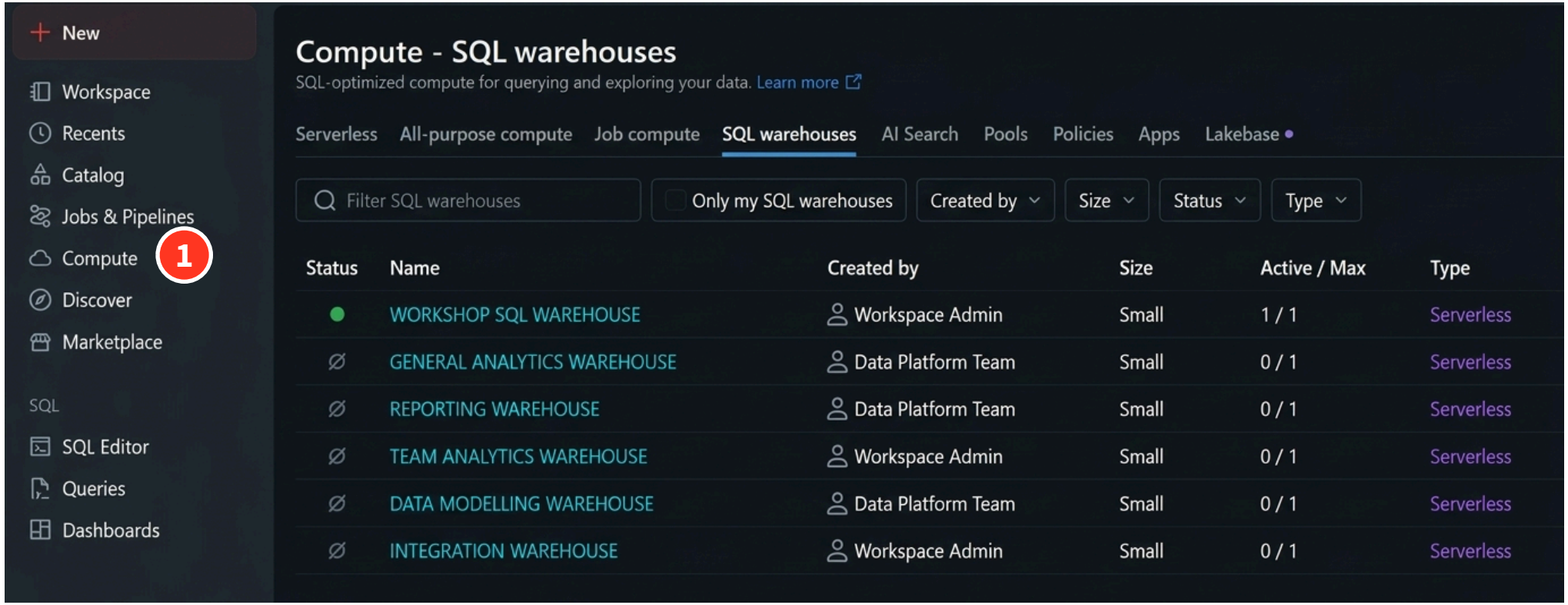


Your tools use these to connect to the workspace and SQL warehouse.

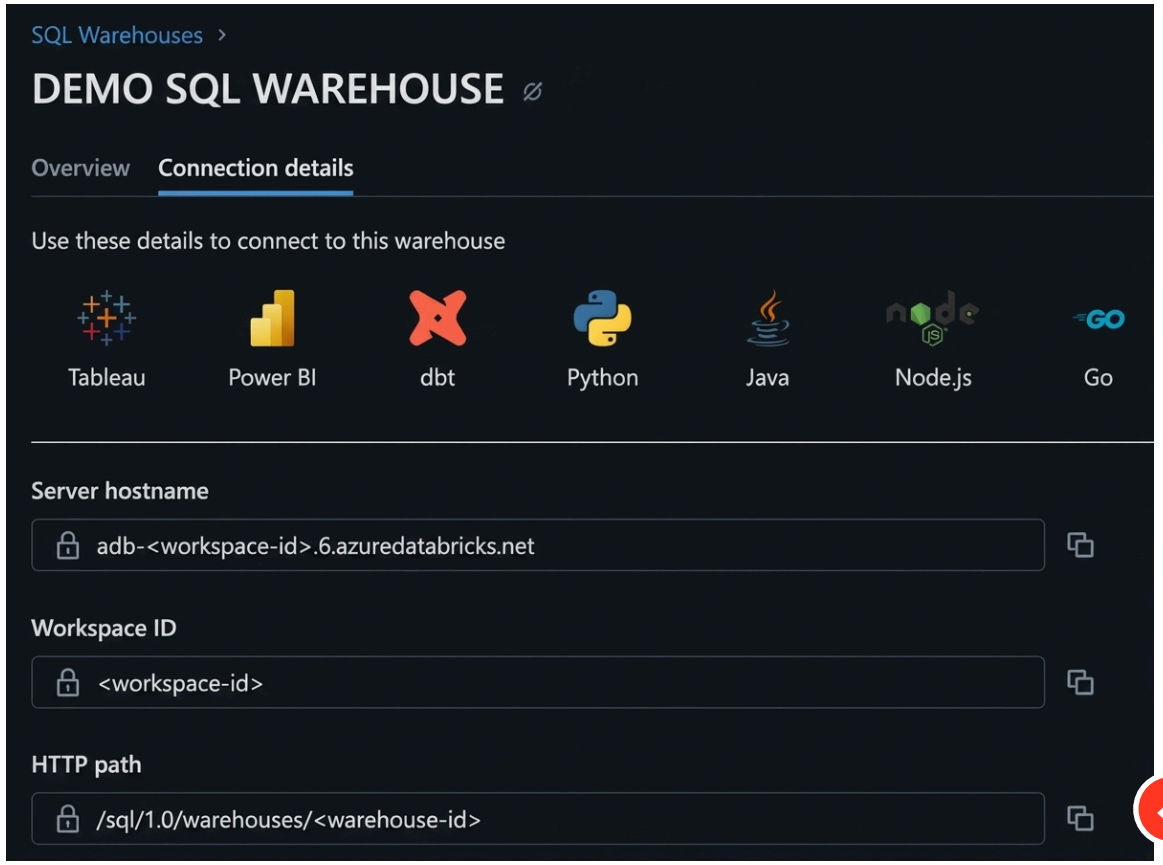
Next: find the connection details and token, then save them here.

Choose a SQL warehouse

In your Databricks workspace: **Compute** → **SQL warehouses** → **WORKSHOP SQL WAREHOUSE**.



Copy the connection details



Workspace URL

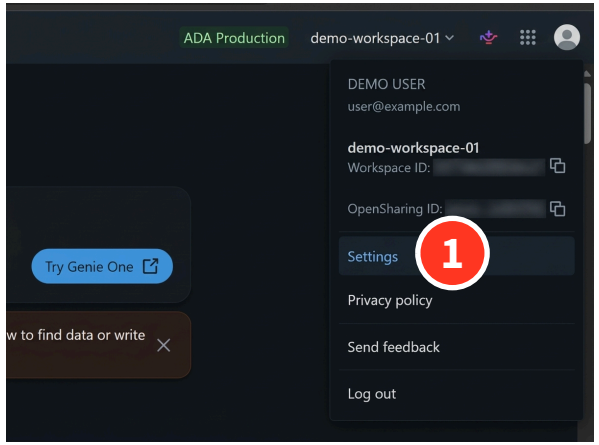
URL + https:// → `DATABRICKS_HOST`.

HTTP path

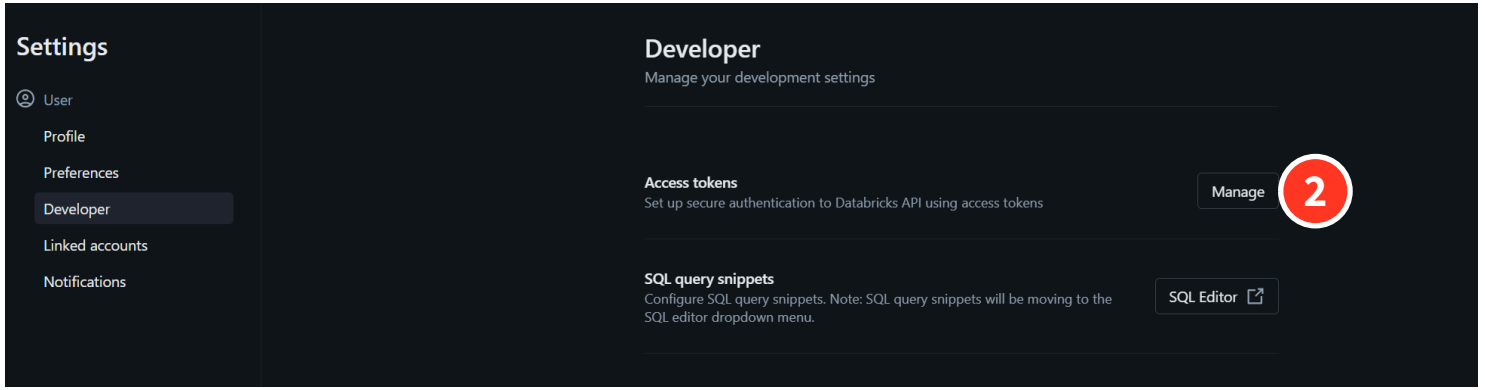
Copy the warehouse path → `DATABRICKS_SQL_PATH`. Adding an extra leading forward slash helps fix some bugs: `//sql/1.0/...`

Find your access tokens

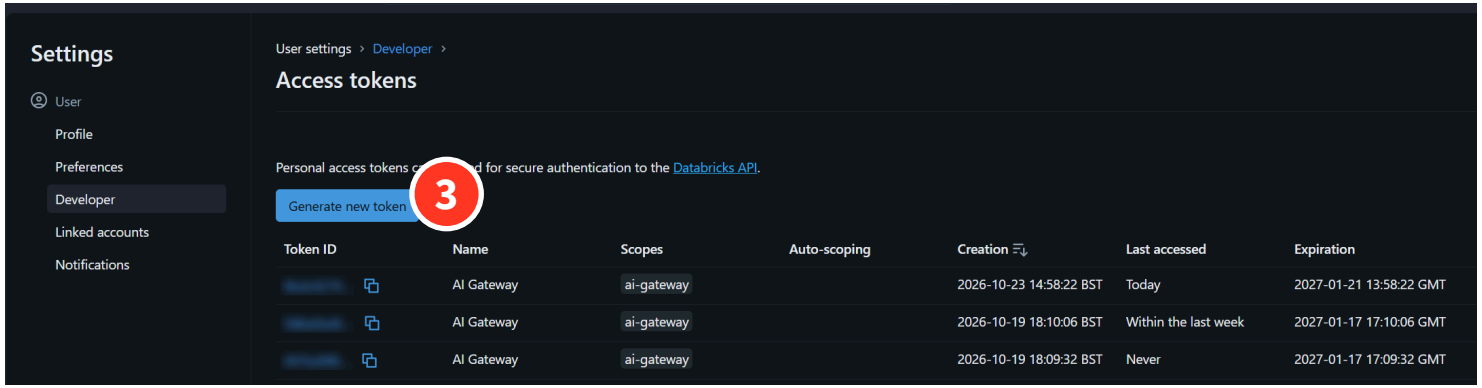
1 · Account → Settings



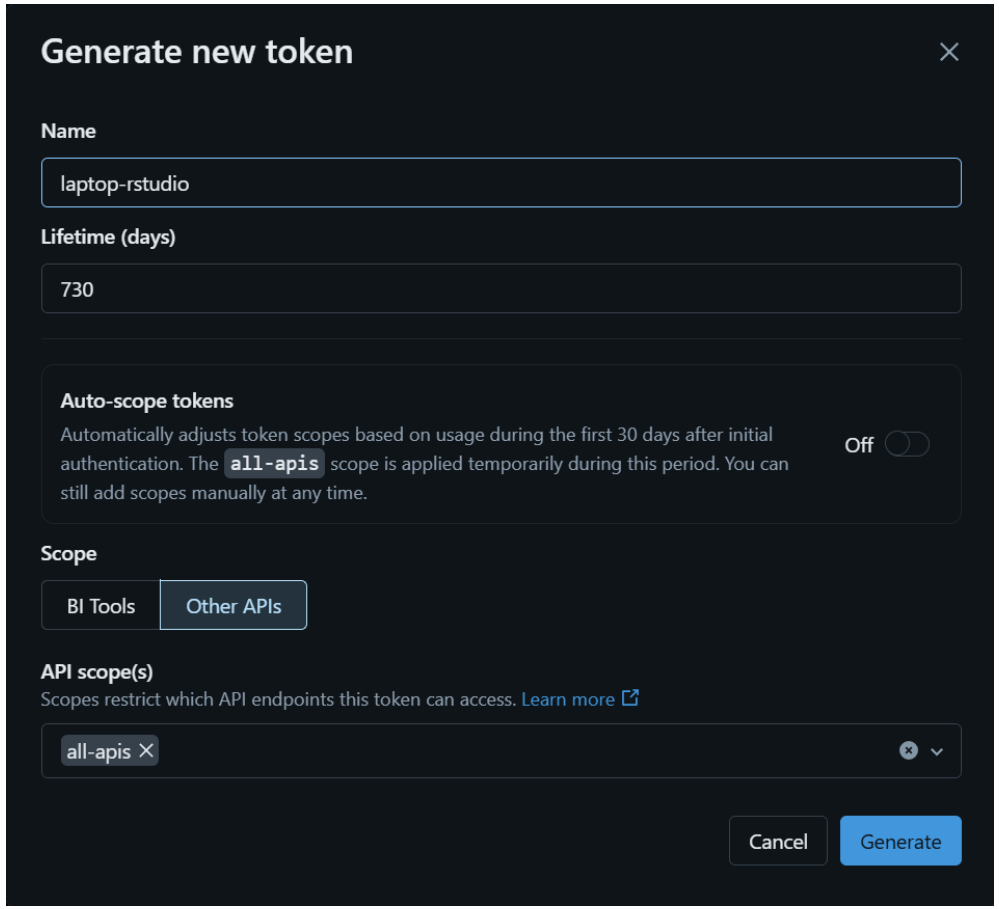
2 · Developer → Manage



3 · Generate new token

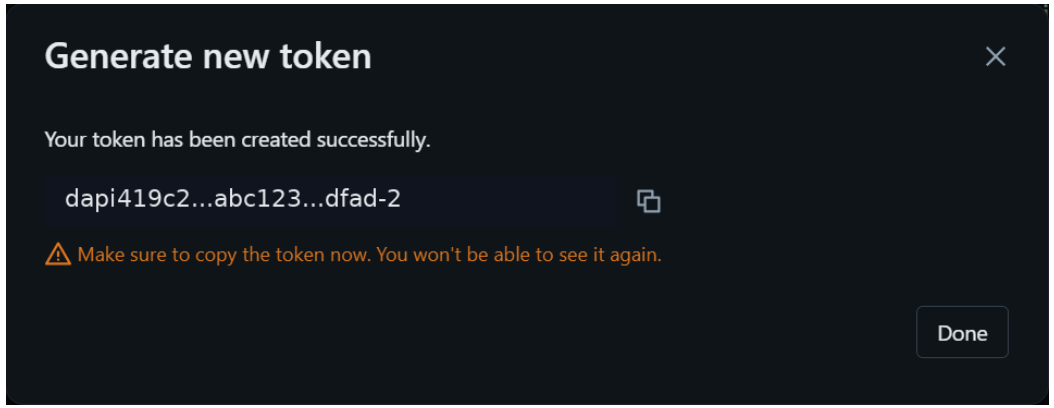


Generate and copy your token



Name, lifetime, and scopes

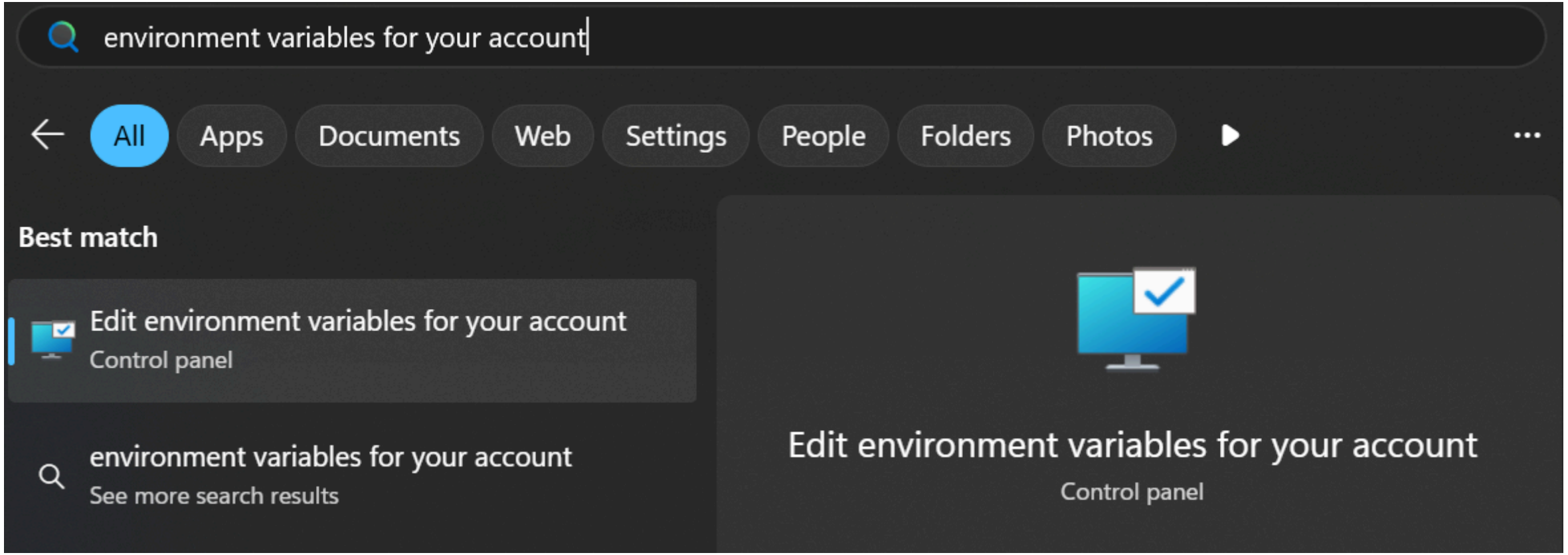
Name doesn't matter, lifetime can be set to 730 days, doing all scopes is easier.



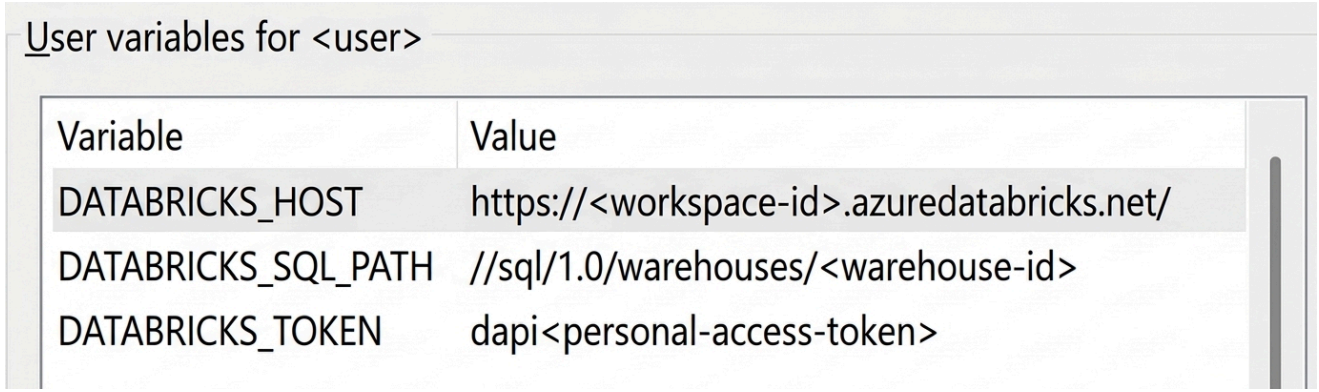
Copy once → `DATABRICKS_TOKEN`.

Open your user environment variables

In Windows Start, search for **environment variables for your account**.



Add three user variables



Under **User variables**, use **New…** or **Edit…**

`DATABRICKS_HOST`	Workspace URL with https://
`DATABRICKS_SQL_PATH`	Warehouse HTTP path
`DATABRICKS_TOKEN`	Your personal access token

Save with OK. Fully close and reopen your IDE.

If using R, install packages



Open the project folder in Positron or RStudio. Run in the R console:

```
install.packages(c("DBI", "odbc", "tidyverse", "dbplyr"))
```

You can install packages from

- CRAN: `install.packages("package")`
- GitHub: `remotes::install_github("username/package")`.

If using R: connect

1 · Open a warehouse connection

```r
library(tidyverse)
library(dbplyr)

con <- DBI::dbConnect(
  odbc::databricks(),
  httpPath = Sys.getenv("DATABRICKS_SQL_PATH"),
  useNativeQuery = FALSE
)
```

Set `useNativeQuery` to `FALSE` if you're writing back to your `eerc` area. `TRUE` is quicker if only reading.

If using R: query a remote table



```r
gias <- tbl(con, I("catalog_30_bronze.gias_public.vw_gias_mains"))
```

R

```r
query <- gias |>
  filter(
    LAcode == 201L,
    PercentageFSM >= 0,
    PercentageFSM <= 100
  ) |>
  mutate(
    fsm_share = PercentageFSM / 100,
    high_fsm = PercentageFSM >= 30
  ) |>
  select(URN, fsm_share, high_fsm)

query |> show_query()
```

SQL Translation

```sql
SELECT
  `URN`,
  `PercentageFSM` / 100.0 AS `fsm_share`,
  `PercentageFSM` >= 30.0 AS `high_fsm`
FROM
  catalog_30_bronze.gias_public.vw_gias_mains
WHERE
  (`LAcode` = 201) AND
  (`PercentageFSM` >= 0.0) AND
  (`PercentageFSM` <= 100.0)
```

`show_query()` displays SQL.

`query |> collect()` runs it and downloads the result.

If using Python: prepare the project



In the project folder, create an environment and install packages:

```
uv venv
uv pip install databricks-sql-connector pyarrow pandas
```

Use the three environment variables set earlier.

If using Python: connect



1 · Open a warehouse connection

```python
import os
from databricks import sql

host = os.environ["DATABRICKS_HOST"]
con = sql.connect(
    server_hostname=host.removeprefix("https://").rstrip("/"),
    http_path=os.environ["DATABRICKS_SQL_PATH"],
    access_token=os.environ["DATABRICKS_TOKEN"],
)
```

If using Python: fetch 10 rows



2 · Run SQL and return a small result

```
table = "catalog_30_bronze.gias_public.vw_gias_mains"

with con.cursor() as cur:
    cur.execute(f"SELECT * FROM {table} LIMIT 10")
    data = cur.fetchall_arrow().to_pandas()

print(data)
con.close()
```

SQL runs in Databricks; 10 rows become a pandas frame.

Run SQL saved in a file

gias-preview.sql

```sql
SELECT *
FROM catalog_30_bronze.gias_public.vw_gias_mains
LIMIT 10;
```

R · use the connection from the previous example

```r
sql <- readr::read_file("gias-preview.sql")
preview <- DBI::dbGetQuery(con, sql)
print(preview)
```

If using Stata

Either:

- **Connect directly with ODBC.** Investigate the connection setup in Stata's ODBC documentation ↗.

  Or

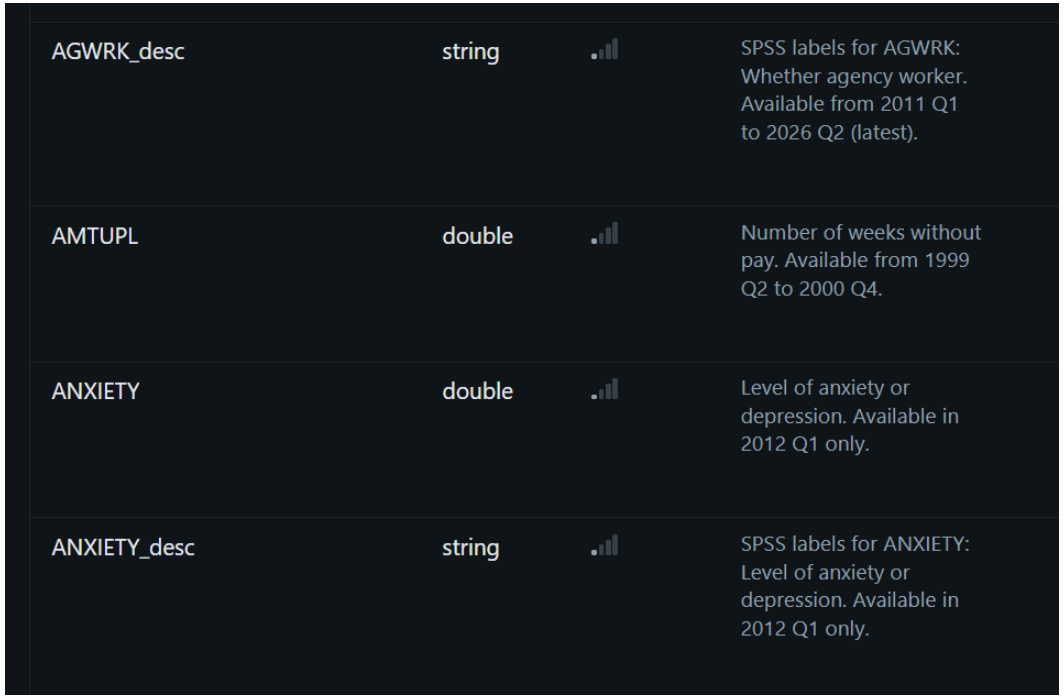
- **Use R to download the data** into a format Stata can read. Avoid CSV; try `haven::write_dta()` or `arrow::write_parquet()` if your Stata version supports Parquet.

Best practice

Documentation · Version control · AGENTS.md · Model settings · Computation

Documentation

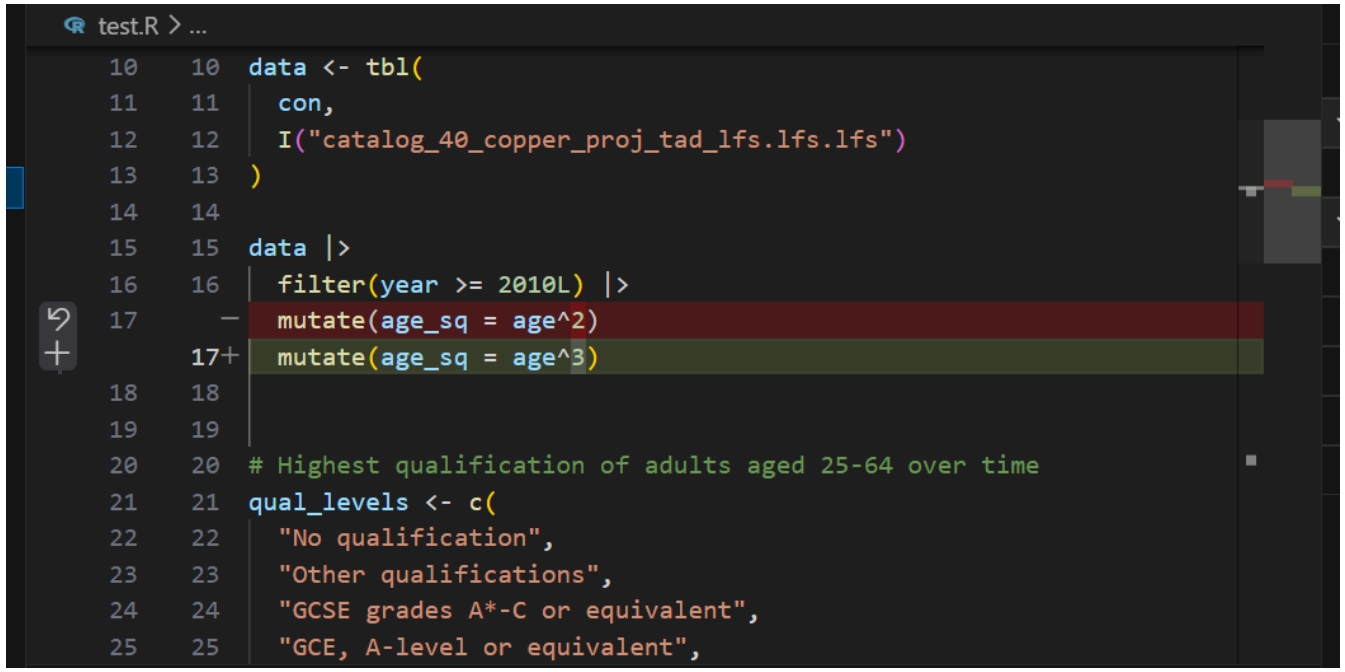
- Make sure the tables you're using have documentation for agents (and yourself).
- For TAD-managed data, this is already there: SWC, TALIS, LFS and ASHE.
- For everything else, download the data dictionaries to your working area. Convert them to Markdown once so the agent can read them easier.
- Ask the data owners to improve their documentation.



AGWRK_desc	string		SPSS labels for AGWRK: Whether agency worker. Available from 2011 Q1 to 2026 Q2 (latest).
AMTUPL	double		Number of weeks without pay. Available from 1999 Q2 to 2000 Q4.
ANXIETY	double		Level of anxiety or depression. Available in 2012 Q1 only.
ANXIETY_desc	string		SPSS labels for ANXIETY: Level of anxiety or depression. Available in 2012 Q1 only.

LFS column comments: definitions and availability.

Version control



Review the diff. Is this the change you intended?

If you're using AI heavily, use Git in Databricks and on your laptop.

Save Databricks notebooks as source (`.R` / `.py`) to make changes easier to review.

Ask your agent to create a branch, commit and push. With `gh`, it can also draft the pull request.

AGENTS.md and skills

If you keep repeating instructions, write them once in `AGENTS.md` at the root of your repo.

Examples: use R and `sparklyr`; save figures as PNGs with `ggsave`; commit and push regularly.

For specific tasks, add a **skill**: a folder with a `SKILL.md` explaining the steps.

On your laptop:
`~/.agents/skills/`

On Databricks:
`/Workspace/Users/<your-email>/.assistant/skills/`

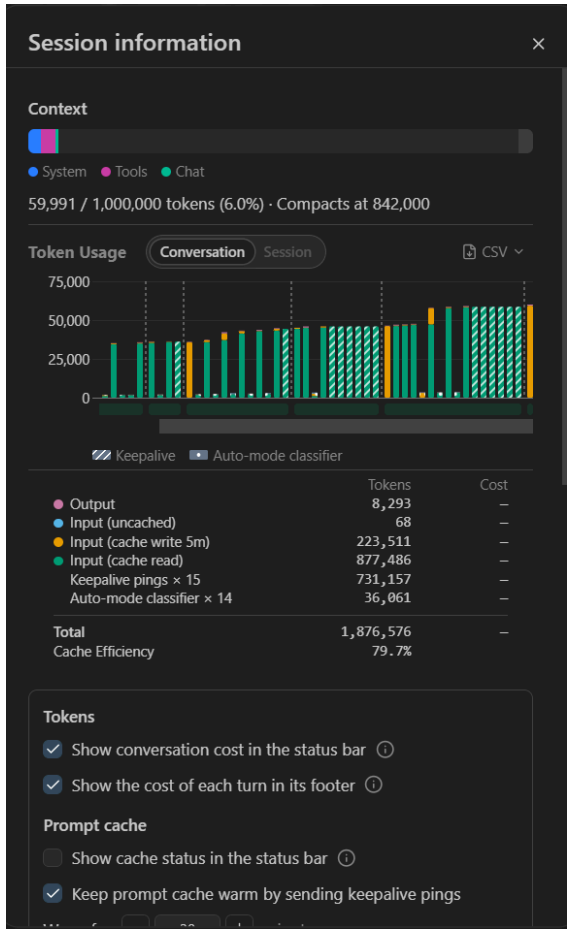
For example, put your chart-formatting instructions in a `finalise-figures` skill, then invoke it with `/finalise-figures`.

```
example-eerc-project/
├── AGENTS.md
├── analysis/
│   └── 01-clean-data.R
└── .agents/skills/
    └── finalise-figures/
        └── SKILL.md
```

AGENTS.md

```
# Project instructions
- Use R and sparklyr.
- Save figures in outputs/.
- Use ggsave() for PNGs.
- Commit and push regularly.
```

Model settings: context and Auto mode



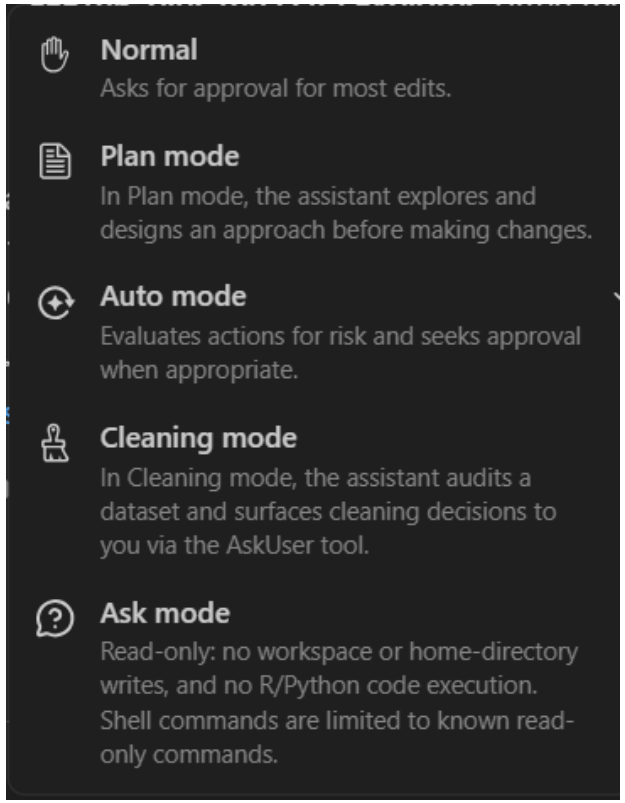
Context window

The model's working space: instructions, chat, files it reads and tool results.

Compaction summarises long chats: use this to reduce the time you spend with a filled up context window. Save decisions in files for later. Start fresh when changing tasks. Models tend to do better with less irrelevant context.

Auto mode

A classifier allows or blocks risky actions. Allows the agent to run unattended for longer, in a safe(r) way.



Choose a model for the task

01 **Haiku**

Cheap, but shouldn't be your daily driver: use for subagent workflows.

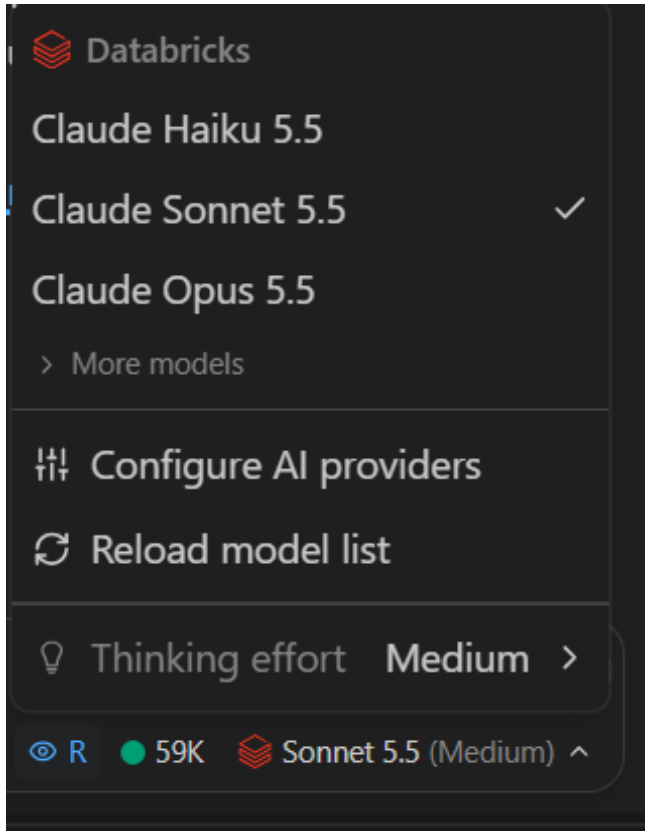
02 **Sonnet**

Default for most tasks.

03 **Opus**

For problems Sonnet is struggling with.

The best models change every few weeks, and our availability on Databricks is limited (for now). Check Arena LLM leaderboard for up-to-date comparison of LLM capability.



Where does computation happen?

R: keep heavy work in the remote `tbl()` pipeline, before `collect()`.

```
# conn: con on laptop, sc in notebook
# table_name: table
summary <- tbl(conn, I(table_name)) |>
  count(LAcode) |>
  collect()

# Only the small summary enters R.
# Plot or format it here.
```

Filter, select columns, join and summarise remotely.
Everything before `collect()` runs on the Databricks SQL Warehouse; after `collect()` the results are downloaded into R.
For regressions using Spark/SQL check out dbreg.

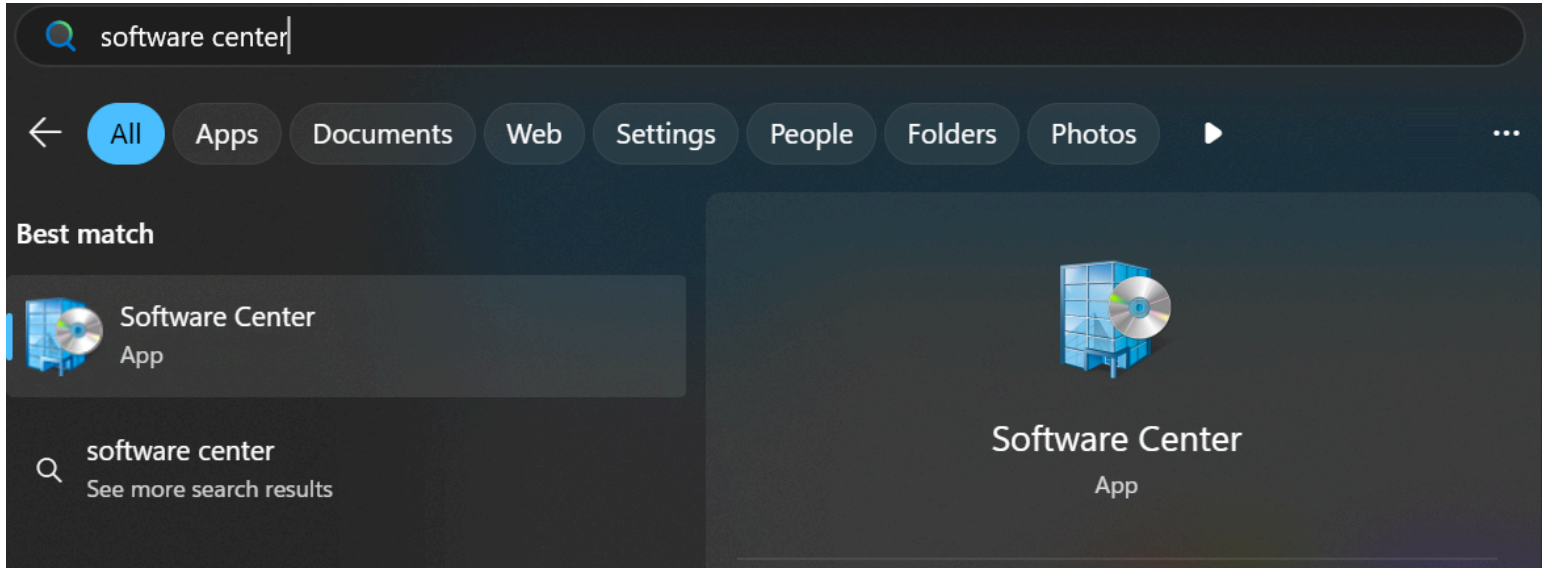
Extra slides

Start with Software Center

On your Windows computer

Start → **Software Center**.

Find **Positron** or **RStudio** and select **Install**.

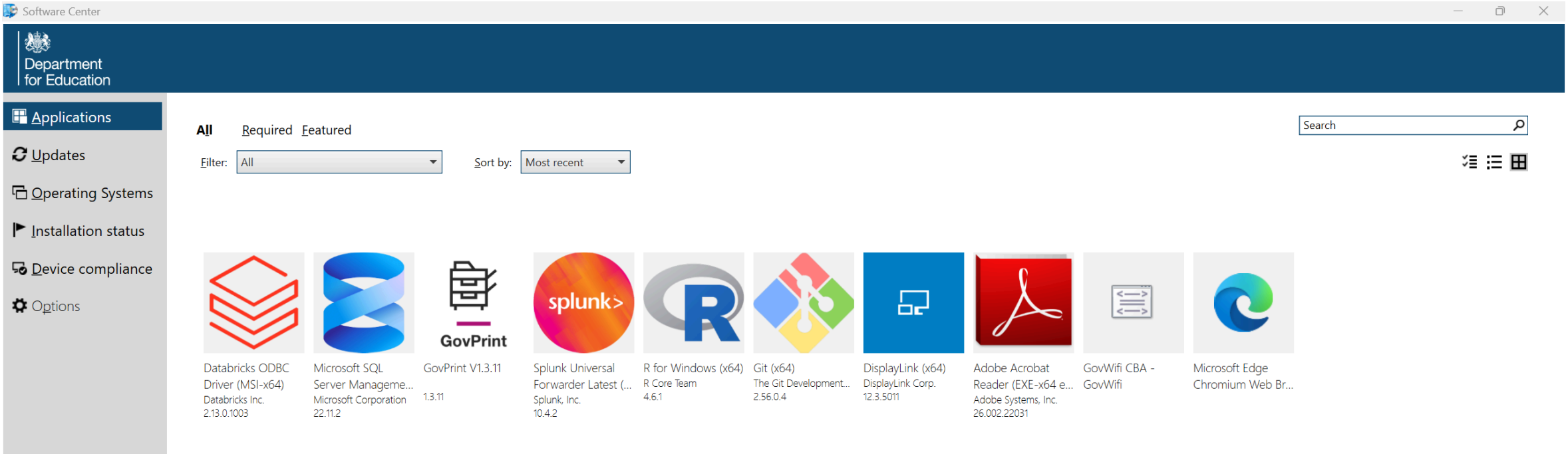


← Return to working environments

See the application list →

Find your app in Software Center

Select **Applications**, then search for your tool or select its tile.



← Return to working environments

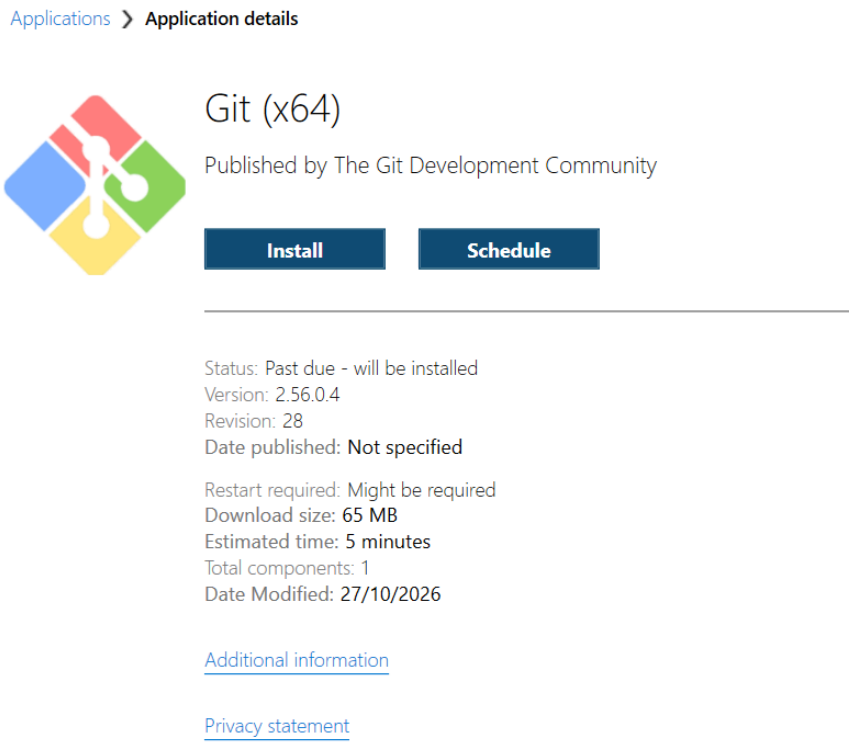
See the Install button →

Open the app and select Install

Git shown as an example

Open the application’s details, check the name, then select **Install**.

Use **Installation status** to check progress.



← Return to working environments

← Return to supporting tools

Install R and Rtools

R



R for Windows (x64)
R Core Team
4.6.1

Rtools



Rtools 45 (x64)
Cran project
4.5

Install **R** and compatible **Rtools** from Software Center. Open each app's details and select **Install**. You might have to do it twice because it's buggy.

← Return to language choice

If using Python: install with uv



In PowerShell, install uv:

```
winget install -e --id astral-sh.uv
```

Reopen PowerShell, then install Python:

```
uv python install 3.13
```

← Return to language choice

Install the supporting tools

Git



Git (x64)
The Git Development...
2.56.0.3

Track changes to code and project files.

Databricks ODBC



Databricks ODBC
Driver (MSI-x64)
Databricks Inc.
2.13.0.1003

Connect compatible tools to Databricks through ODBC.

Open each app's details in Software Center and select **Install**.

← Return to supporting tools

VS Code

Run this in Windows PowerShell:

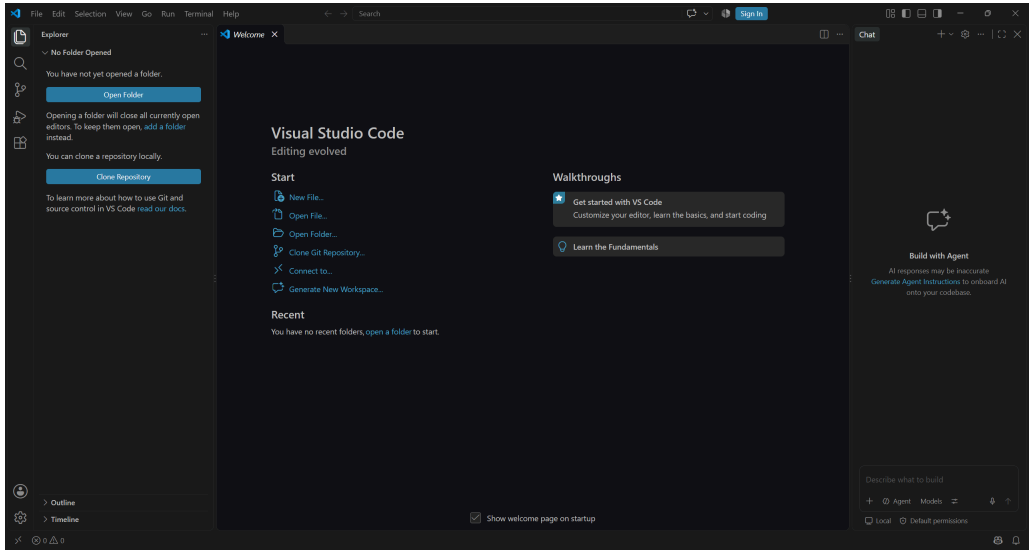
```
winget install -e --id Microsoft.VisualStudioCode
```

01 **Open Visual Studio Code**

Wait for installation, then open it from Start.

02 **Open the project folder**

Use File → Open Folder.



← Return to working environments

Claude Code + UG · 1 / 2

This setup requires **both Claude Code and Unity Gateway (UG).**

1. In PowerShell, install uv if you do not already have it:

```
winget install -e --id astral-sh.uv
```

2. Reopen PowerShell, then install both tools:

```
winget install -e --id Anthropic.ClaudeCode
uv tool install git+https://github.com/databricks/unity-gateway
```

3. Reopen PowerShell again. Check both installations:

```
claude --version
ug --help
```

← Return to working environments

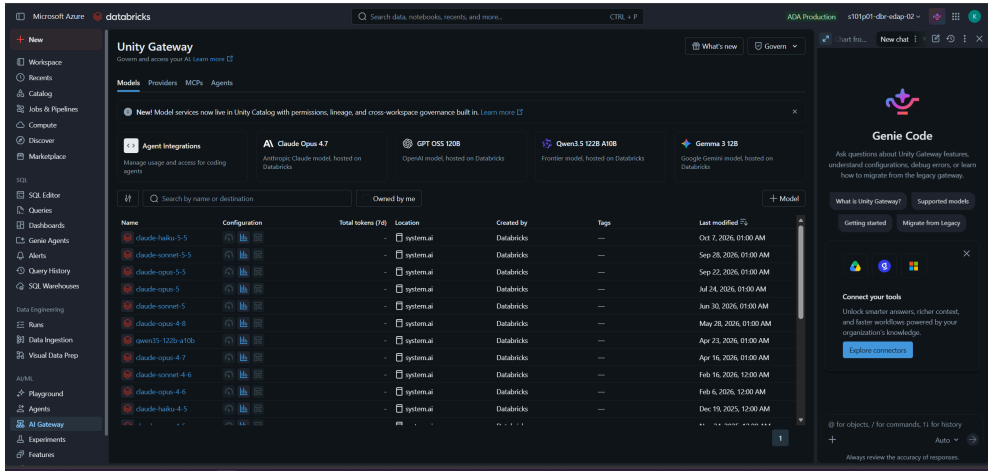
Connect and launch →

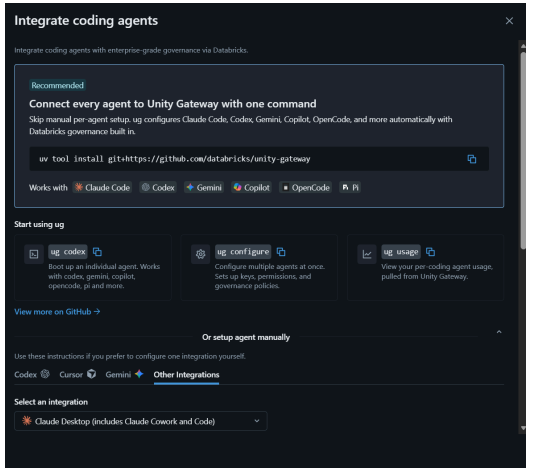
Claude Code + UG · 2 / 2

Databricks: **AI Gateway → Agent Integrations**.

With both tools installed, open PowerShell in your project folder:

```
ug claude
```





[Unity Gateway setup guide ↗]

Launch from your project folder

Claude Code
`ug claude`

Codex
`ug codex`

OpenCode
`ug opencode`

First launch: select your workspace and authenticate.

[← Install Claude Code + UG]

[← Return to working environments]