

SQLRooms

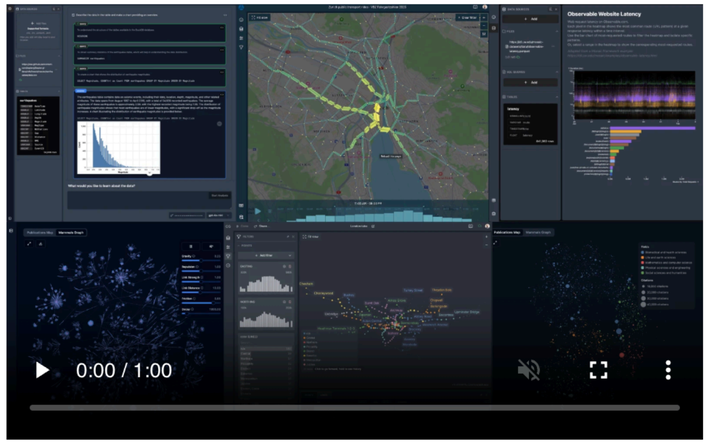
Ilya Boyandin, Foursquare

Open Visualization Summit Zurich 2026

SQLRooms

Build data-centric apps with DuckDB

An open source React toolkit for human + agent collaborative analytics apps



What is SQLRooms? | Example Apps | View on GitHub

Local Analytics, No Backend Required

Leverage DuckDB's powerful SQL capabilities, enabling fast in browser data processing without a backend

Own Your Data

Data remains on your local device for maximum privacy, sub-second analytics on large datasets, and offline functionality

Privacy-Preserving AI Integration

Use agents that can write and execute SQL queries, and generate insights without sharing your data with model providers

Modular Architecture

Pick and choose the functionality you need for composable, extensible applications, with integrations for popular data visualization libraries.

Modern UI Components

Comprehensive set of React components including data tables, layouts, and visualization tools for building beautiful analytics interfaces

Offline Use

Work with your data, run queries, and analyze results even without an internet connection. SQLRooms supports offline workflows using persistent storage via OPFS.

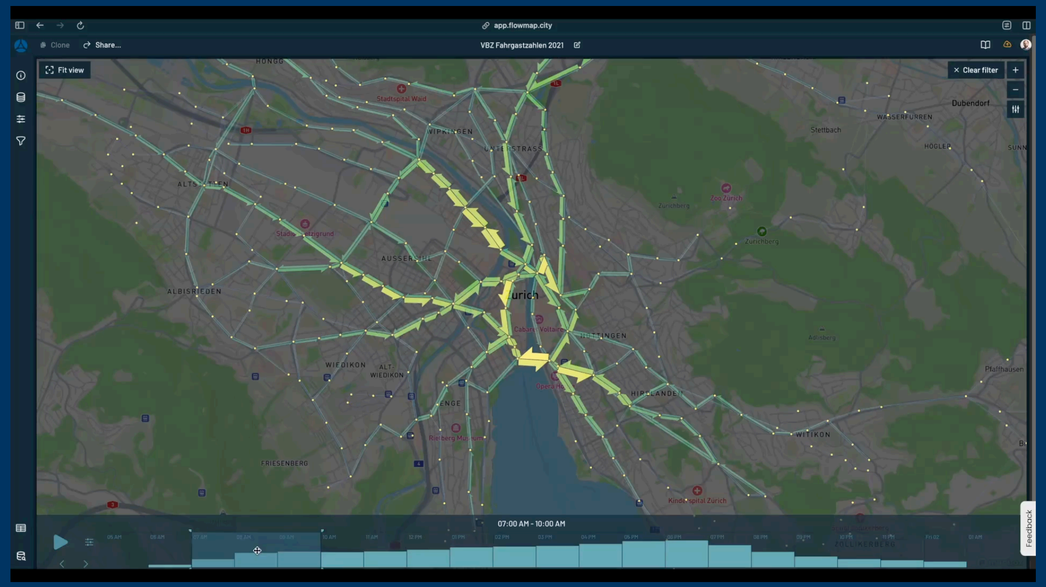
OpenJS Foundation

Supported by

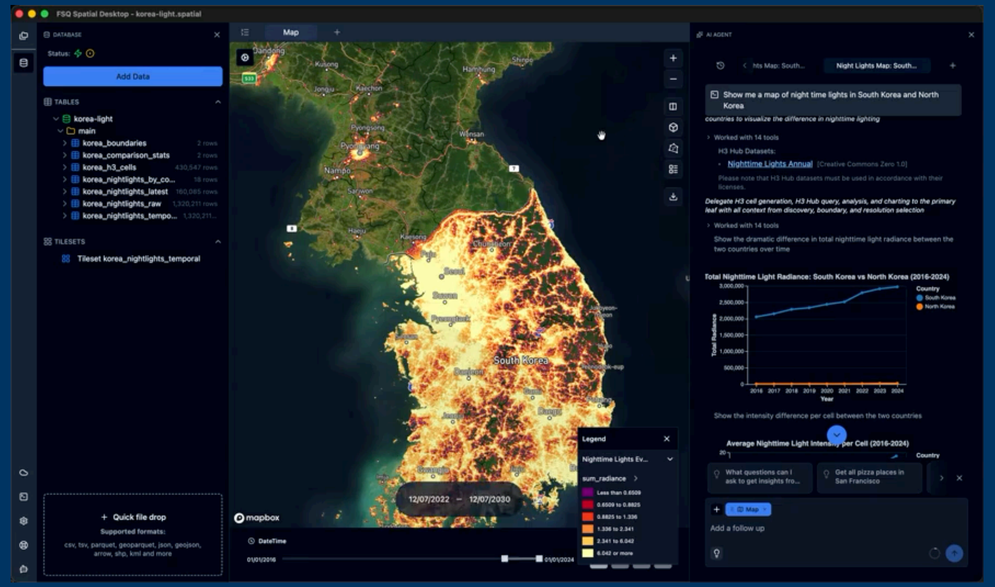
FOURSQUARE

vis.gl

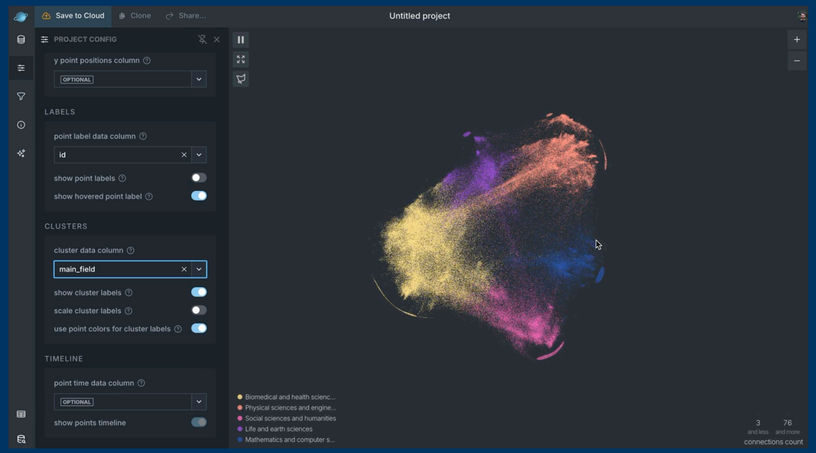
Flowmap.city



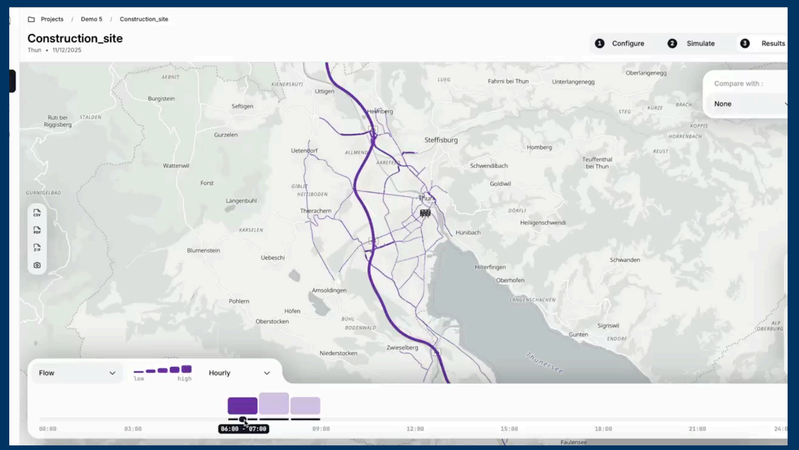
FSQ Spatial Desktop



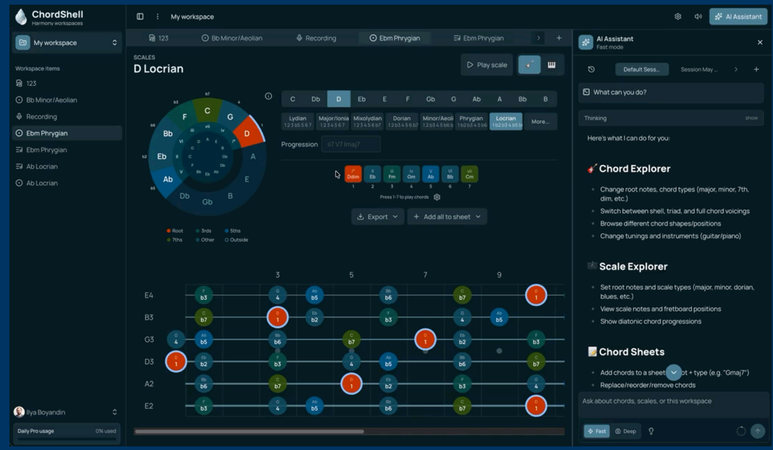
Cosmograph.app



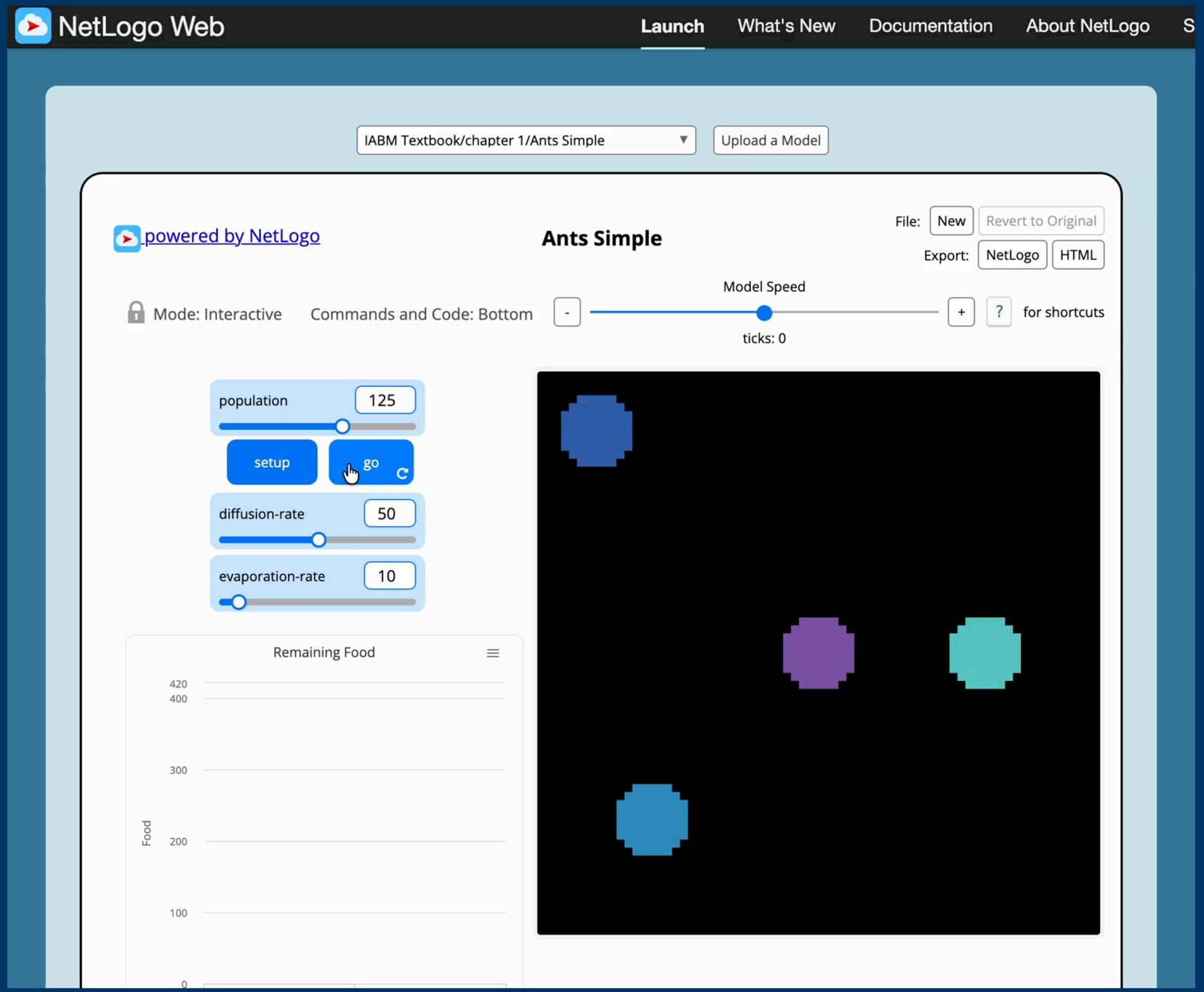
Transcality



ChordShell




SQLRooms
SQL
DuckDB
DuckDB
DuckDB
DuckDB
DuckDB
DuckDB


NetLogo Web
Launch
What's New
Documentation
About NetLogo
S
IABM Textbook/chapter 1/Ants Simple
Upload a Model
powered by NetLogo
Ants Simple
File:
New
Revert to Original
Export:
NetLogo
HTML
Mode: Interactive
Commands and Code: Bottom
Model Speed
-
+
?
for shortcuts
ticks: 0
population
125
setup
go
diffusion-rate
50
evaporation-rate
10
Remaining Food
420
400
300
200
100
0
Food

Core Packages

Package	Description
@sqlrooms/artifacts	Artifact registry, workspace tab helpers, and layout integration for dashboards, notebooks, documents, canvases, pivots, and apps.
@sqlrooms/blocks	Shared contracts for composable blocks designed for BlockDocuments and other block containers; @sqlrooms/documents provides the editor, persistence, and runtime.
@sqlrooms/db	DuckDB-centered orchestration layer for multi-database execution.
@sqlrooms/duckdb	WebAssembly build of DuckDB plus helper hooks for query execution and data import.
@sqlrooms/duckdb-core	Shared DuckDB connector contracts and core query and import utilities.
@sqlrooms/duckdb-node	Node.js DuckDB connector for server-side SQLRooms runtimes.
@sqlrooms/layout	State and rendering for panel-based UIs with resizable splits, tabs, docking, and responsive grid layouts.
@sqlrooms/room-shell	Central application shell and Zustand-based state manager with panel system and DuckDB integration.
@sqlrooms/room-store	Core state management utilities, RoomStore, and React context providers.
@sqlrooms/ui	Tailwind-powered component library and theme manager used across SQLRooms packages.

Utility Packages

Package	Description
@sqlrooms/utils	Shared helper functions for colors, formatting, random IDs, and string utilities.

AI Packages

Package	Description
@sqlrooms/ai	Natural-language querying and AI-assisted analytics tools.
@sqlrooms/ai-core	Lower-level AI slice, chat UI primitives, and tool-streaming utilities.
@sqlrooms/ai-settings	AI provider and model settings state and UI components.

Feature Packages

Package	Description
@sqlrooms/codemirror	CodeMirror 6 editor components with SQLRooms theme integration and JSON and JavaScript editing helpers.
@sqlrooms/color-scales	Color-scale config, validation, legend models, and React legend components.
@sqlrooms/cosmos	High-performance WebGL graph visualization with Cosmos.
@sqlrooms/data-table	Interactive data grid for SQL results with sorting and pagination.
@sqlrooms/deck	Deck.gl map integration with JSON-driven map specs and DuckDB and Arrow dataset binding.
@sqlrooms/documents	Markdown document artifacts, structured BlockDocument and Document surfaces, rich editors, document assets, and document CRDT sync.
@sqlrooms/dropzone	Drag-and-drop file uploads with type validation and progress tracking.
@sqlrooms/kepler	Kepler.gl integration for map-first SQLRooms analytics experiences.
@sqlrooms/monaco-editor	VS Code's Monaco editor with SQL-aware autocompletion.
@sqlrooms/mosaic	Declarative charting powered by UW IDL's Mosaic library.
@sqlrooms/motherduck	MotherDuck connector using the WebAssembly client.
@sqlrooms/recharts	Responsive charts built with Recharts.
@sqlrooms/schema-tree	Interactive database-schema explorer.
@sqlrooms/sql-editor	SQL editor with history, syntax highlighting, and result docking.
@sqlrooms/vega	Vega-Lite visualization components for sophisticated interactive charts.

Experimental Packages

Package	Description
@sqlrooms/ai-rag	Retrieval-augmented generation slice for semantic search over DuckDB-backed embeddings.
@sqlrooms/app-runtime	Sandboxed iframe runtime bridge for generated HTML apps and their SQLRooms hosts.
@sqlrooms/canvas	React Flow-based artifact canvas for SQL, Vega, and dataflow node DAGs.
@sqlrooms/cells	Shared cells model and UI primitives used by notebook and canvas views.
@sqlrooms/crdt	Loro-backed CRDT slice, persistence, and sync helpers.
@sqlrooms/db-settings	Database connection settings state and UI components.
@sqlrooms/discuss	Threaded discussion system with anchor links to data points.
@sqlrooms/mcp	Transport-neutral MCP capability runtime with schema validation, invocation policies, and a browser bridge protocol.
@sqlrooms/notebook	Artifact-scoped notebook UI and state slice for SQL, text, Markdown, Vega, and input cells.
@sqlrooms/pivot	Slice-driven pivot table UI backed by DuckDB SQL and Vega-Lite charts.
@sqlrooms/python	Python runtime, Pyodide adapter, commands, and block primitives for SQLRooms block documents.
@sqlrooms/s3-browser	S3-compatible storage browser with uploads and directory management.
@sqlrooms/webcontainer	WebContainer state slice and runtime helpers for app-building workflows.

What remains valuable in an agent world?

1. Deep engines. DuckDB, Postgres, Chromium, React, deck.gl, Arrow, etc. Agents consume them rather than regenerate them.

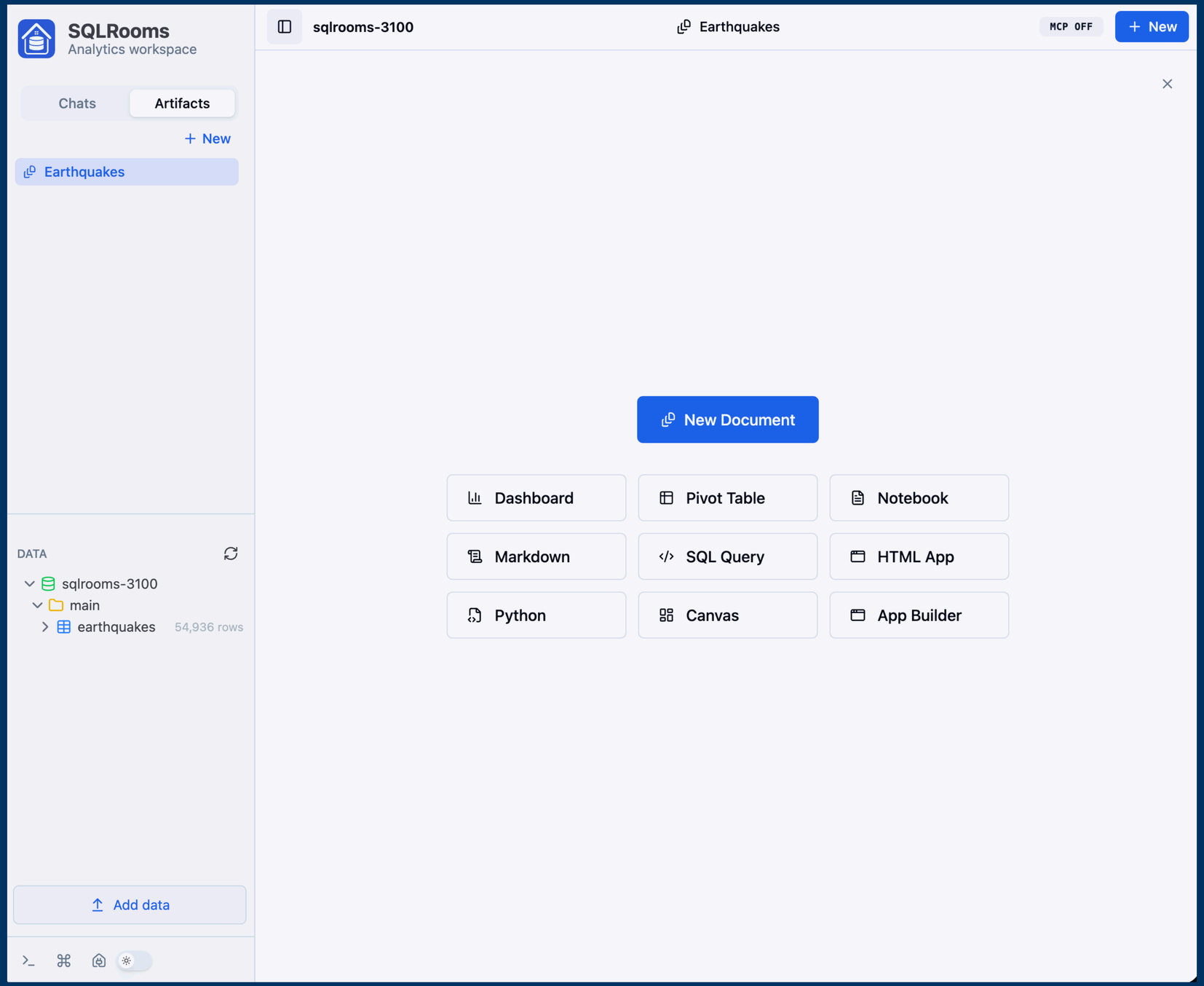
2. Ecosystems and standards. A Vega-Lite specification is useful because thousands of systems understand what it means. An arbitrary JSON visualization specification Claude invented five minutes ago isn't.

3. Persistent shared semantics. If SQLRooms can establish what a "dataset", "selection", "map layer", “document", "dashboard", etc. means, different humans and agents can operate on the same thing.

4. Distribution/trust. "Open this SQLRooms project" can eventually mean something predictable. Random generated application #473 has no such expectation.

```
uv tool install sqlrooms
```

```
sqlrooms ./my-project.db
```


SQLRooms
Analytics workspace
Chats
Artifacts
+ New
Earthquakes
DATA
sqlrooms-3100
main
earthquakes
54,936 rows
Add data
sqlrooms-3100
Earthquakes
MCP OFF
+ New
New Document
Dashboard
Pivot Table
Notebook
Markdown
SQL Query
HTML App
Python
Canvas
App Builder

SQLRooms Framework

* **lightweight** enough for local exploration
* **composable** enough for building custom apps
* **structured** enough for agents
* **inspectable** enough for humans

Modular

```
import {createAiSlice} from "@sqlrooms/ai";
import {createArtifactsSlice} from "@sqlrooms/artifacts";
import {createDeckMapsSlice} from "@sqlrooms/deck";
import {createRoomStore} from "@sqlrooms/room-shell";
import {createSqlEditorSlice} from "@sqlrooms/sql-editor";

export const {roomStore, useRoomStore} = createRoomStore<RoomState>(
  (set, get, store) => ({
    ...createRoomShellSlice()(set, get, store),
    ...createSqlEditorSlice()(set, get, store),
    ...createArtifactsSlice()(set, get, store),
    ...createAiSlice()(set, get, store),
    ...createDeckMapsSlice()(set, get, store),
 })
);
```

Composable

```
import {RoomShell} from "@sqlrooms/room-shell";
import {roomStore} from "./store";

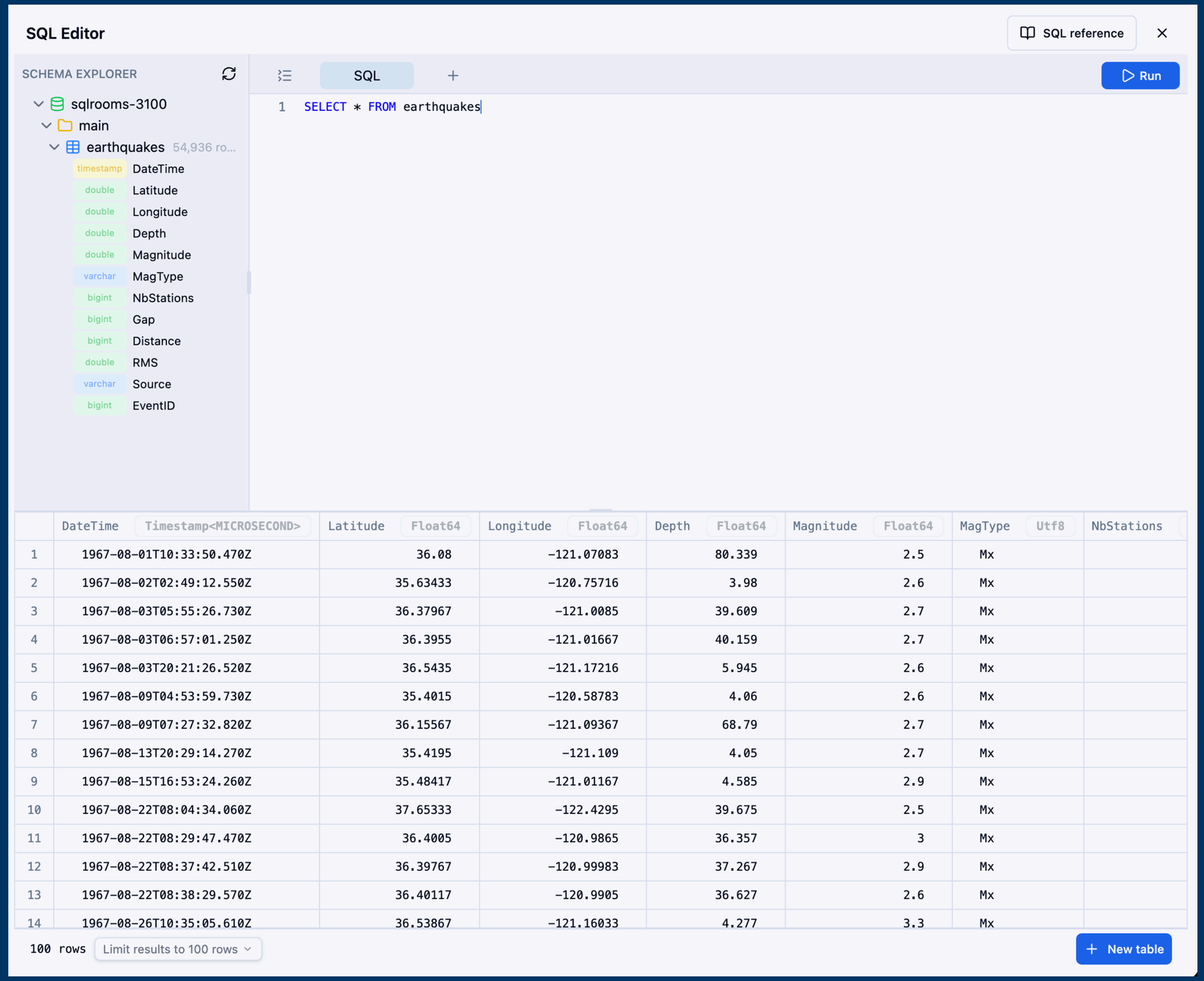
export const Room = () => (
 <RoomShell className="h-screen" roomStore={roomStore}>
    <RoomShell.Sidebar />
    <RoomShell.LayoutComposer />
    <RoomShell.LoadingProgress />
    <RoomShell.CommandPalette />
  </RoomShell>
);
```

DuckDB integration

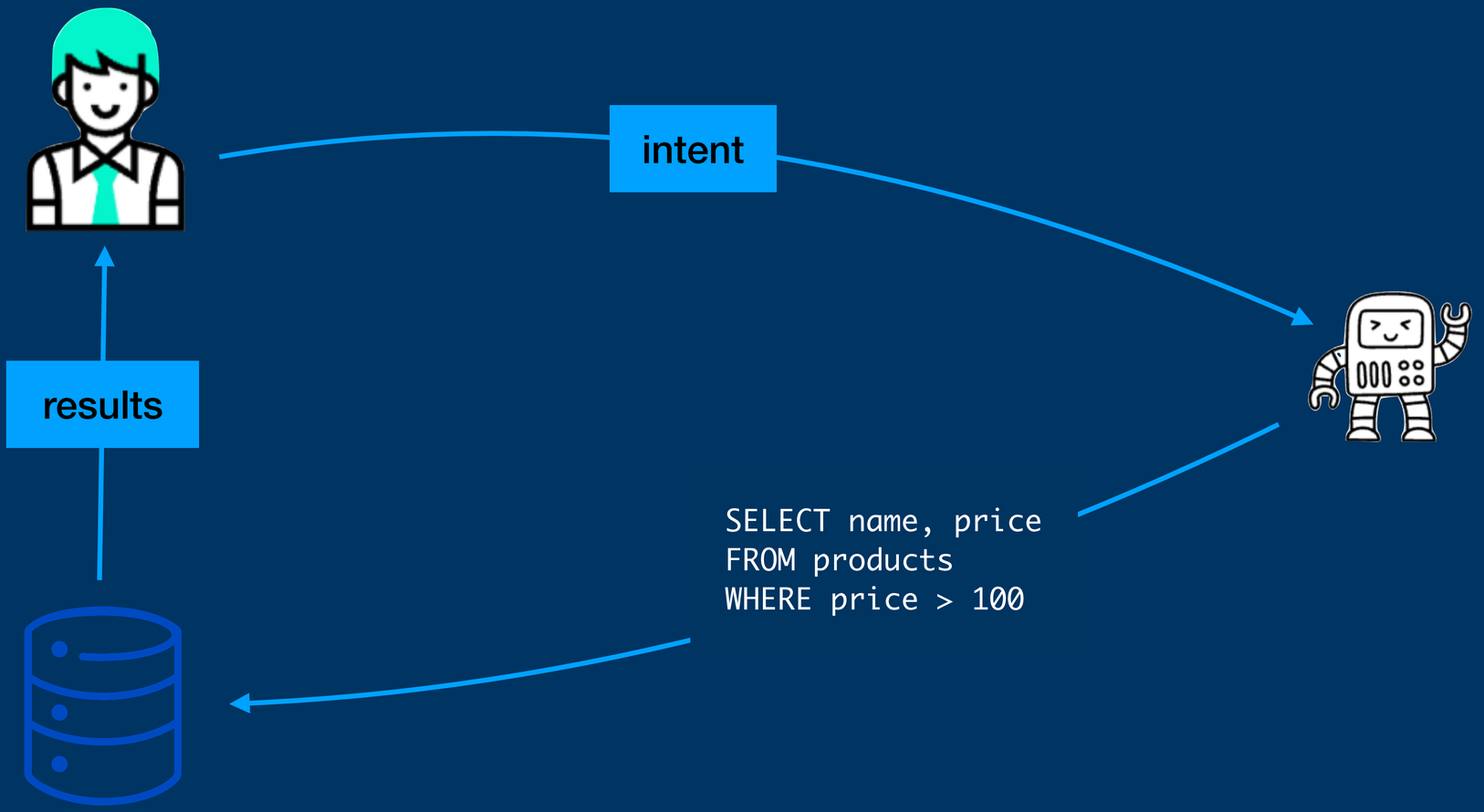
Connectors

Catalog

Query management



AI harness


intent
results
SELECT name, price
FROM products
WHERE price > 100

FSQ Spatial Desktop - Italian-Winery.spatial

DATABASE

Status:

Add Data

TABLES

- Italian-Winery
  - main
    - italy_climate 389,126 rows
      - bigint cellId
      - varchar hexId
      - double mean_annual_temp_baseline
      - double mean_annual_precip_baseli...
    - italy_elevation 389,126 rows
    - italy_h3_cells 389,126 rows
    - italy_landcover 389,126 rows
    - italy_regions 20 rows
    - italy_wine_by_region 20 rows
    - italy_wine_suitability 389,126 ro...
      - bigint cellId
      - varchar hexId
      - double temp_celsius
      - double precip_mm
      - double elevation_m
      - double land_cover
      - double temp_score
      - double precip_score
      - double elevation_score
      - double landcover_score
      - double composite_score

Quick file drop

Supported formats:

csv, tsv, parquet, geoparquet, json, geojson, arrow, shp, kml and more

Map



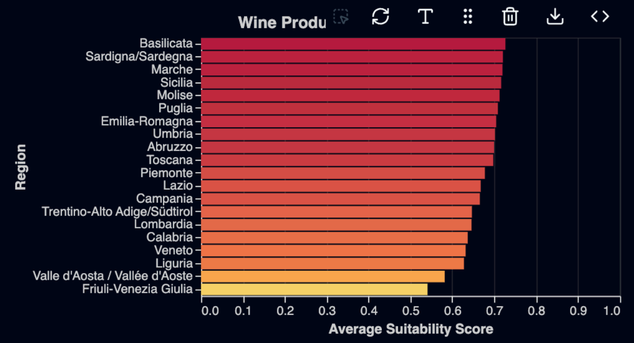
AI AGENT

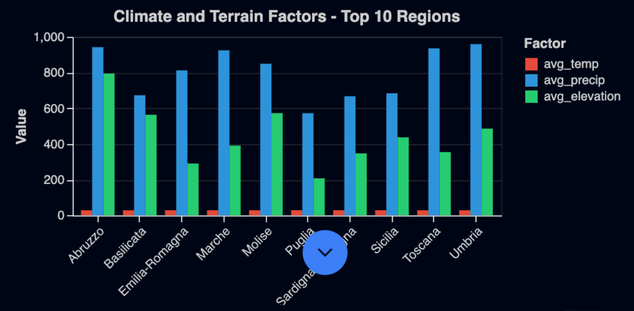
st of Counties in Nevada | Best Climate and Terrai...

Which areas in Italy have best climate and terrain for winery?

Delegate the H3 cell generation, H3 Hub querying, and analysis to the primary h3hub-analysis leaf skill, passing all context from discovery and boundary steps

Worked with 36 tools





What questions can I ask to get insights fro...

Get all pizza places in San Francisco

Map

Add a follow up

FSQ Spatial Desktop - korea-light.spatial

DATABASE

Status:

Add Data

TABLES

- korea-light
  - main
    - korea_boundaries 2 rows
    - korea_comparison_stats 2 rows
    - korea_h3_cells 430,547 rows
    - korea_nightlights_by_co... 18 rows
    - korea_nightlights_latest 160,085 rows
    - korea_nightlights_raw 1,320,211 rows
    - korea_nightlights_tempo... 1,320,211...

TILESETS

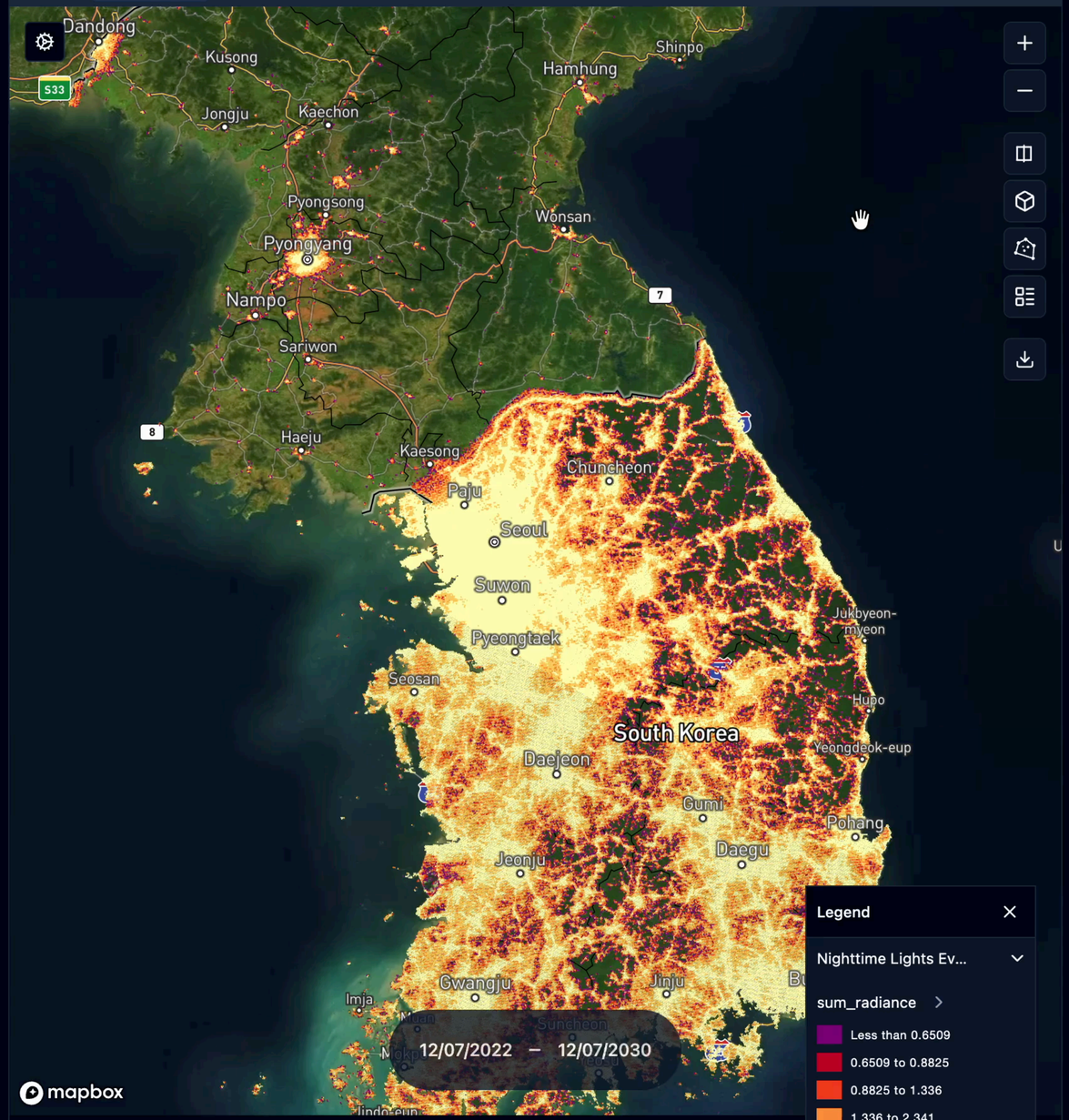
Tileset korea_nightlights_temporal

+ Quick file drop

Supported formats:

csv, tsv, parquet, geoparquet, json, geojson, arrow, shp, kml and more

Map



Legend

Nighttime Lights Ev...

sum_radiance

- Less than 0.6509
- 0.6509 to 0.8825
- 0.8825 to 1.336
- 1.336 to 2.341
- 2.341 to 6.042
- 6.042 or more

DateTime

01/01/2016 — 01/01/2024

AI AGENT

hts Map: South... | Night Lights Map: South...

Show me a map of night time lights in South Korea and North Korea

countries to visualize the difference in nighttime lighting

Worked with 14 tools

H3 Hub Datasets:

- Nighttime Lights Annual [Creative Commons Zero 1.0]

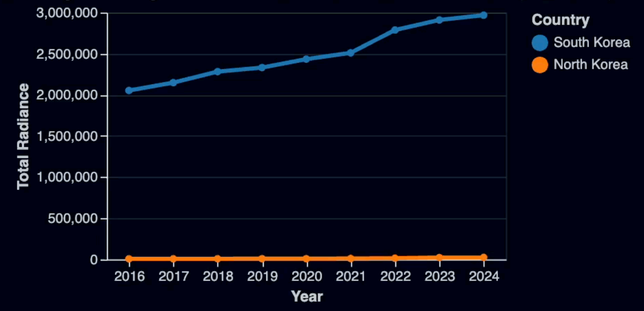
Please note that H3 Hub datasets must be used in accordance with their licenses.

Delegate H3 cell generation, H3 Hub query, analysis, and charting to the primary leaf with all context from discovery, boundary, and resolution selection

Worked with 14 tools

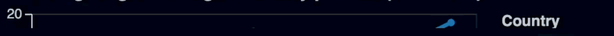
Show the dramatic difference in total nighttime light radiance between the two countries over time

Total Nighttime Light Radiance: South Korea vs North Korea (2016-2024)



Show the intensity difference per cell between the two countries

Average Nighttime Light Intensity per Cell (2016-2024)



What questions can I ask to get insights fro...

Get all pizza places in San Francisco

Map

Add a follow up

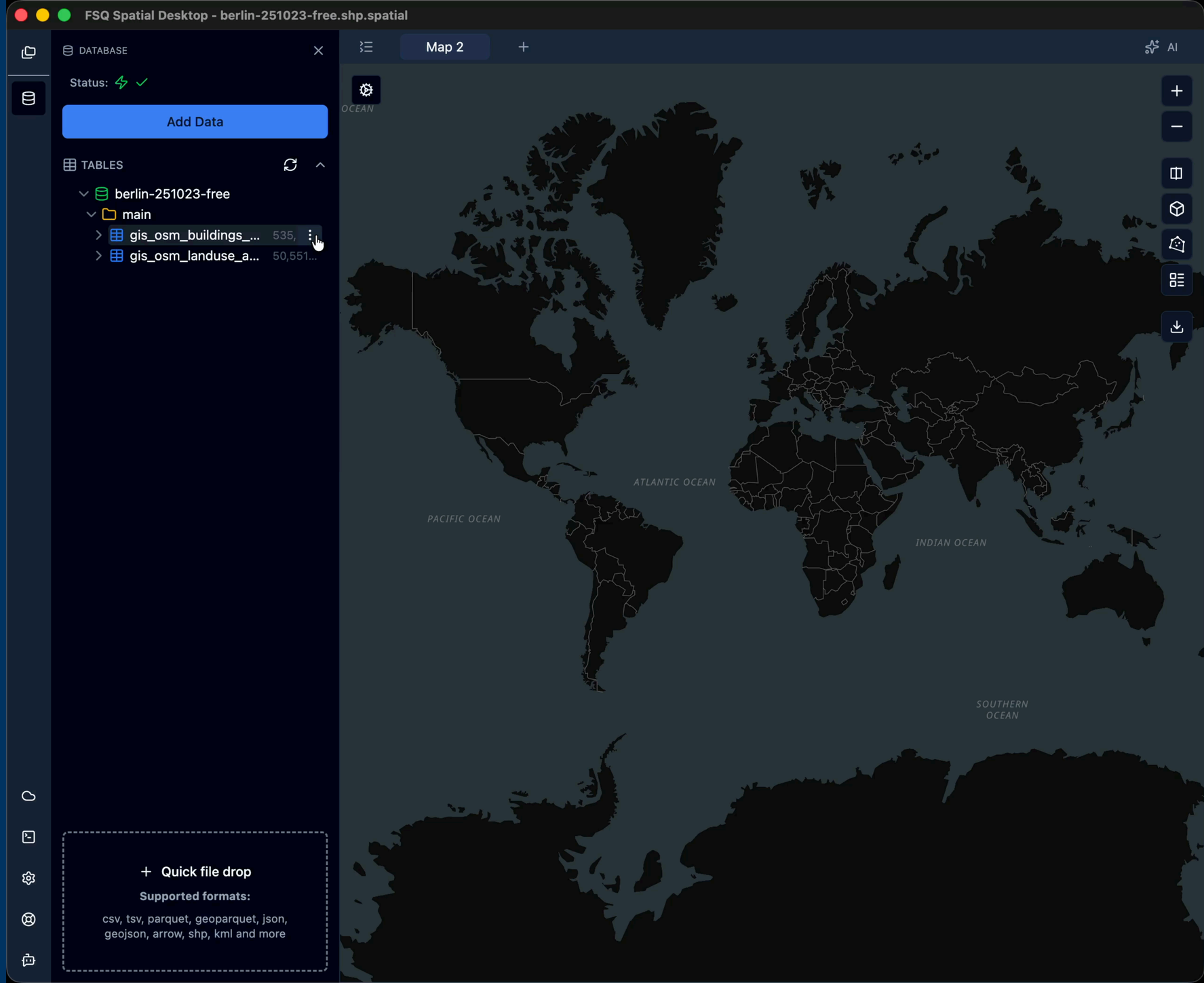
Dynamic tiling

duckdb spatial

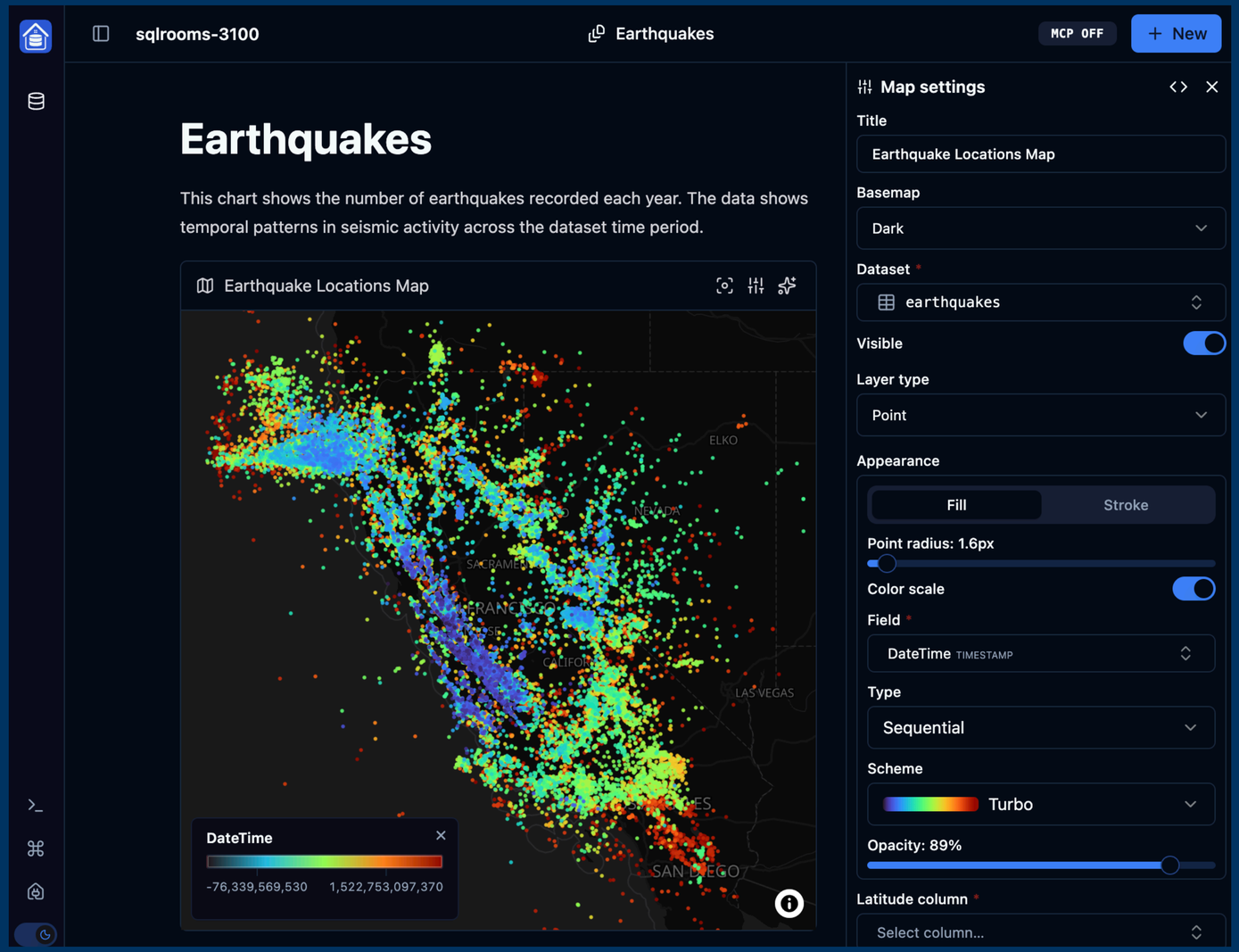
ST_AsMVTGeom()

deck.gl

MVTLayer



@sqlrooms/deck




```tsx
import {DeckJsonMap} from '@sqlrooms/deck';

const spec = {
  initialViewState: {
    longitude: -122.4,
    latitude: 37.74,
    zoom: 10,
    pitch: 0,
    bearing: 0,
  },
  controller: true,
  layers: [
    {
      '@@type': 'GeoArrowScatterplotLayer',
      id: 'airports',
      _sqlroomsBinding: {
        dataset: 'airports',
        geometryColumn: 'geom',
      },
      getFillColor: {
        '@@function': 'colorScale',
        field: 'scalerank',
        type: 'sequential',
        scheme: 'YlOrRd',
        domain: 'auto',
      },
      getRadius: 6,
      radiusUnits: 'pixels',
      radiusMinPixels: 2,
    },
  ],
};


...

  return (
    <DeckJsonMap
      spec={spec}
      datasets={{
        airports: {
          sqlQuery:
            'SELECT name, abbrev, scalerank, ST_AsWKB(geom) AS geom FR
          geometryColumn: 'geom',
          geometryEncodingHint: 'wkb',
        },
      }}
    />
  );

...
```

Create Map with Polygon Layer

sqlrooms-3100

Buildings in Melbourne

MCP OFF

New

Create a map with a POlygon layer

I'll create a map with a polygon layer for you. Since you have the `buildings_australia` table which contains geometry data, I'll use that to create a polygon layer showing the buildings.

The user wants to create a map with a polygon layer in the current worksheet. The buildings_australia table contains geometry data that can be used for polygon visualization. The block_document_agent will handle creating the map block with the polygon layer.

Worked with 3 tools

Perfect! I've successfully created a map with a polygon layer in your worksheet. The map displays:

Polygon Layer Features:

- **597 building polygons** from the `buildings_australia` table
- **Color-coded by height** using a Yellow-Orange-Red color scheme:
  - Lighter yellow = shorter buildings
  - Darker orange/red = taller buildings
- **Semi-transparent polygons** (80% opacity) with dark gray outlines for clarity
- **Interactive**: You can click or hover on buildings to view their attributes
- **Auto-fitted view** to show all buildings in the dataset

The polygon layer uses the `geom` geometry column from the buildings data and provides a clear visualization of the building footprints with height-based coloring for easy interpretation. You can interact with the map to explore the individual building polygons!

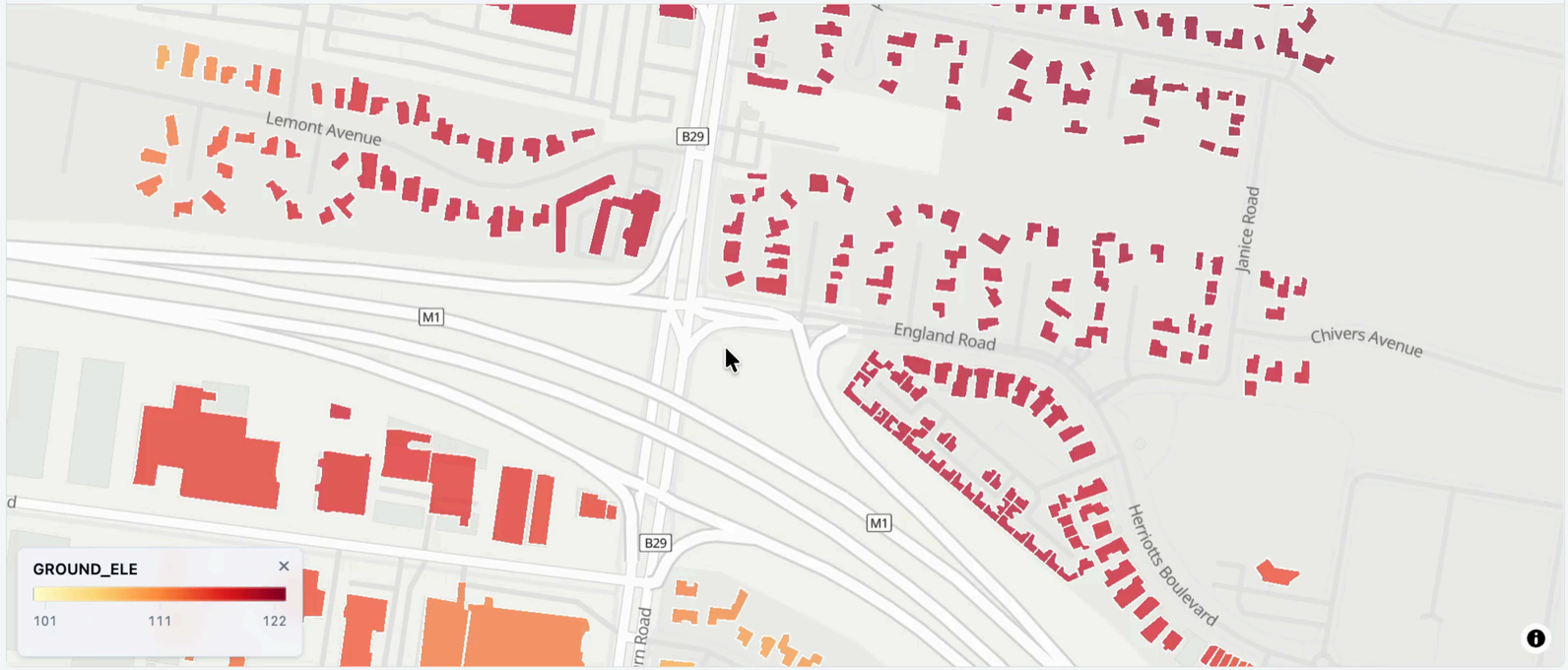
Add context

What would you like to learn about the data?

claude-sonnet-4.5

Buildings in Melbourne

Australia Buildings Map



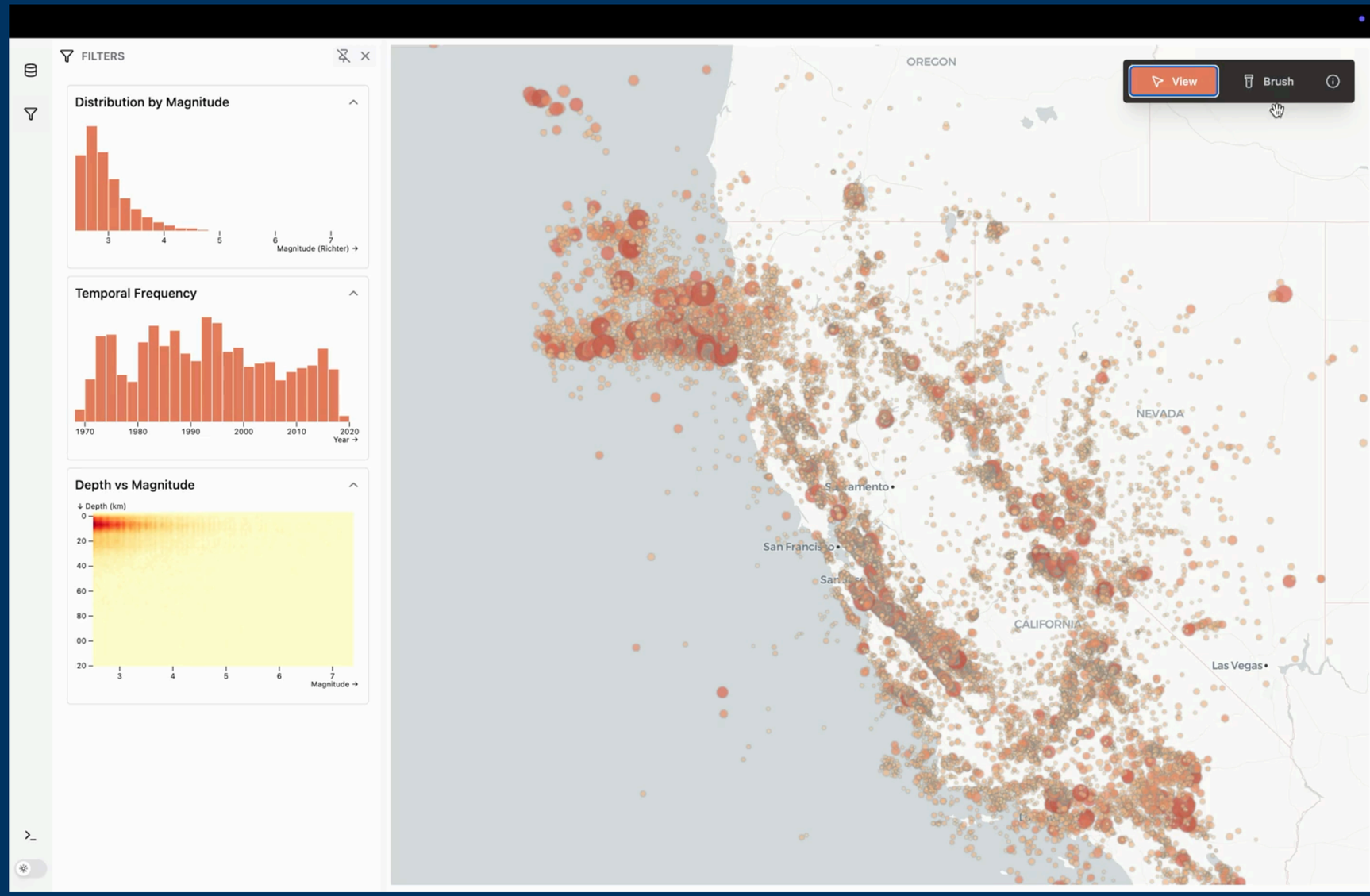
Press '/' to change block type

@sqlrooms/mosaic

UWData Mosaic

Powerful cross filtering

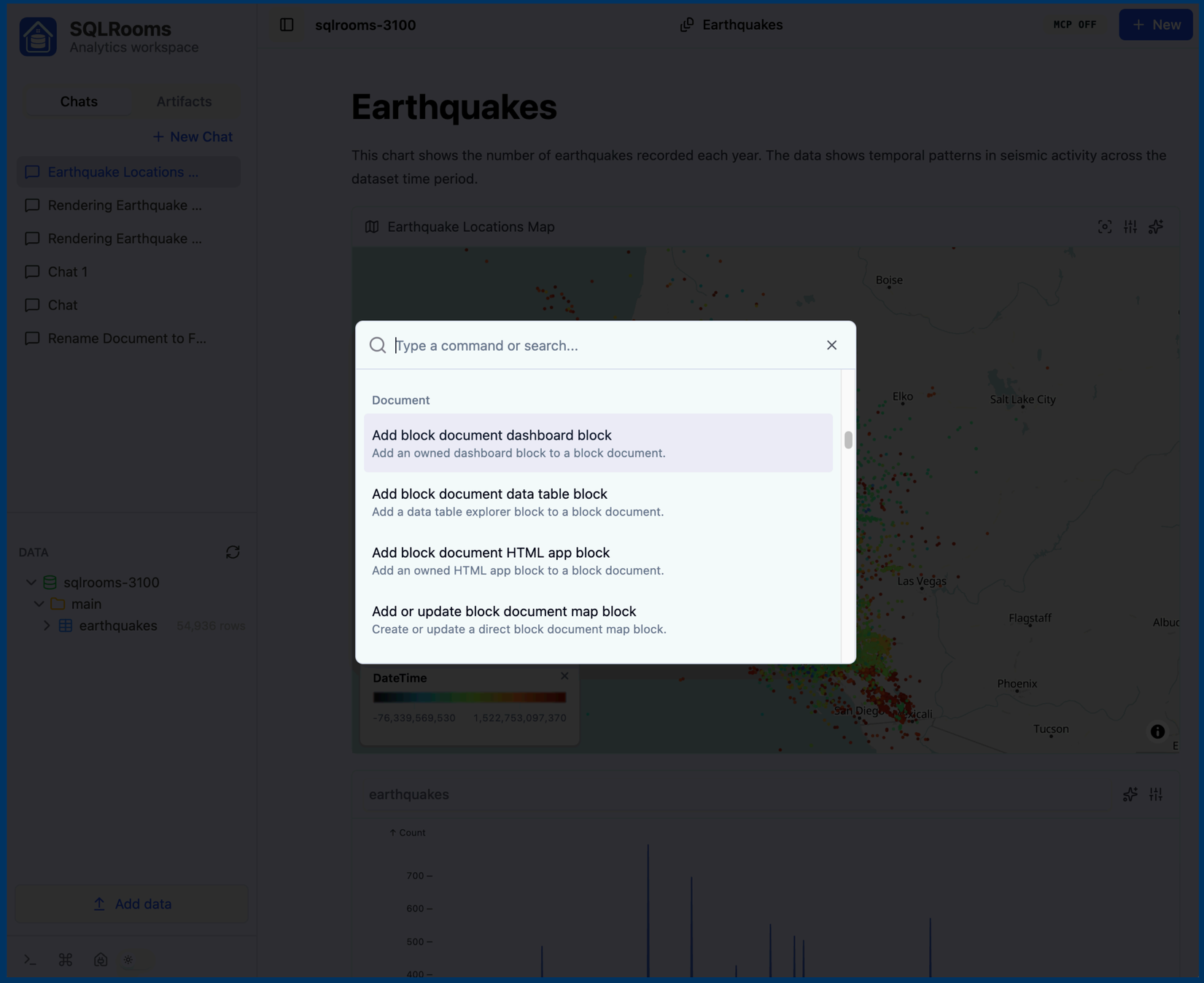
vgplot charts



Commands

Shared vocabulary for discovering and invoking actions

- command palette
- UI controls
- AI tools
- CLI
- MCP



READ-ONLY ARCHITECTURE EXPLORER

FSQ Spatial AI topology

Source analysis · not a recorded execution · 144 components

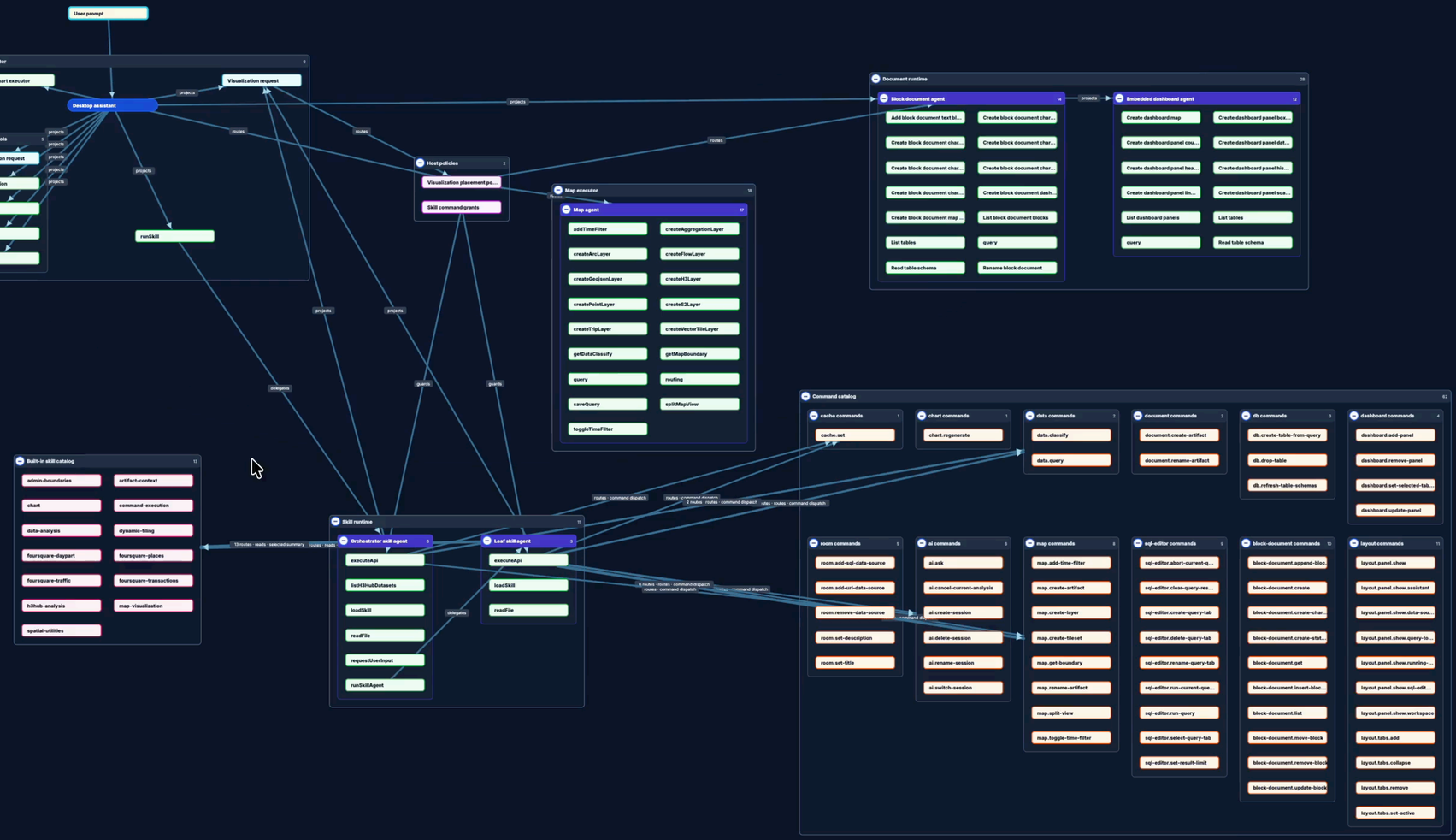
fsq-spatial-desktop `3fad7d457d76`
2026-09-07 12:00:00.000 UTC

Inspect an agent... | ▸ Find, filter & legend | ▾ Inspect selection | Paste trace | Commands

Fit view | Reset layout | Expand all regions | Collapse all regions

Drag nodes or group headers · Scroll to zoom

Colors: Entry point · Assistant · Agent · Skill runtime · Tool · Skill instructions · Gateway · Policy · Command · State · External system · Selected · Trace step



144 components · 30 routes · 25 regions

ENTRY-POINT

User prompt

A user submits a chat prompt to start or continue the Desktop assistant conversation.

Visibility	host-only
Stable ID	`user-prompt`
Group path	Standalone

TOOLS, DELEGATES AND EFFECTS

Source relationships · includes collapsed and filtered components

Desktop assistant assistant

invokes · submit prompt

Top-level SQLRooms model boundary.

CALLERS AND REGISTRATION

Source relationships · includes collapsed and filtered components

No relationships recorded in this snapshot.

SUBMISSION CONTRACT

Inputs: User-authored prompt text, optional attachments, and the active chat session.

Returns: A submitted message starts the session assistant run with captured run context.

SOURCE EVIDENCE

packages/electron-app/src/renderer/src/components/panels/assistant/EmptyChatState.tsx:75–87 — Copy

IMPLEMENTATION

`Chat.Composer`

The initial chat screen renders the SQLRooms composer for the user prompt.

packages/sqlrooms/packages/ai-core/src/components/composer/ChatComposerContext.tsx:171–196 — Copy

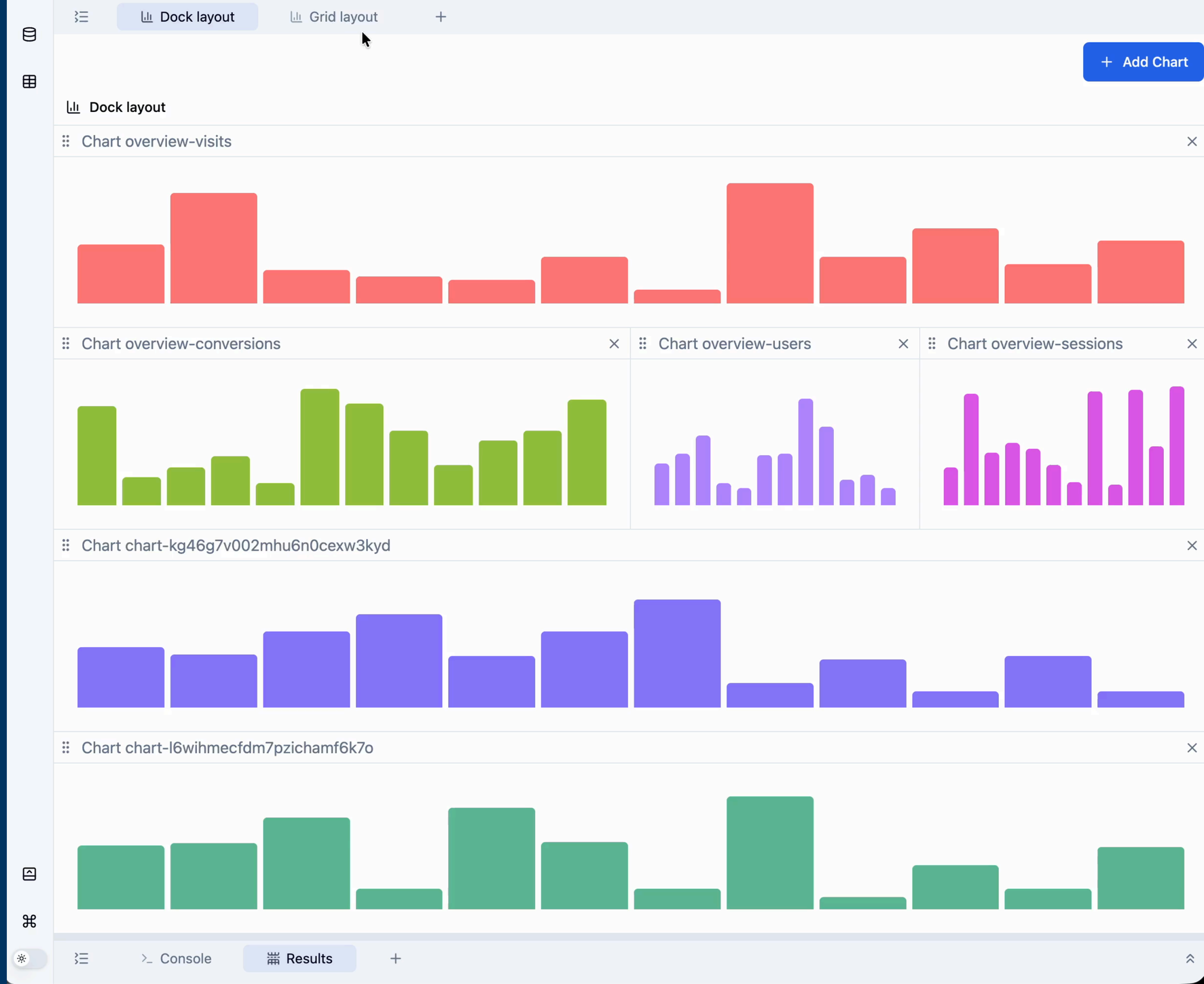
IMPLEMENTATION

`rawSend`

Submitting the composer starts analysis for an existing session, or creates a session and starts analysis when ready.

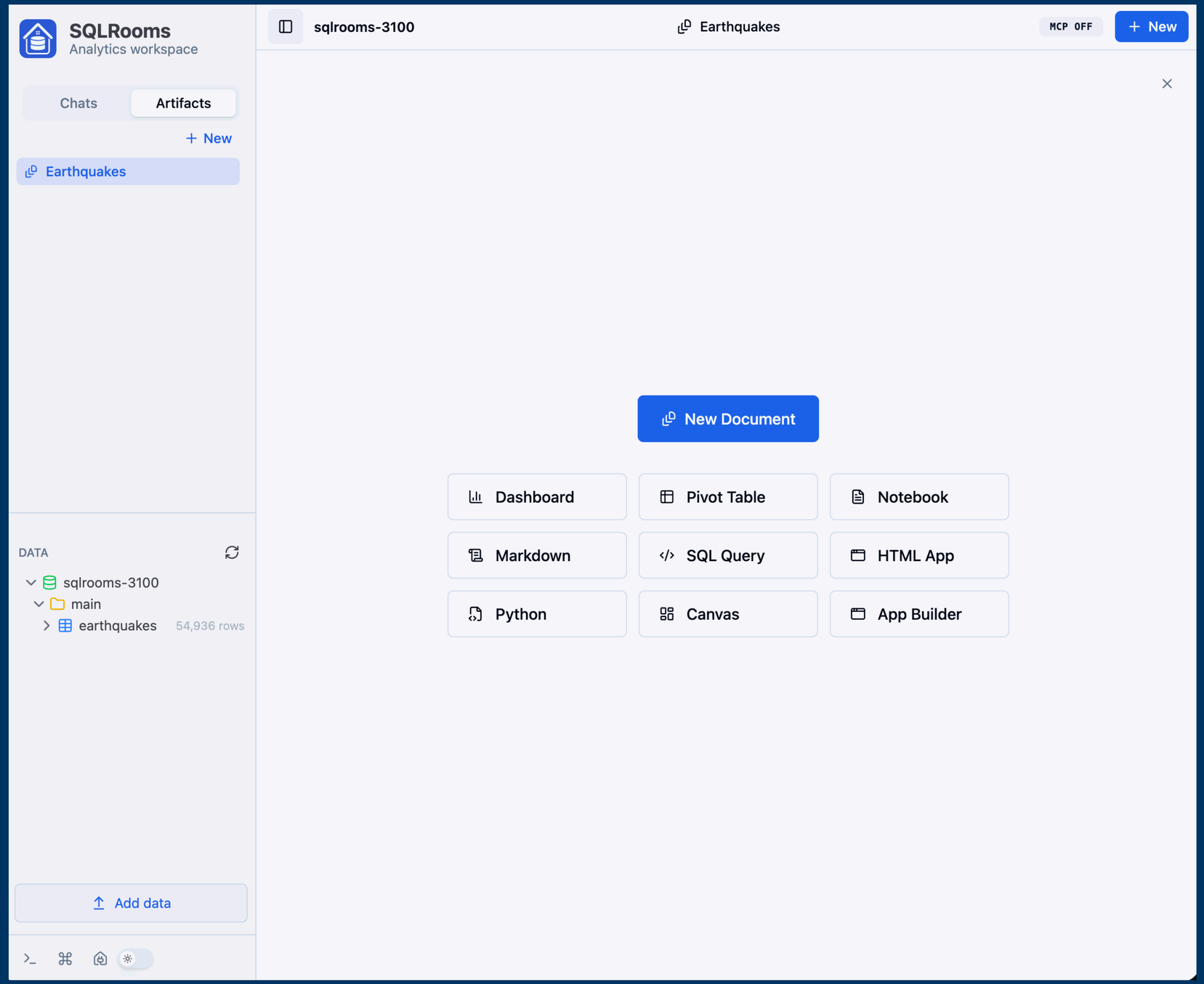
@sqlrooms/layout

- Tabs
- Resizable split panels
- Dock
- Grid



@sqlrooms/artifacts

Durable, top-level workspace entries

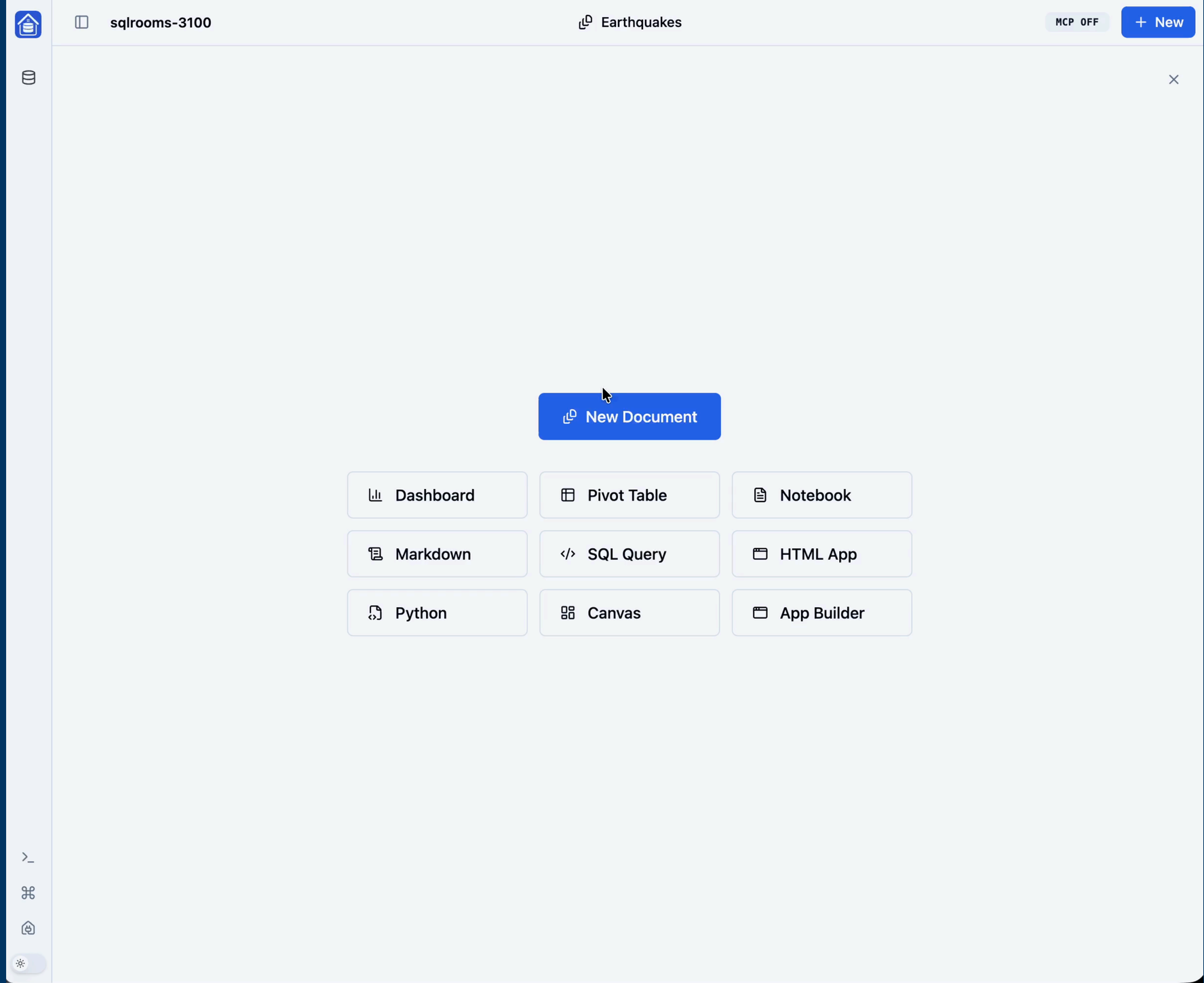


@sqlrooms/documents

Notion-like

Customizable blocks

AI and human editable



SQLRooms
Analytics workspace

Add data

DATA

- cars
  - main
    - cars 406 rows

WORKSPACE + New

- App: Playful car showro...
- App: MPG vs Horsepow...
- Cars insights
- Python
- Cars with Python chart

cars / Cars insights

+ New

AI ASSISTANT

Car Dataset Analysis Insights

analyse and describe interesting insights about cars dataset

I'll analyze the cars dataset to discover interesting insights. Let me start by exploring the data structure and then perform various analyses.

> Worked with 9 tools

Now let me create a comprehensive analysis with visualizations in the worksheet:

The user asked to analyze the cars dataset and describe interesting insights. Since the primary artifact is a worksheet, I should use the worksheet agent to create comprehensive visualizations and analysis blocks that explore fuel efficiency trends, origin comparisons, correlations, and temporal patterns.

> Worked with 13 tools

Perfect! I've created a comprehensive analysis of the cars dataset in your worksheet. Here are the **key insights** I discovered:

🚗 Major Findings:

1. Fuel Efficiency Evolution (1970s-1980s)

- Fuel efficiency impro dramatically over the model years in the d et
- Average MPG increased from the early 70s to

Add context

What would you like to learn about the data?

claude-sonnet-4.5

Cars insights

Cars Dataset Analysis - Key Insights

Fuel efficiency improved significantly over time, with MPG increasing from the 1970s to the 1980s. This trend reflects both technological advancements and regulatory pressures during the oil crisis era.

Fuel Efficiency by Origin

cars

Chart settings

Chart type *

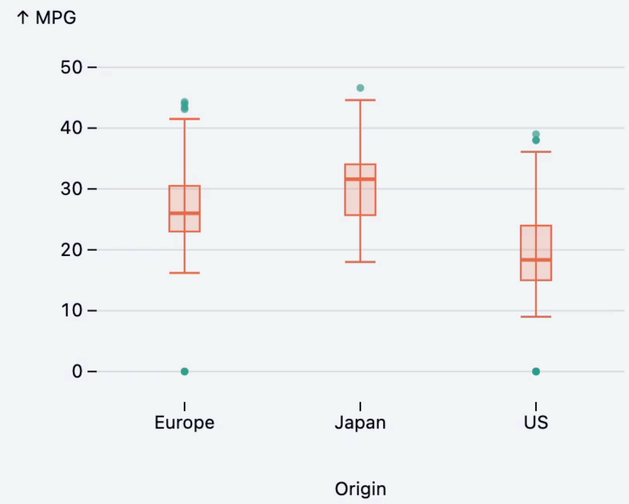
Box Plot

X Field (categorical) *

Origin VARCHAR

Y Field (numeric) *

MPG DOUBLE



Japanese cars demonstrated the best fuel efficiency, followed by European cars, with US cars showing the lowest MPG on average. This reflects different design philosophies and market demands across regions during this period.

Weight vs MPG Relationship

cars

Chart settings

Chart type *

Scatter Plot

X Field *



SQLRooms
Analytics workspace

cars / Dashboard

+ New

Add data

DATA

- cars
  - main
    - cars 406 rows
      - varchar Car
      - double MPG
      - bigint Cylinders
      - double Displacement
      - bigint Horsepower
      - bigint Weight
      - double Acceleration
      - bigint Model

WORKSPACE + New

- Dashboard
- App: Playful car showro...
- App: MPG vs Horsepow...
- Cars insights
- Python
- Cars with Python chart

AI ASSISTANT

Cars Dataset Dashboard Ch...

add charts to dashboard for exploring the cars dataset

I'll help you explore the cars dataset by adding informative charts to your dashboard. Let me start by examining the data structure and then populate the dashboard with various charts.

The user wants to explore the cars dataset through the dashboard. The dashboard agent will analyze the data and create multiple relevant visualizations showing distributions, relationships, and comparisons across key attributes like MPG, horsepower, weight, cylinders, and origin.

Worked with 13 tools

Add context

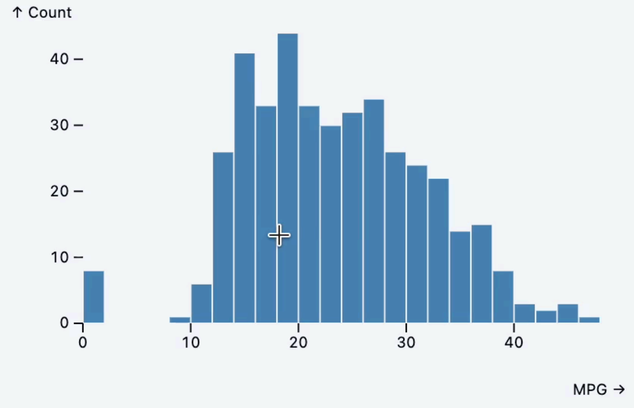
What would you like to learn about the data?

claude-sonnet-4.5

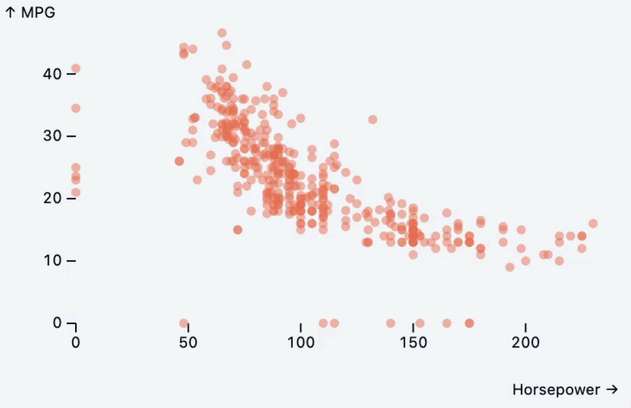
Dashboard

cars

Distribution of MPG (Fuel Efficiency)



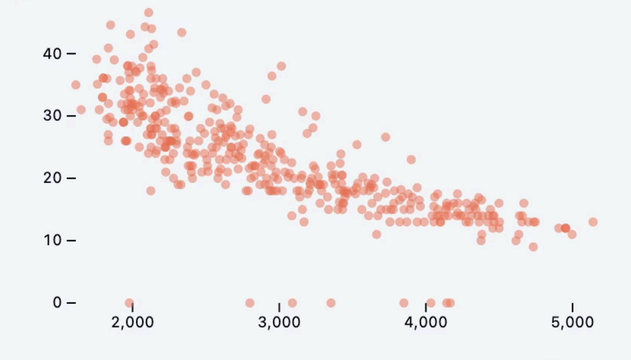
Horsepower vs MPG



Average MPG by Origin (Region)



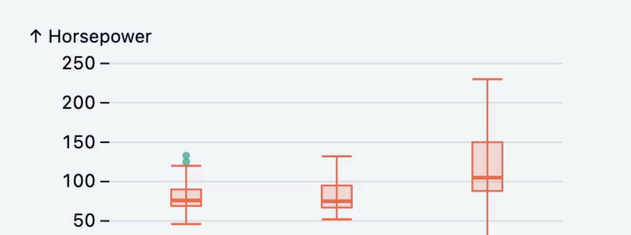
Weight vs MPG



Distribution of Cars by Number of Cylinders



Horsepower by Origin



SQLRooms
Analytics workspace

Add data

DATA

- cars
  - main
    - cars 406 rows

WORKSPACE + New

- App: Playful car showro...
- App: MPG vs Horsepow...
- Cars insights
- Python
- Cars with Python chart

cars / Cars with Python chart

+ New

Cars with Python chart

Python
Last run succeeded in 2610 ms

Success

```python
import altair as alt
import sqlrooms

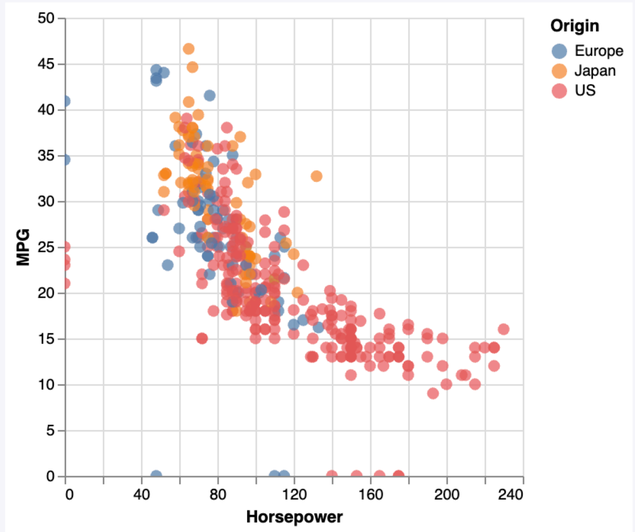
source = sqlrooms.query_df("SELECT * FROM cars")

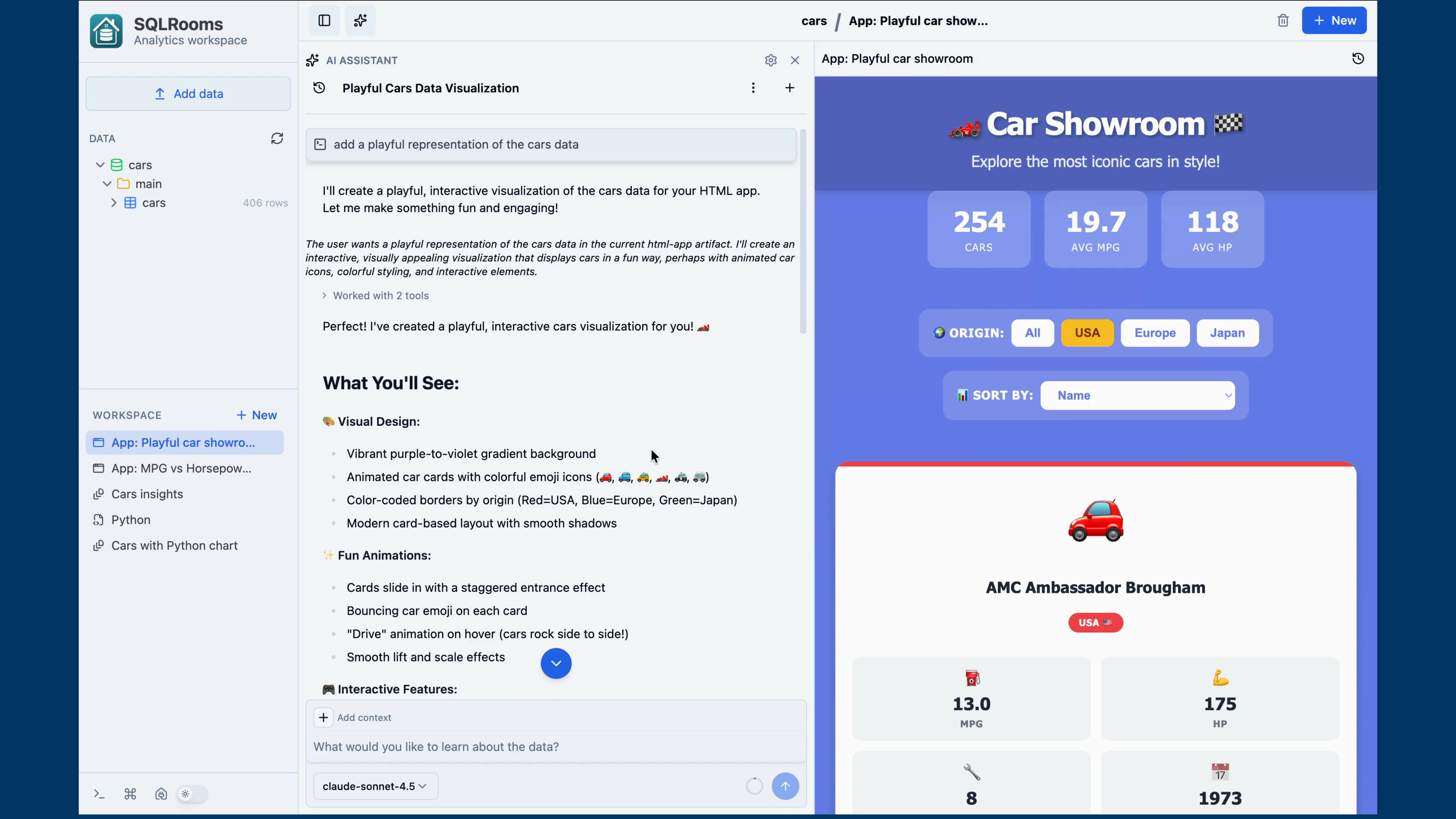
alt.Chart(source).mark_circle(size=60).encode(
    x='Horsepower',
    y='MPG',
    color='Origin',
    tooltip=['Model', 'Origin', 'Horsepower', 'MPG']
).interactive()
```

stdout

```
numpy already loaded from default channel
python-dateutil already loaded from default channel
pytz already loaded from default channel
Loading pandas
Loaded pandas
```

result




SQLRooms
Analytics workspace
Add data
DATA
cars
main
cars
406 rows
WORKSPACE
New
App: Playful car showro...
App: MPG vs Horsepow...
Cars insights
Python
Cars with Python chart
cars / App: Playful car show...
New
AI ASSISTANT
Playful Cars Data Visualization
add a playful representation of the cars data
I'll create a playful, interactive visualization of the cars data for your HTML app.
Let me make something fun and engaging!
The user wants a playful representation of the cars data in the current html-app artifact. I'll create an interactive, visually appealing visualization that displays cars in a fun way, perhaps with animated car icons, colorful styling, and interactive elements.
Worked with 2 tools
Perfect! I've created a playful, interactive cars visualization for you!
What You'll See:
Visual Design:
Vibrant purple-to-violet gradient background
Animated car cards with colorful emoji icons
Color-coded borders by origin (Red=USA, Blue=Europe, Green=Japan)
Modern card-based layout with smooth shadows
Fun Animations:
Cards slide in with a staggered entrance effect
Bouncing car emoji on each card
"Drive" animation on hover (cars rock side to side!)
Smooth lift and scale effects
Interactive Features:
Add context
What would you like to learn about the data?
claude-sonnet-4.5
App: Playful car showroom
Car Showroom
Explore the most iconic cars in style!
254
CARS
19.7
AVG MPG
118
AVG HP
ORIGIN:
All
USA
Europe
Japan
SORT BY:
Name
AMC Ambassador Brougham
USA
13.0
MPG
175
HP
8
1973

Thanks!

DuckDB-native analytical apps

↓

lightweight, portable, inspectable, reproducible

sqlrooms.org

linkedin.com/in/ilyabo