

DuckDB as an analytical runtime

Local-first analytics apps with SQLRooms



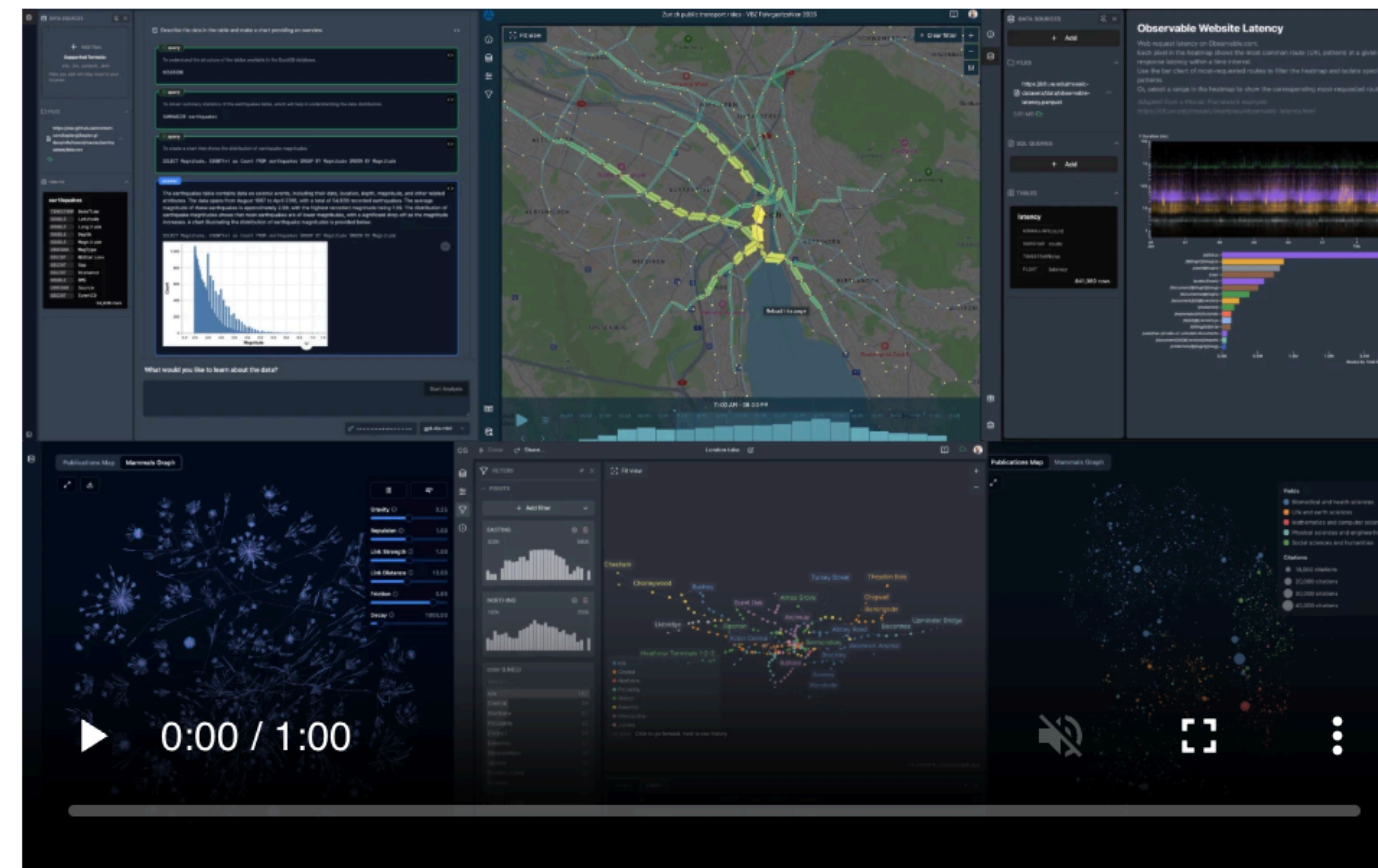
**What if a DuckDB file could contain
an entire analytics workspace**



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.



Supported by

FOURSQUARE



DuckDB integration

Connectors

Table/query management

The screenshot shows the SQLRooms Query Workbench interface. On the left, a sidebar displays the database structure: `database` (expanded) contains `main` (expanded), which contains the `trips` table. The table schema is listed as follows:

- `route_id` (varchar)
- `service_id` (varchar)
- `trip_id` (varchar)
- `trip_headsign` (varchar)
- `trip_short_name` (varchar)
- `direction_id` (integer)
- `block_id` (varchar)
- `shape_id` (varchar)
- `wheelchair_accessible` (integer)
- `bikes_allowed` (integer)

The main query editor shows the following SQL query:

```
1 SELECT * FROM trips
2
```

The query results are displayed in a table with 12 rows and 6 columns:

	route_id	service_id	trip_id	trip_headsign	trip_sho
1	1	1622	1064081	NULL	NULL
2	1	1622	1109021	NULL	NULL
3	1	1622	1131309	NULL	NULL
4	1	1622	222089	NULL	NULL
5	1	1622	377415	NULL	NULL
6	1	1622	559587	NULL	NULL
7	1	1622	566581	NULL	NULL
8	1	1622	605543	NULL	NULL
9	1	5563	1129148	NULL	NULL
10	1	5563	1447209	NULL	NULL
11	1	5563	1510494	NULL	NULL
12	1	5563	396975	NULL	NULL

At the bottom of the interface, it indicates "100 rows" and "Limit results to 100 rows". A "+ New table" button is visible in the bottom right corner.

UI/Layout

Panels

Tabs

Grid

The screenshot displays a data visualization dashboard with a dark theme. On the left, a sidebar contains a 'SCHEMA' panel showing the structure of a table named 'sample_data' with columns: id (INTEGER), name (VARCHAR), value (DOUBLE), and created_at (TIMESTAMP). The main area is a grid layout containing four bar charts: 'Chart growth-grid-retention' (blue bars), 'Chart growth-grid-conversions' (orange bars), 'Chart growth-grid-sessi...' (green bars), and 'Chart chart-lek2zv4lcejilz3s0jwfpf6t' (purple bars). A '+ Add Chart' button is in the top right. At the bottom, a 'Console' panel shows the following log output:

```
[10:23:15] Connected to DuckDB
[10:23:16] Loaded sample_data.csv (1,234 rows)
[10:23:18] Query executed successfully (0.032s)
[10:23:18] 50 rows returned
```

Artifacts

Pluggable

Composable

 New Worksheet

 Dashboard

 Pivot Table

 Notebook

 Document

 SQL Query

 HTML App

 Python

 Canvas


 App Builder


```
uv tool install sqlrooms
```

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

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

 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

×

 New Worksheet

 Dashboard

 Pivot Table

 Notebook

 Document

 SQL Query

 HTML App

 Python

 Canvas

 App Builder

↑ 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

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

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

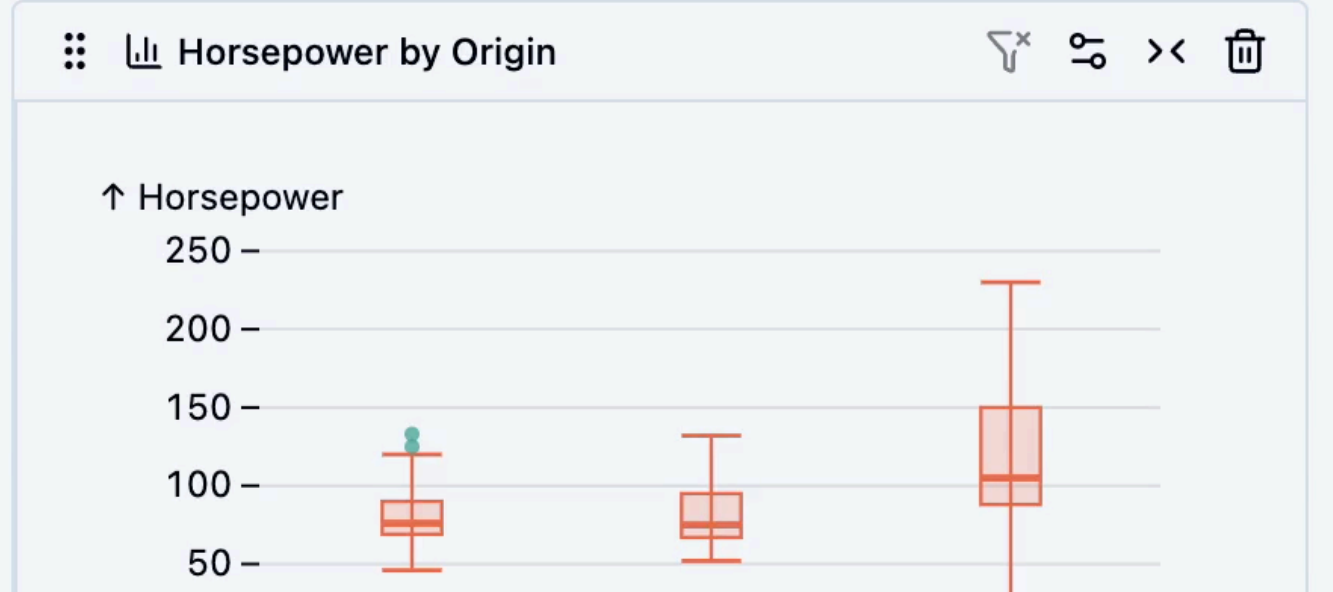
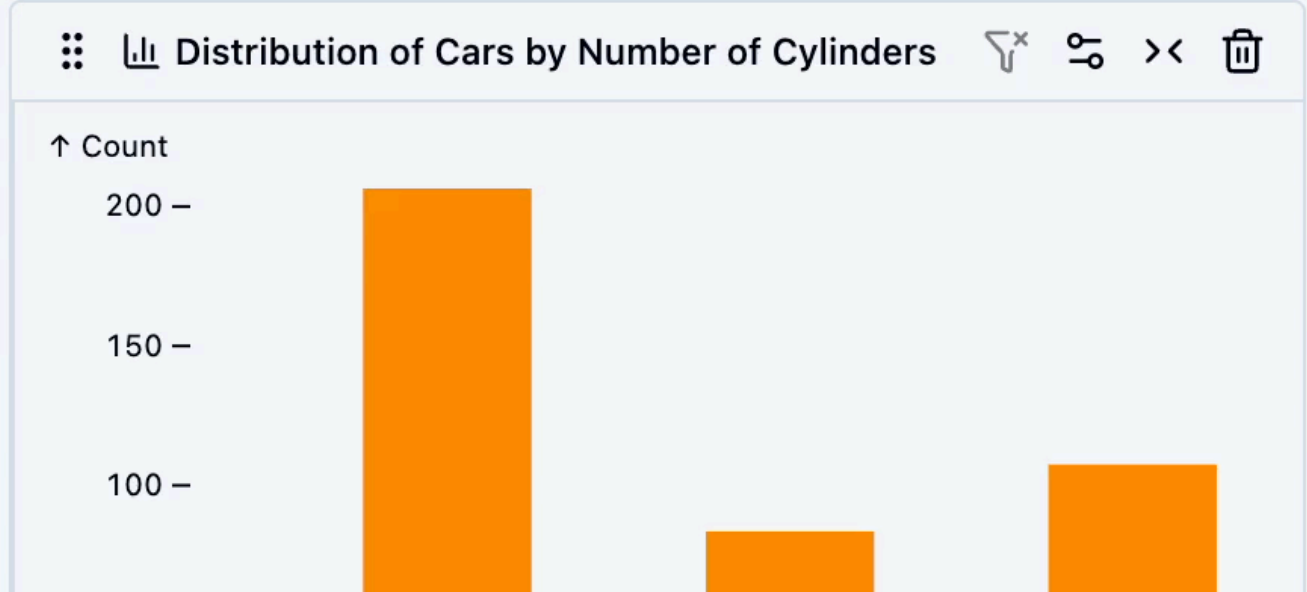
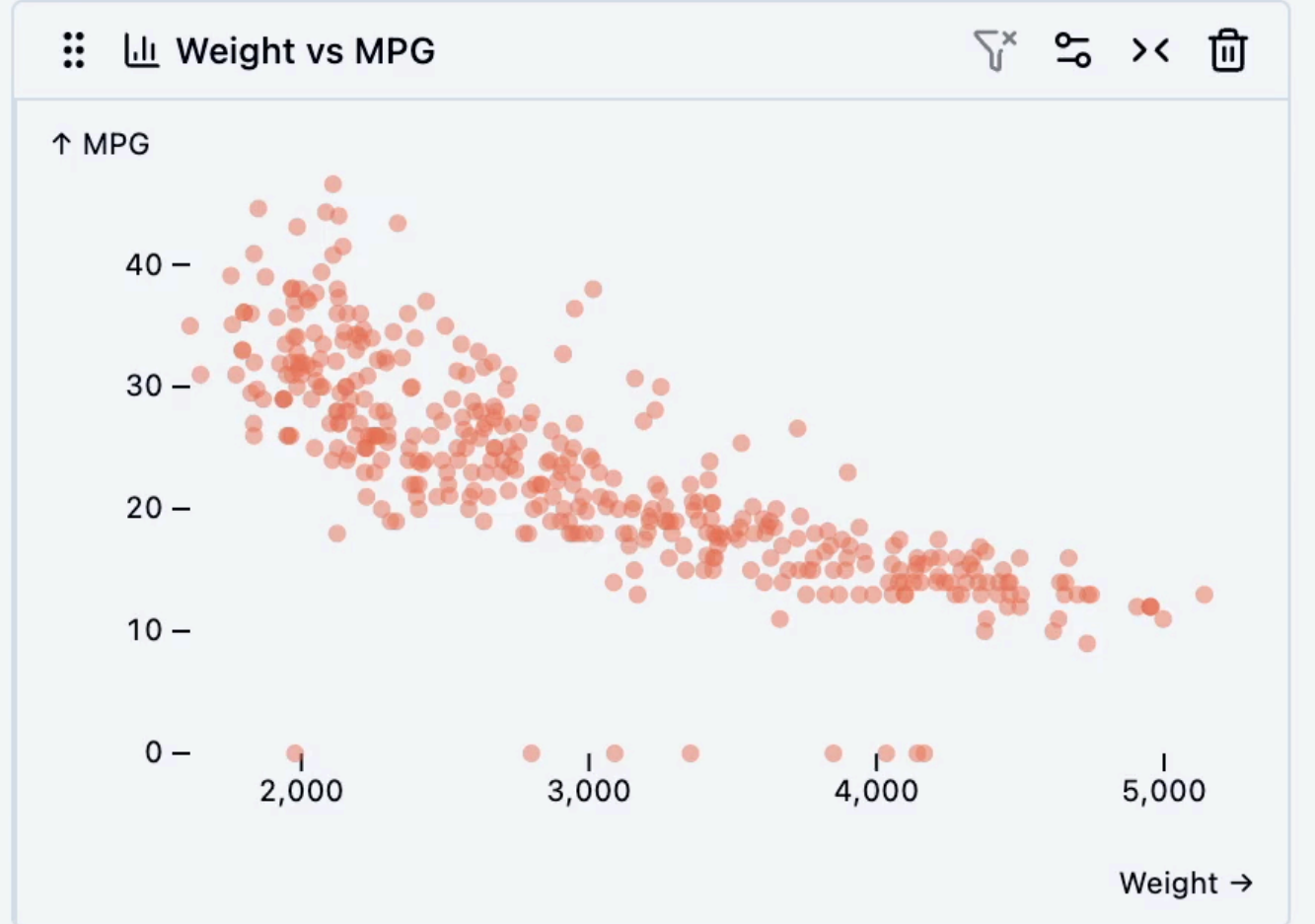
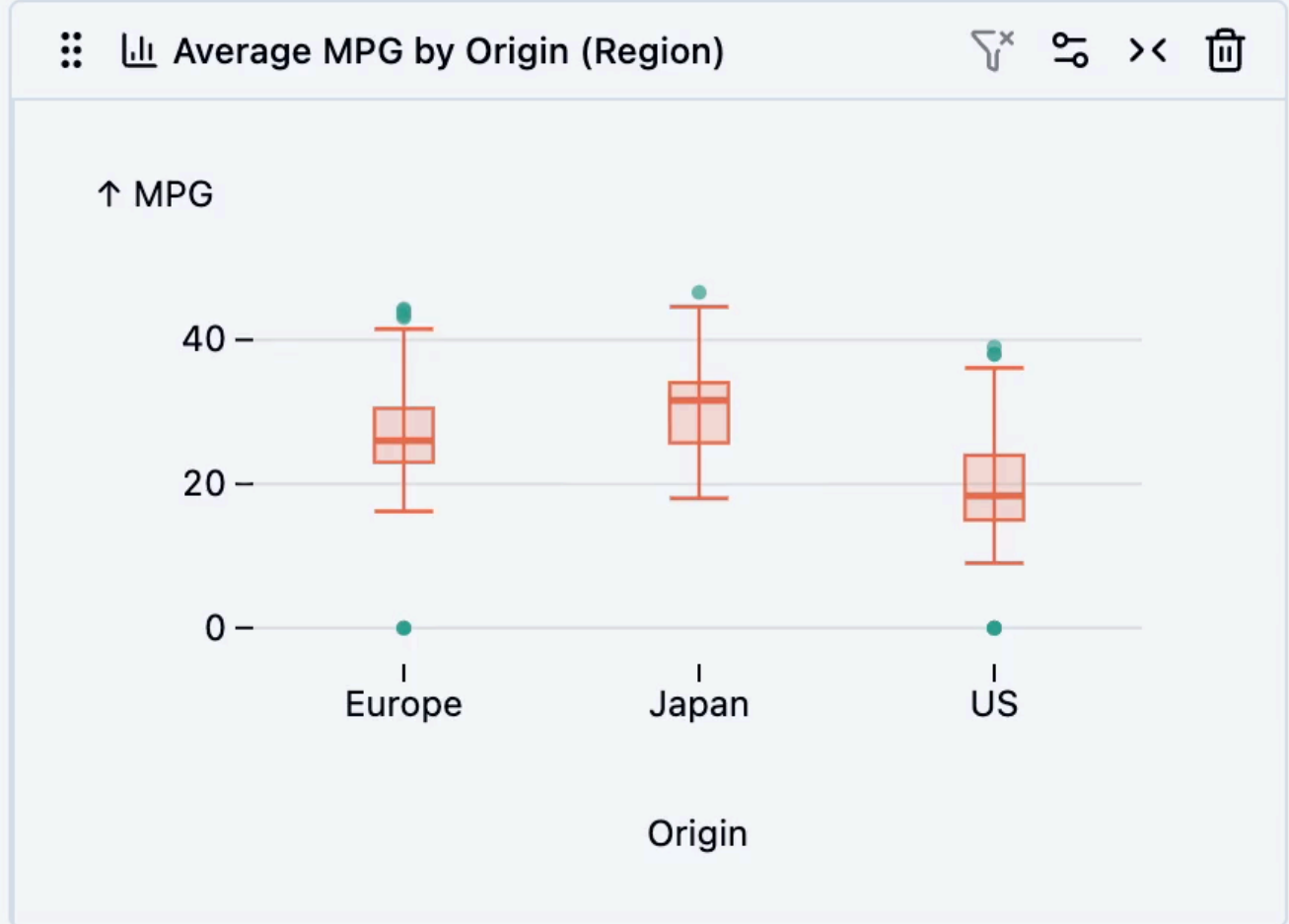
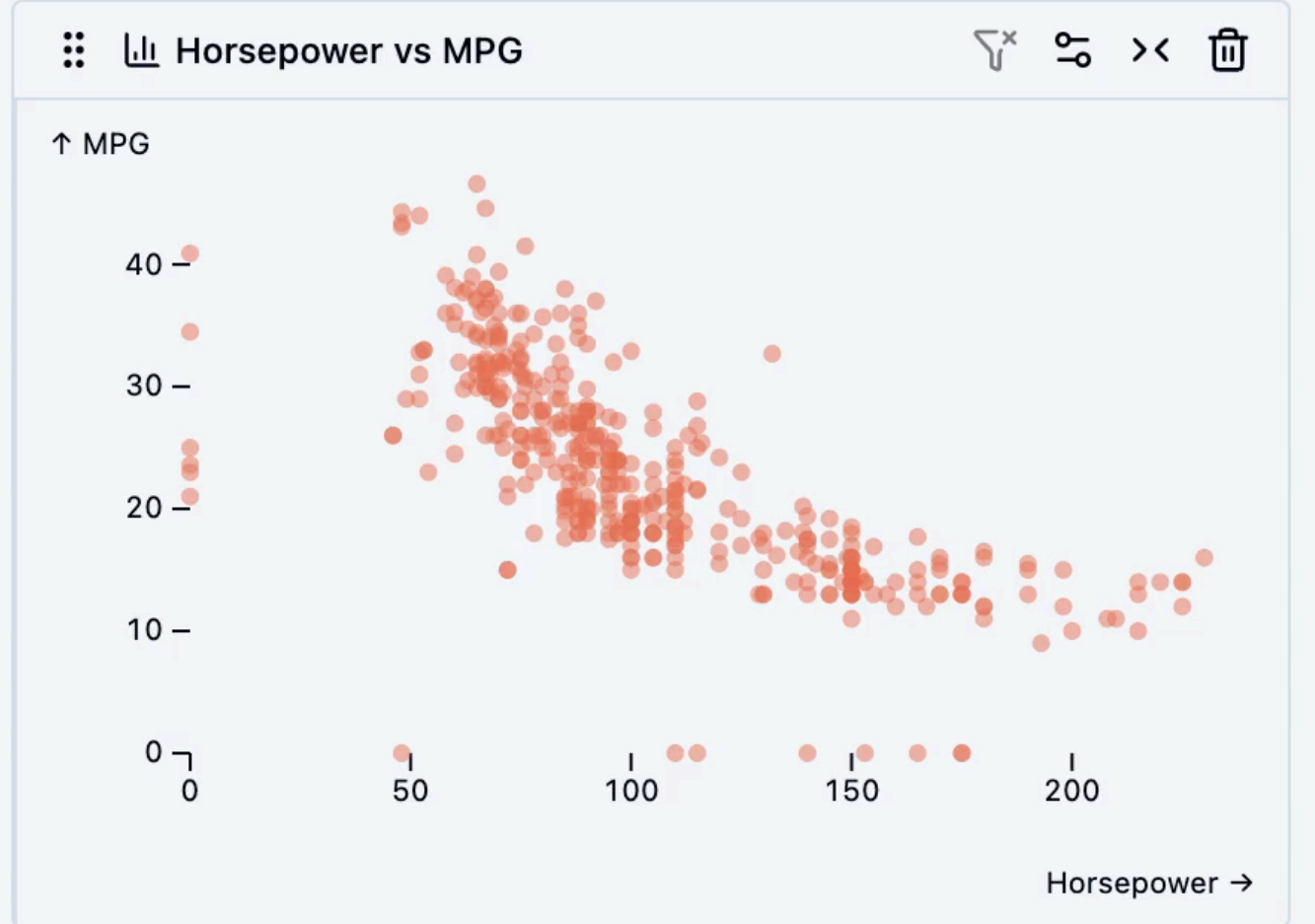
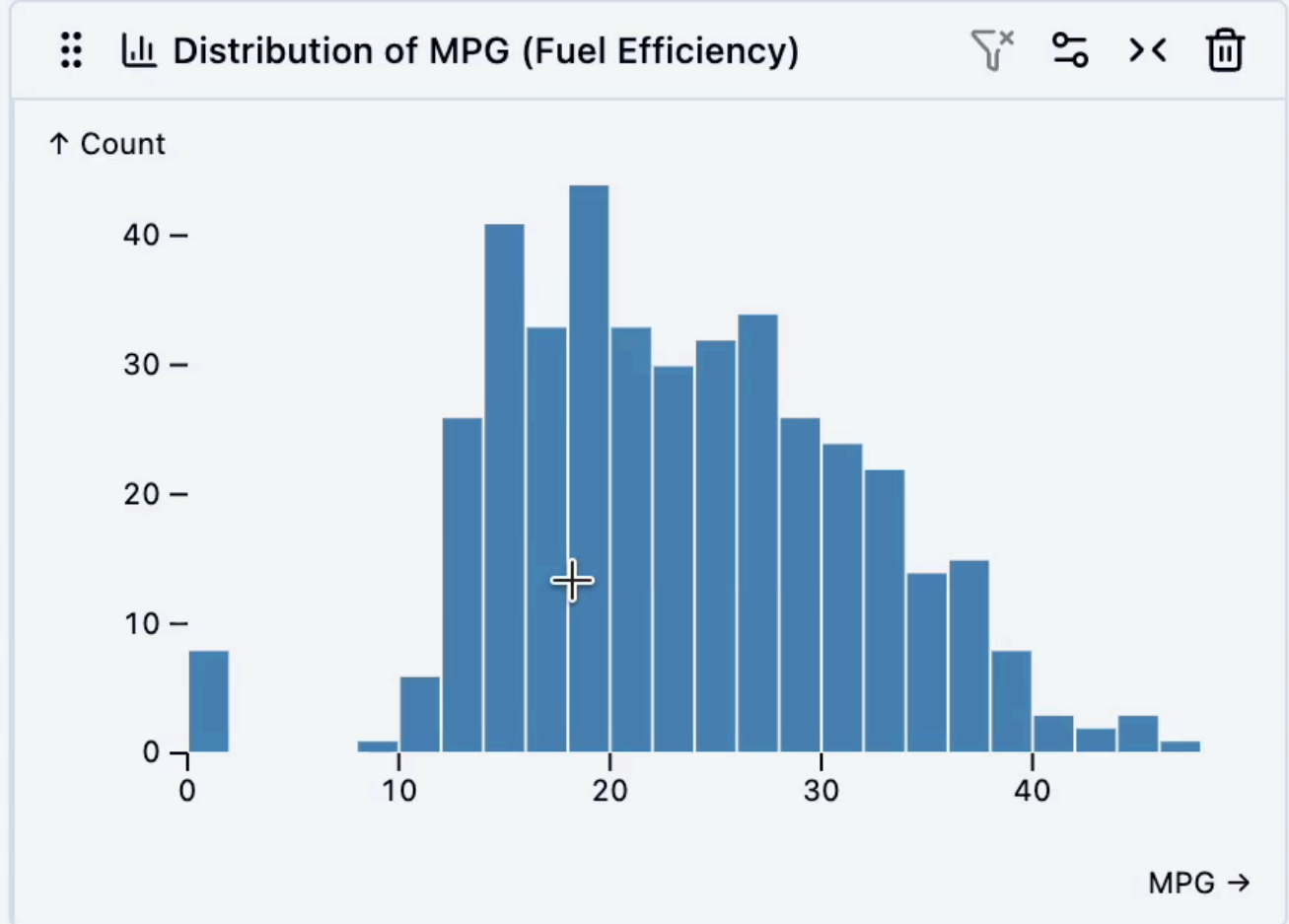
...

+ Add context

What would you like to learn about the data?

claude-sonnet-4.5

Dashboard cars



↑ 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

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 improved **dramatically** over the model years in the dataset
- 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

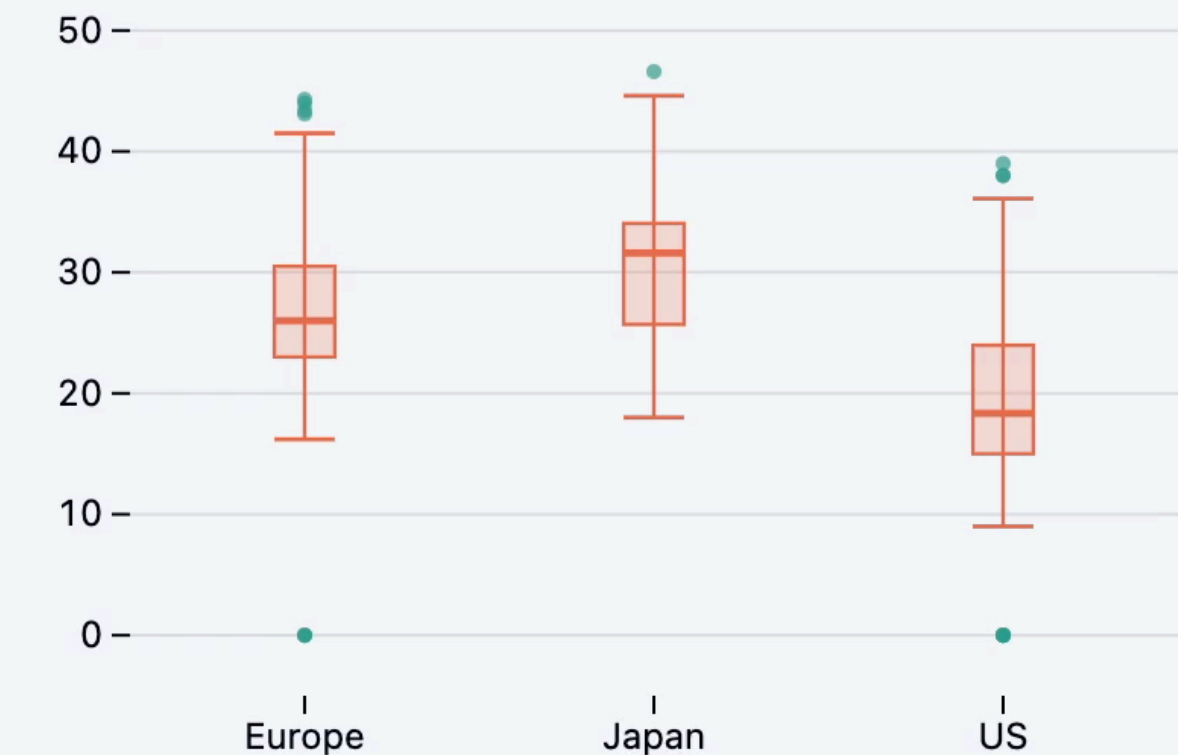
X Field (categorical) *

Origin VARCHAR

Y Field (numeric) *

MPG DOUBLE

↑ MPG



Origin

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

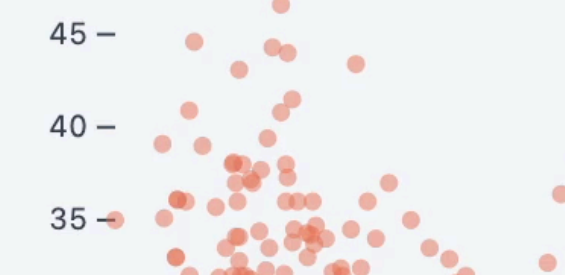
Chart settings

Chart type *

Scatter Plot

X Field *

↑ MPG



↑ 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 with Python chart

Python

Last run succeeded in 2610 ms

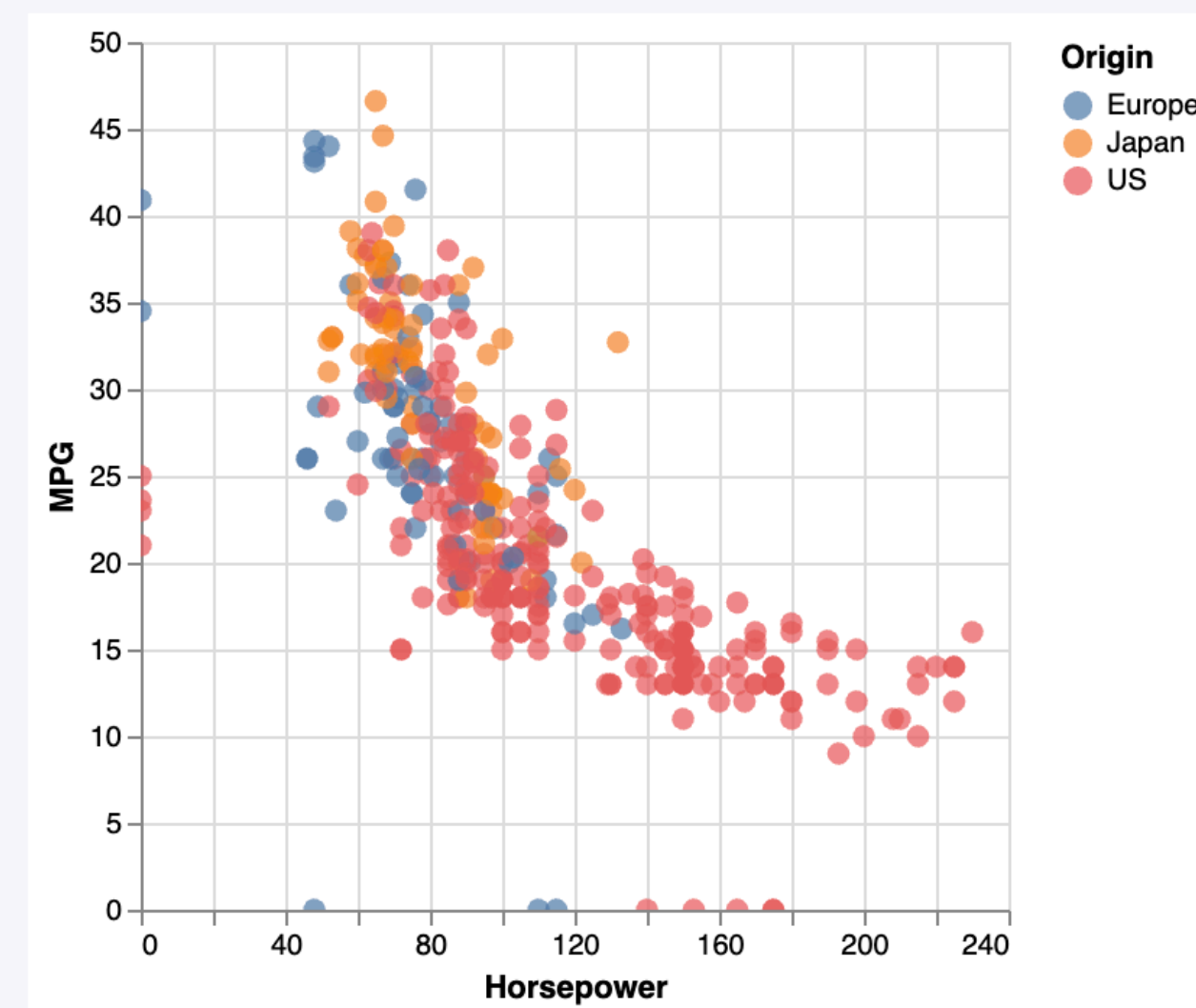
Success

```
1 import altair as alt
2 import sqlrooms
3
4 source = sqlrooms.query_df("SELECT * FROM cars")
5
6 alt.Chart(source).mark_circle(size=60).encode(
7     x='Horsepower',
8     y='MPG',
9     color='Origin',
10    tooltip=['Model', 'Origin', 'Horsepower', 'MPG']
11 ).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



↑ 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

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, local-first,
inspectable, reproducible

sqlrooms.org

linkedin.com/in/ilyabo/