Quick and Easy Statistics and Histograms with DuckDB

Josef Machytka
3 min read · 13 hours ago

DuckDB is an exceptional tool that demonstrates how tasks requiring sometimes considerable manual effort in other tools can be accomplished efficiently and almost effortlessly. In my previous articles I have written for example about using it for ETL tasks, and another great use case is statistical calculations.

While classical databases provide standard aggregation functions, their usage requires crafting specific SQL queries. Similarly, libraries like Pandas offer tools like the `describe` command, but these demand some Python programming or running Jupyter notebooks. DuckDB, however, simplifies the entire process.

Recently, I enjoyed DuckDB's simplicity while analyzing load test results. My Python testing program writes multiple metrics and the test results into a CSV file while running queries with randomized `WHERE` conditions over multiple parallel sessions.

The resulting CSV file is typically modest in size —just a few MBs and several thousands rows — but it contains dozens of columns. I intentionally log as much data as possible to prepare for various performance analysis needs, especially when anomalies occur. Since I cannot always predict which columns will be crucial or what anomalies I might encounter, flexibility is key.

In the past, I used Python scripts to parse and analyze these results. However, maintaining and adapting these scripts to evolving requirements is tedious, to say the least. This time, I decided to leverage DuckDB and see how it would perform.

In this case, I noticed already during tests a broader spread in runtime values than expected, so I started with basic statistics using the `summarize` command:

```
.mode line

summarize select total_run_time_milliseconds from 'testing_results.csv';

    column_name = total_run_time_milliseconds
    column_type = DOUBLE
            min = 6.01
            max = 13187.19
  approx_unique = 5821
            avg = 531.9073925925913
            std = 649.844882272996
            q25 = 274.9243782877842
            q50 = 412.318166933531
            q75 = 647.6378482942634
          count = 6750
null_percentage = 0.00
```

Max value is extremely high, but the average and quartiles suggest that I am facing rather smaller amount of outliers. So let's confirm it by creating a histogram with buckets rounded to multiplies of 200:

```
select
    key_value['key'] as bucket_max,
    key_value['value'] as data_points
from (
    select unnest(map_entries(histogram(
        round(total_run_time_milliseconds/200)*200))) as key_value
    from 'testing_results.csv'
);
```

```
┌────────────┬─────────────┐
│ bucket_max │ data_points │
│   double   │   uint64    │
├────────────┼─────────────┤
│        0.0 │         275 │
│      200.0 │        1755 │
│      400.0 │        2110 │
│      600.0 │        1209 │
│      800.0 │         553 │
│     1000.0 │         375 │
│     1200.0 │         127 │
│     1400.0 │         269 │
│     1600.0 │          58 │
│     2200.0 │           1 │
│     3600.0 │           1 │
│     4000.0 │           1 │
│     5400.0 │           1 │
│     6000.0 │           2 │
│    12800.0 │           8 │
│    13000.0 │           2 │
│    13200.0 │           3 │
├────────────┴─────────────┤
│ 17 rows        2 columns │
└──────────────────────────┘
```

Now, I can see the useful distribution of data points and focus my analysis on outliers. By correlating these outliers with other columns, I can uncover

additional performance anomalies, such as memory usage, load average, or disk I/O, related to these data points. But it goes beyond intended scope of this article and I will discuss it next time.

This short example shows how DuckDB by introducing simple yet powerful statistical commands eliminates the complexity of SQL queries or notebooks, enabling us to focus on insights rather than implementation and get very quickly useful results without too much manual work.

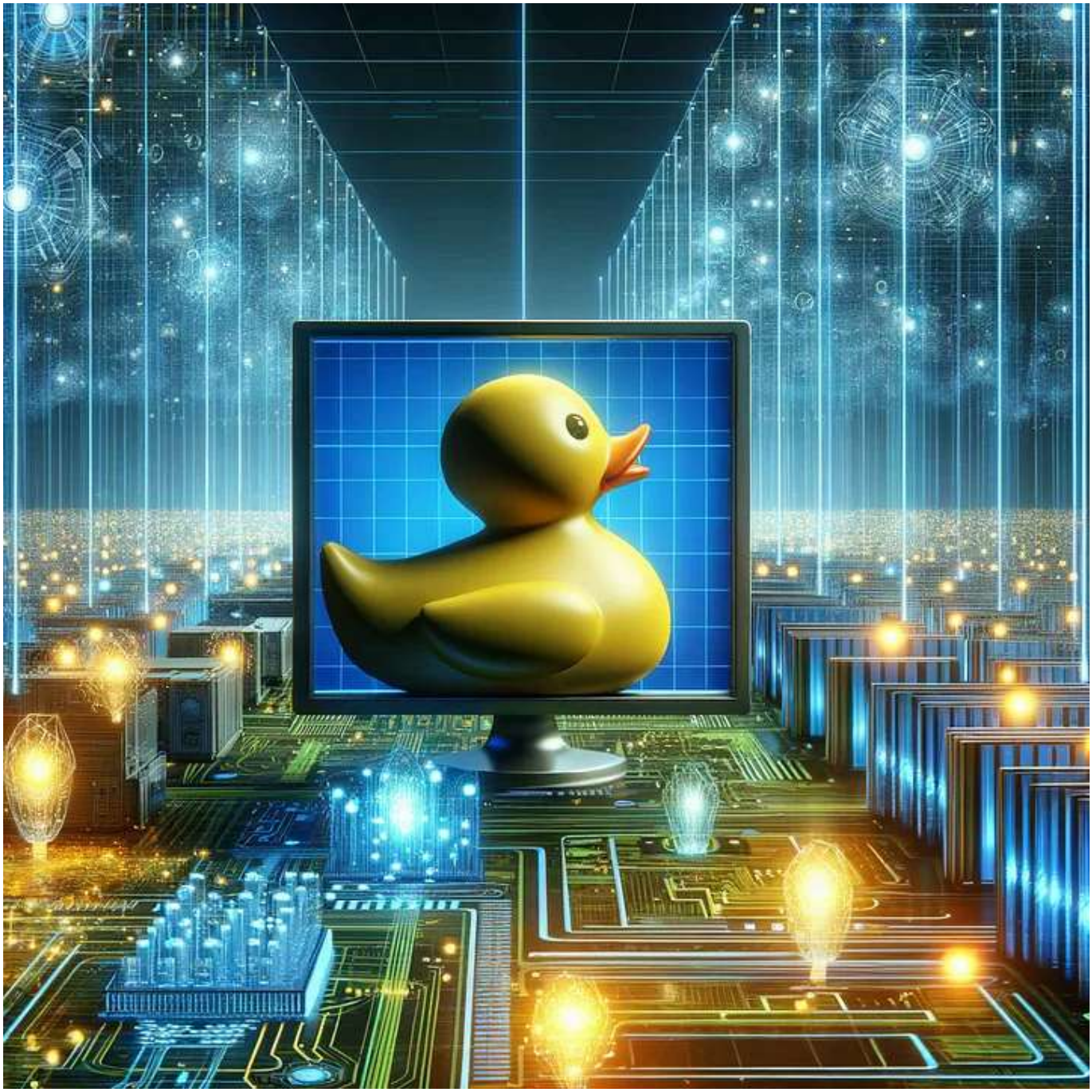


Image created by the author using DeepDreamGenerator

Duckdb Histograms Statistics

Written by Josef Machytka

39 Followers · 7 Following

Edit profile

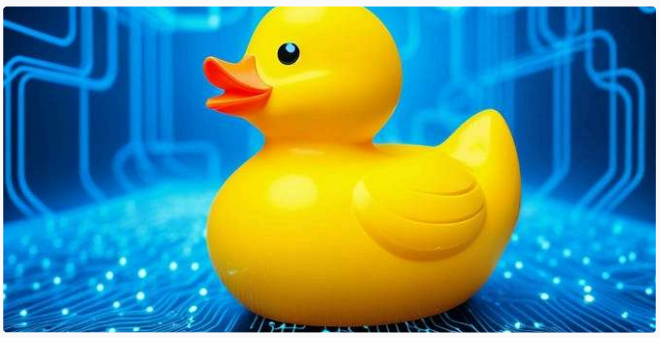
I work as Professional Service Consultant - PostgreSQL specialist in NetApp Deutschland GmbH, Open Source Services division.

No responses yet

What are your thoughts?

Respond

More from Josef Machytka



Josef Machytka

Extending DuckDB ETL Capabilities with Python

Josef Machytka

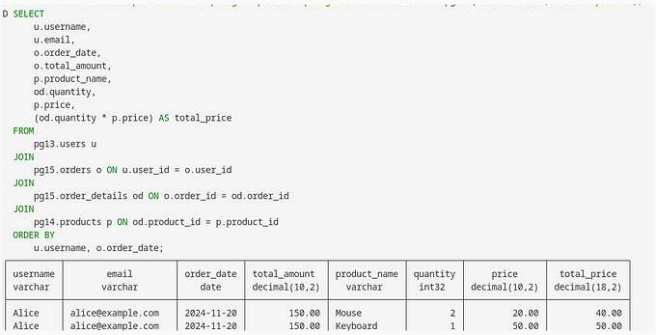
DuckDB as a Rudimentary Data Migration Tool

DuckDB has recently become my go-to solution for small ETL tasks. It is an...

Nov 24 20

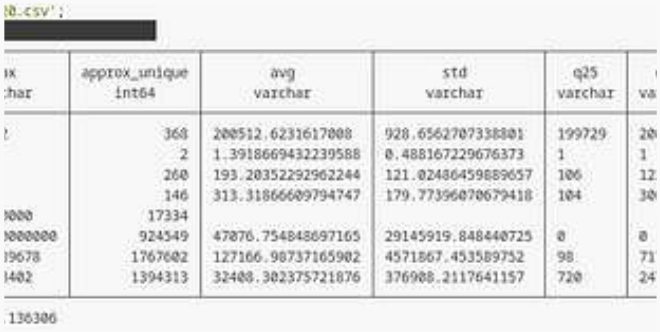
After exploring how to use DuckDB as an intelligent ETL tool for PostgreSQL, and how...

Nov 30 1



```
D SELECT
      u.username,
      u.email,
      o.order_date,
      o.total_amount,
      p.product_name,
      od.quantity,
      p.price,
      (od.quantity * p.price) AS total_price
  FROM
      pg13.users u
  JOIN
      pg15.orders o ON u.user_id = o.user_id
  JOIN
      pg15.order_details od ON o.order_id = od.order_id
  JOIN
      pg14.products p ON od.product_id = p.product_id
  ORDER BY
      u.username, o.order_date;
```

username varchar	email varchar	order_date date	total_amount decimal(10,2)	product_name varchar	quantity int32	price decimal(10,2)	total_price decimal(18,2)
Alice	alice@example.com	2024-11-20	150.00	Mouse	2	20.00	40.00
Alice	alice@example.com	2024-11-20	150.00	Keyboard	1	50.00	50.00



approx_unique int64	avg varchar	std varchar	q25 varchar
368	200512.6231617008	928.6562707338801	199729
2	1.3918669432239588	0.488167229676373	1
260	193.20352292962244	121.02486459889657	106
146	313.31866609794747	179.77396070679418	104
17334			
924549	47076.754848697165	29145919.848440725	0
1767602	127166.98737165902	4571867.453589752	98
1394313	32408.302375721876	376908.2117641157	720

Josef Machytka

Easy Cross-Database Selects with DuckDB

DuckDB was created with simplicity and ease of use in mind. In my previous article, I...

Nov 26 3

Josef Machytka

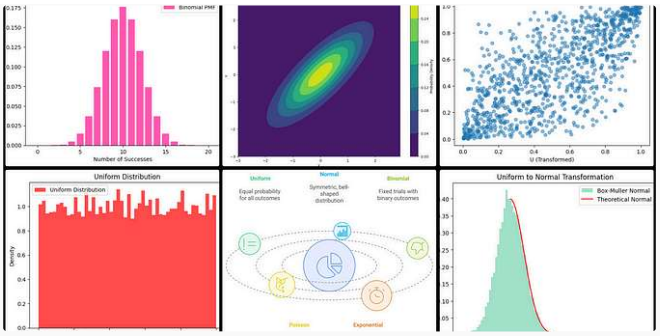
Using DuckDB as an Intelligent ETL tool for PostgreSQL

There is a lot of hype around DuckDB these days. At one PostgreSQL conference, I even...

Nov 2 7

See all from Josef Machytka

Recommended from Medium



In Tech Spectrum by Aarafat Islam

Understanding Statistical Distributions

Python implementation and visualization

Dec 3 161

Garvit Arya

Preventing Data Nightmares: Top 5 Data Quality Checks Every ETL...

Data is the lifeblood of any organization, but dirty or inconsistent data can lead to bad...

Dec 2 34 1

Lists

Predictive Modeling w/ Python

20 stories · 1722 saves

Practical Guides to Machine Learning

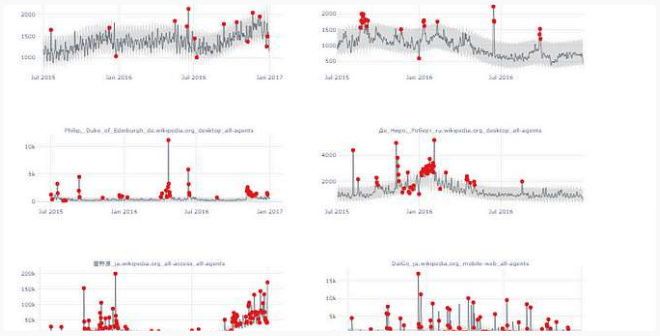
10 stories · 2094 saves

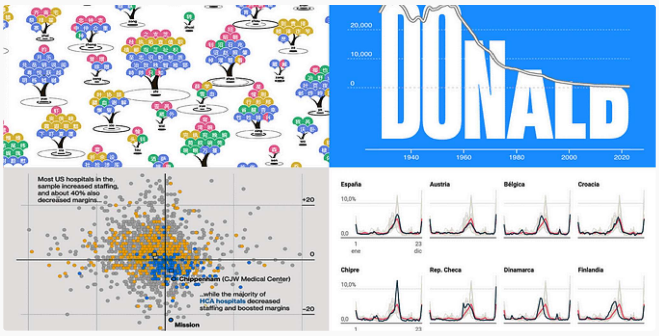
Productivity

242 stories · 639 saves

Medium's Huge List of Publications Accepting...

377 stories · 4130 saves





agus abdul rahman

Detect Anomalies Easily with pyTimeTK

Simplify Time Series Analysis and Spot Irregularities Faster!

Dec 7 64

AnyChart

Recent Data Visualization Projects Worth Exploring — DataViz Weekly

Check out four recent projects that showcase the power of data visualization applied acros...

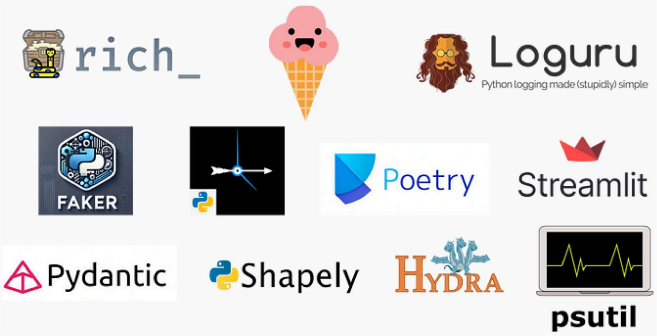
Dec 6 25

In The Pythoneers by Abhay Parashar

14 Foundational Concepts Every Python Programmer Should Master

Key concepts you can't afford to miss.

6d ago 567 4



In Level Up Coding by Md Arman Hossen

11 Python Libraries That Will 10x Your Development Speed in 2024:...

11 Game-Changing Python Libraries You've Been Missing in 2024

Dec 3 487 6

See more recommendations