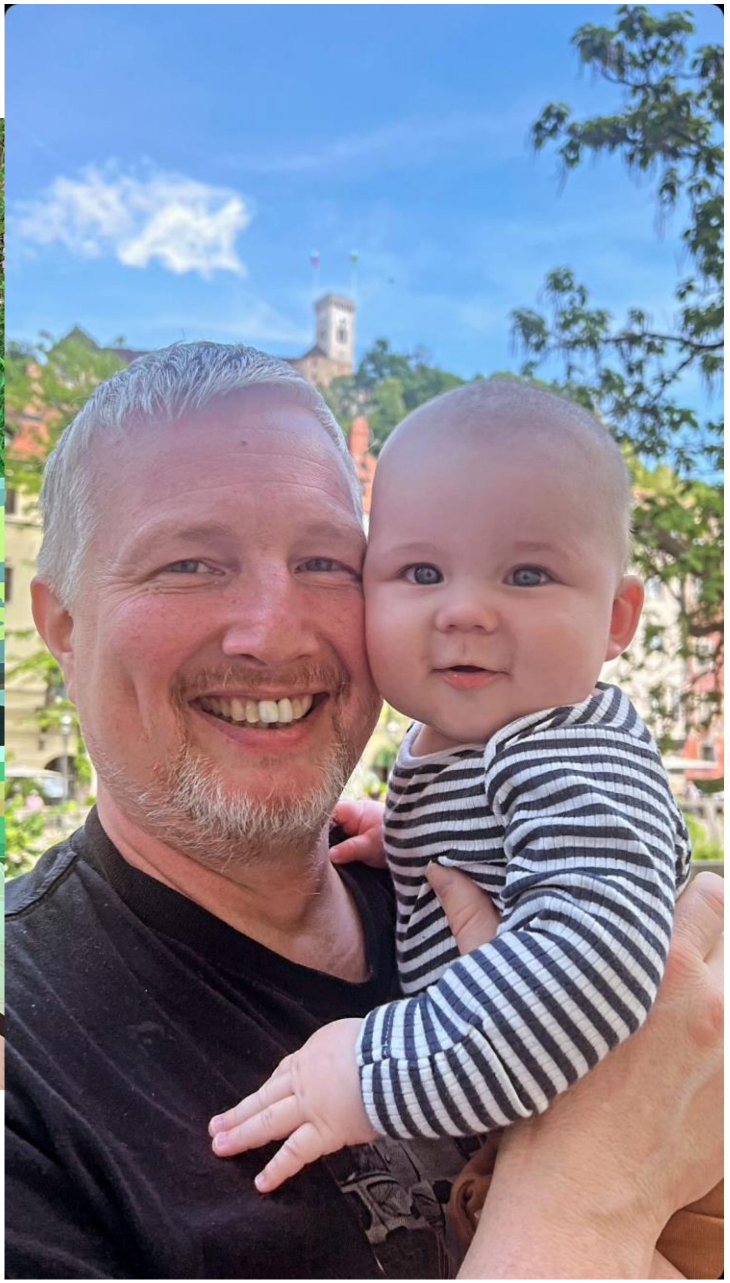
André

❤

DuckDB


DATA SATURDAY Holland

Microsoft MVP 2023 Most Valuable Professional

'98

(@ko_ver)

databricks dbt

@andrekamman | linkedin.com/in/andrekamman | andrekamman.com

Agenda

- What is it and where does it come from
- How do you install it and start it
- What makes it cool and different
  - Storage internals
  - FriendlySQL
  - Dataframe SQL
  - Formats and Extensions
  - CSV auto detection & Faulty line detection and handling
  - duck-dbt

What is it

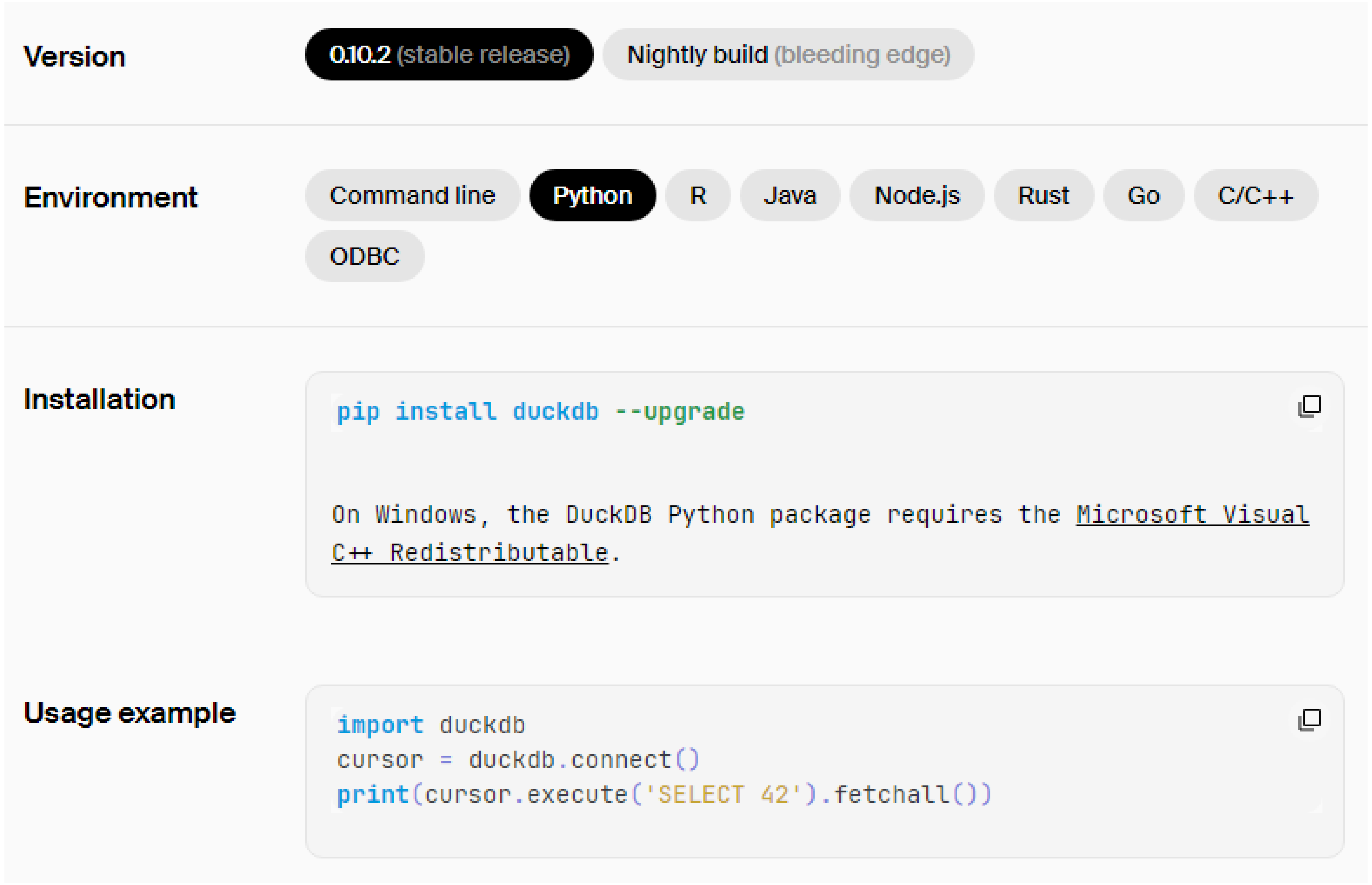
- C++
- In-Process
- MIT License
- 15ms startup time
- CLI, Python, R, Rust, Go, Node.js, Java, C, C++, etc.

Where does it come from?

- **Mark Raasveldt & Hannes Mühleisen**
- **Started in 2019 while working at CWI in Amsterdam**
- **Dutch National center for Math and Computer Science**
- **Guido van Rossum created Python at the same place many years ago**

- **Hannes had a pet duck, hence "DuckDB"**

Designed by Freepik


Version: **0.10.2** (stable release) | Nightly build (bleeding edge)

Environment: Command line | **Python** | R | Java | Node.js | Rust | Go | C/C++ | ODBC

Installation

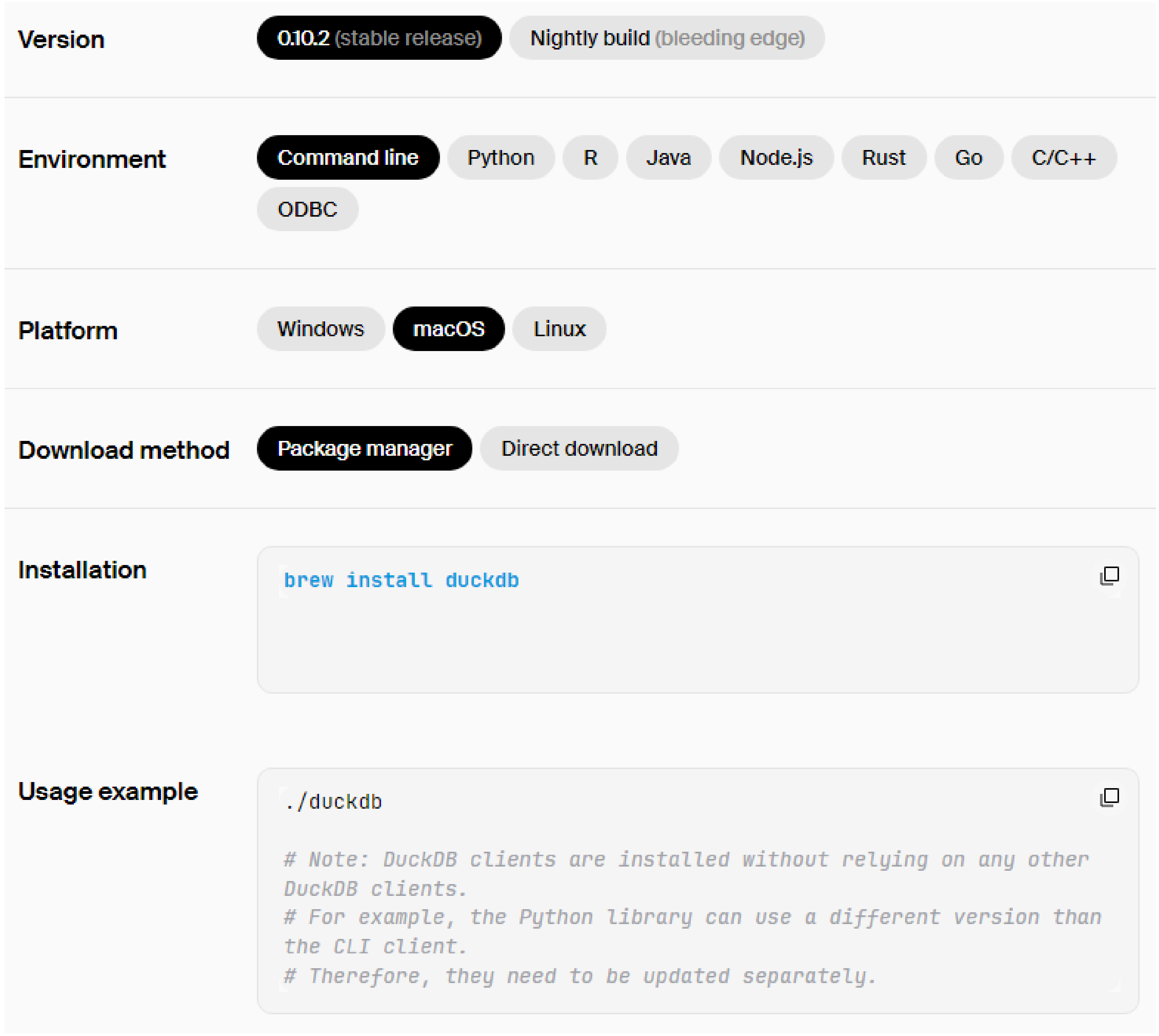
```
pip install duckdb --upgrade
```

On Windows, the DuckDB Python package requires the Microsoft Visual C++ Redistributable.

Usage example

```
import duckdb
cursor = duckdb.connect()
print(cursor.execute('SELECT 42').fetchall())
```

Source: https://duckdb.org/docs/installation



Version [✓ 0.10.2 (stable release)] Nightly build (bleeding edge)

Environment [✓ Command line] Python R Java Node.js Rust Go C/C++ ODBC

Platform Windows [✓ macOS] Linux

Download method [✓ Package manager] Direct download

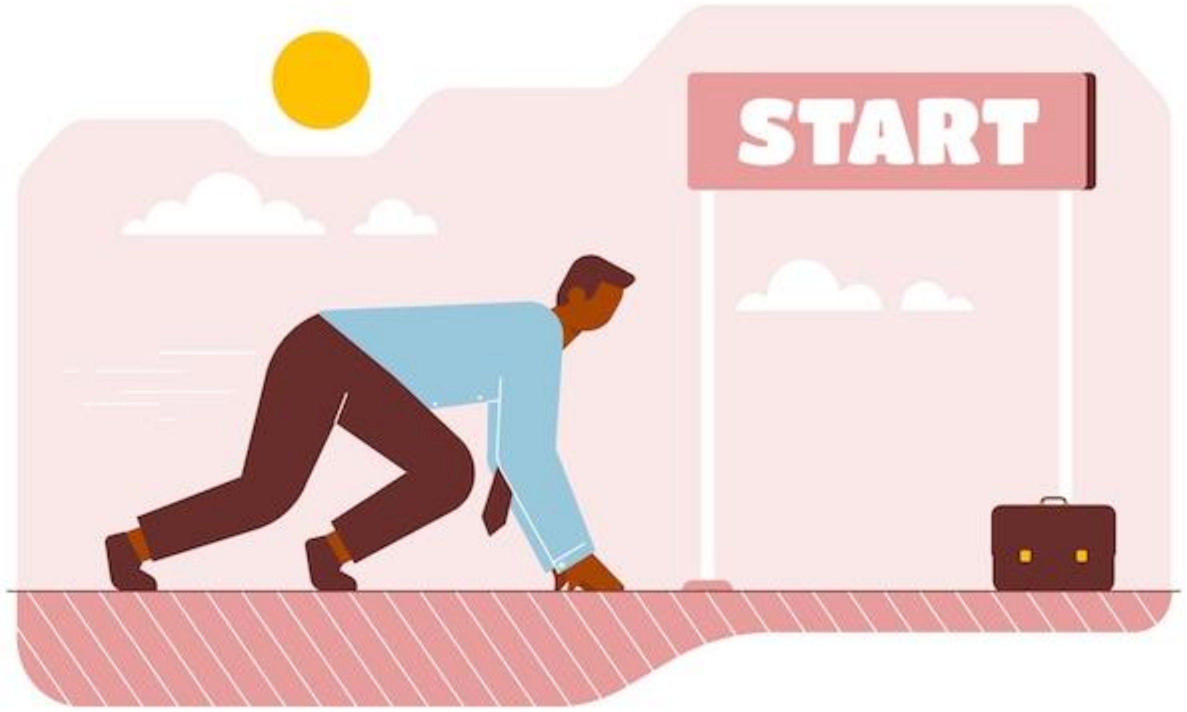
Installation

```
brew install duckdb
```

Usage example

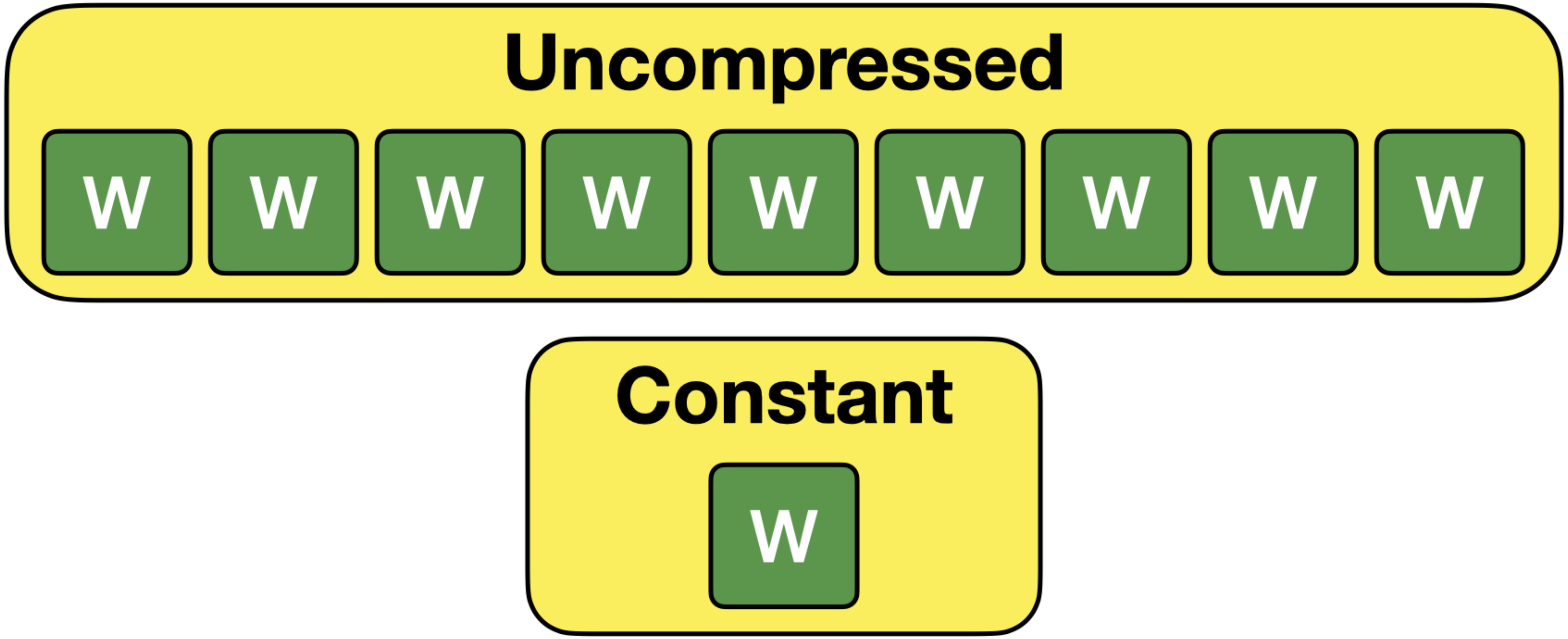
```
./duckdb

# Note: DuckDB clients are installed without relying on any other
DuckDB clients.
# For example, the Python library can use a different version than
the CLI client.
# Therefore, they need to be updated separately.
```

Source: https://duckdb.org/docs/installation

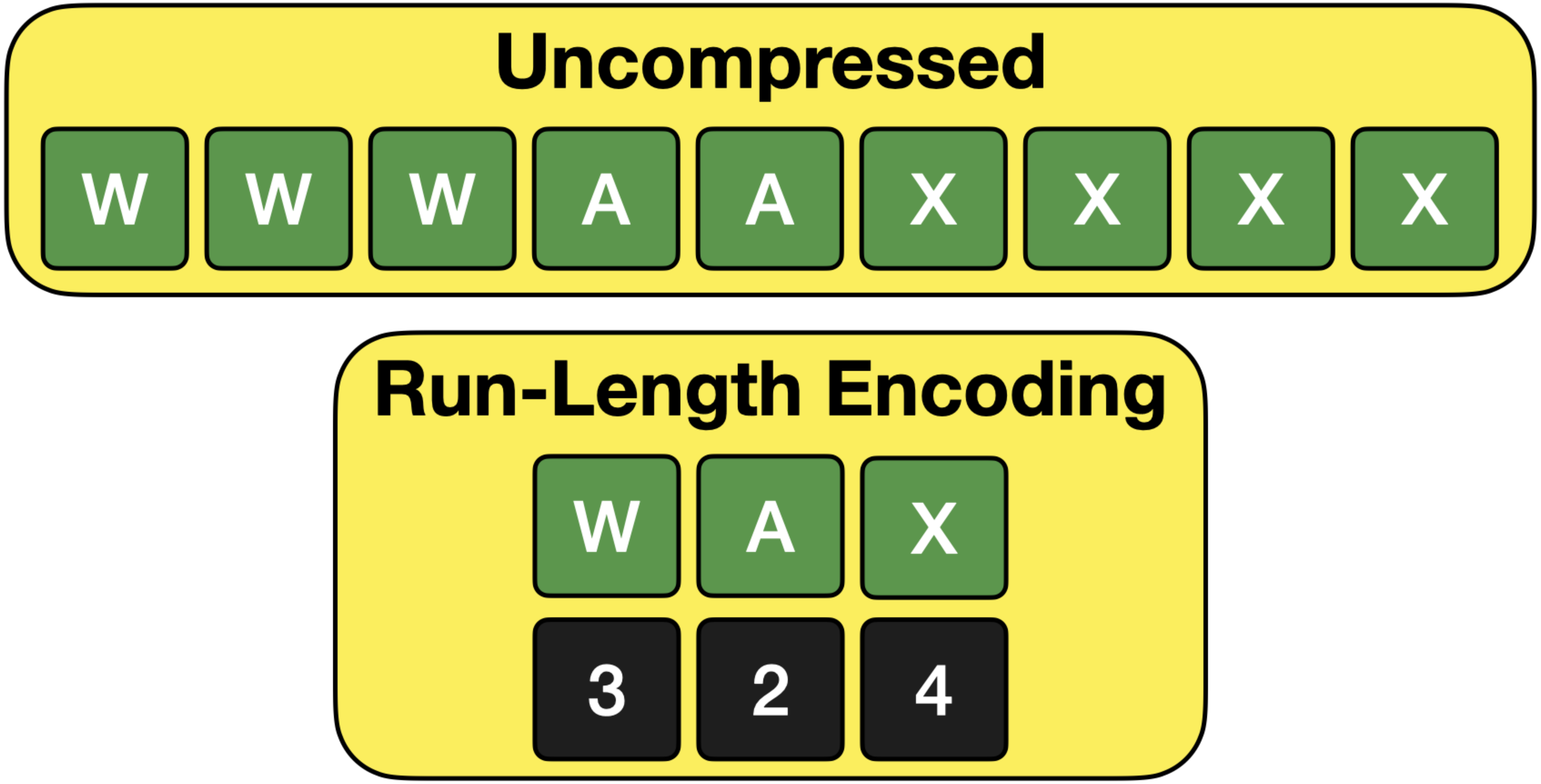


Designed by Freepik

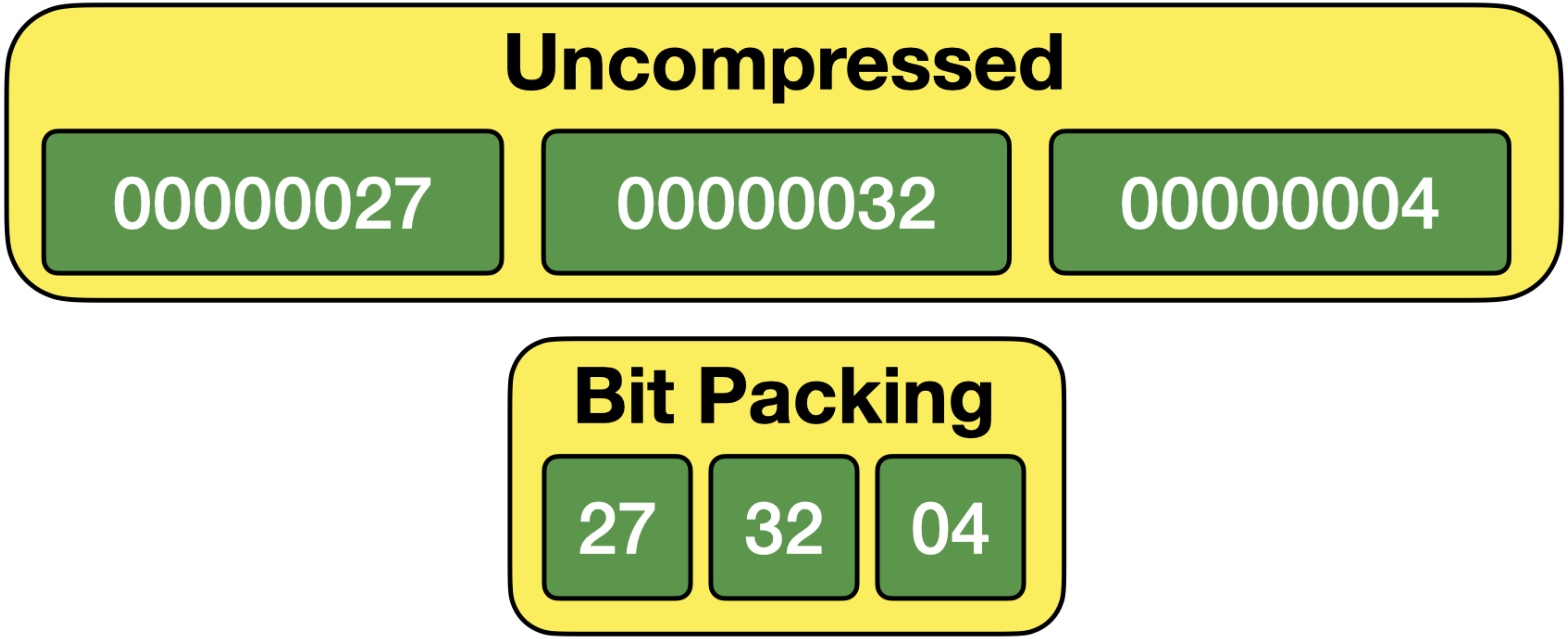
Designed by Freepik



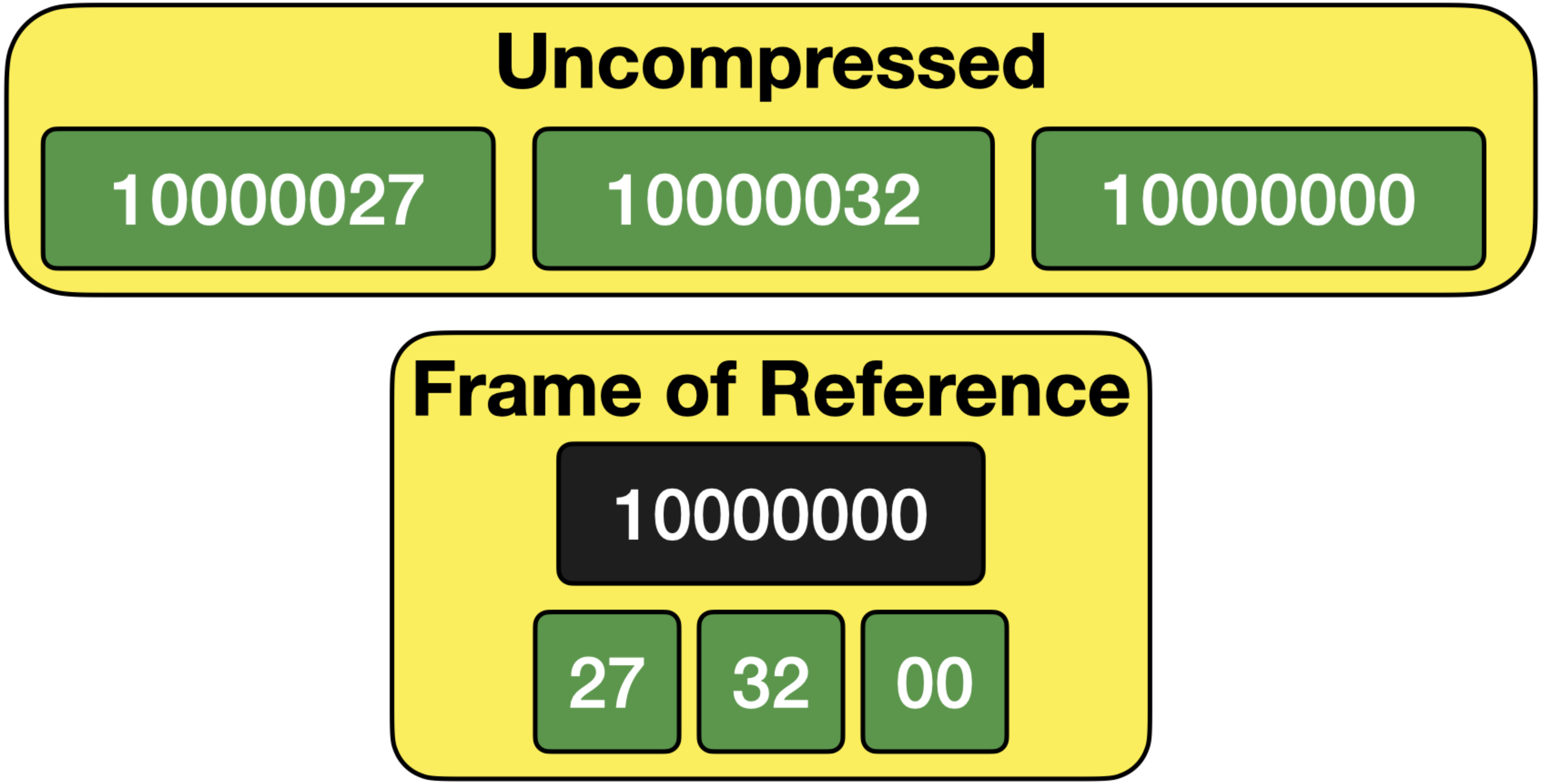
Source: https://duckdb.org/2022/10/28/lightweight-compression



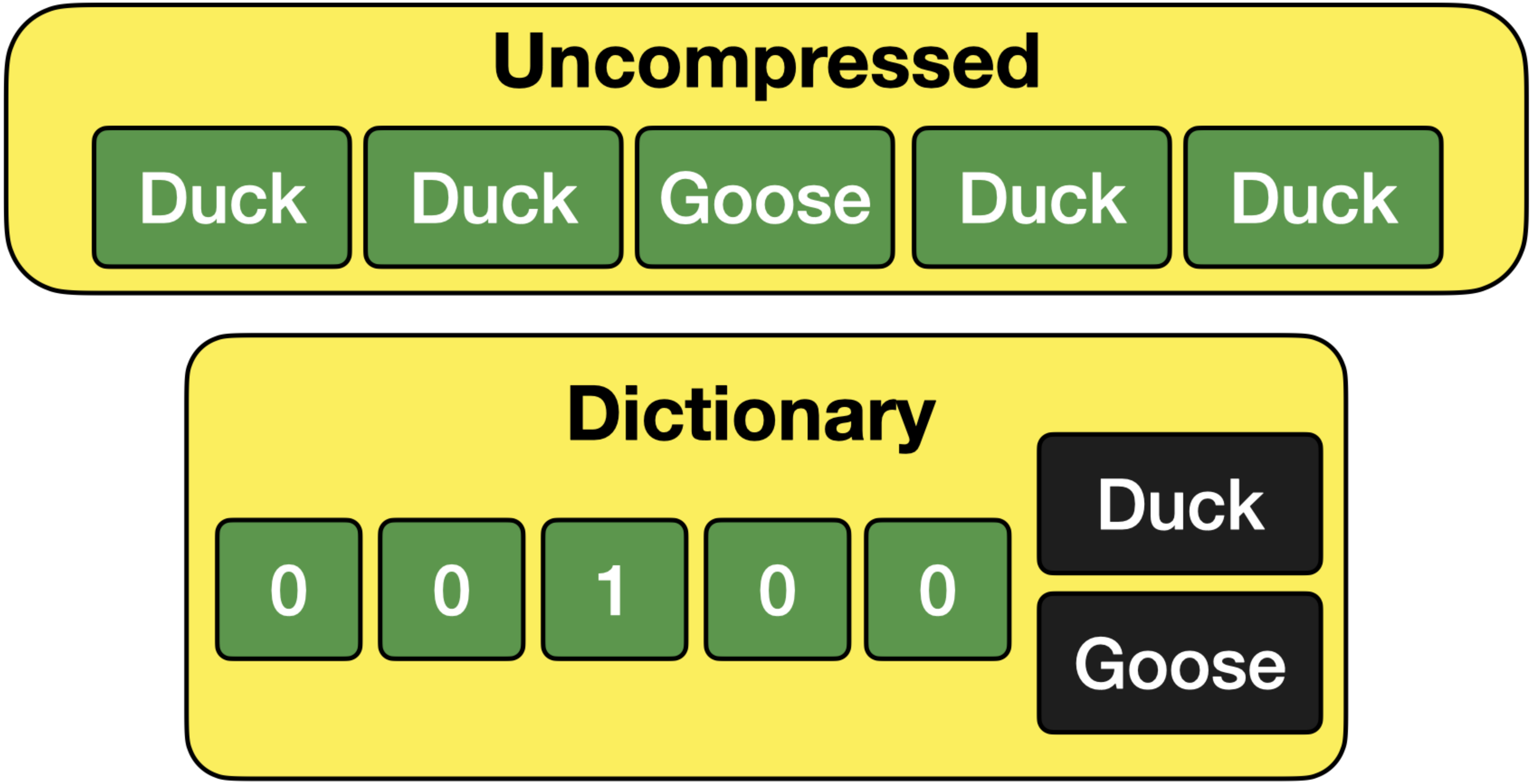
Source: https://duckdb.org/2022/10/28/lightweight-compression



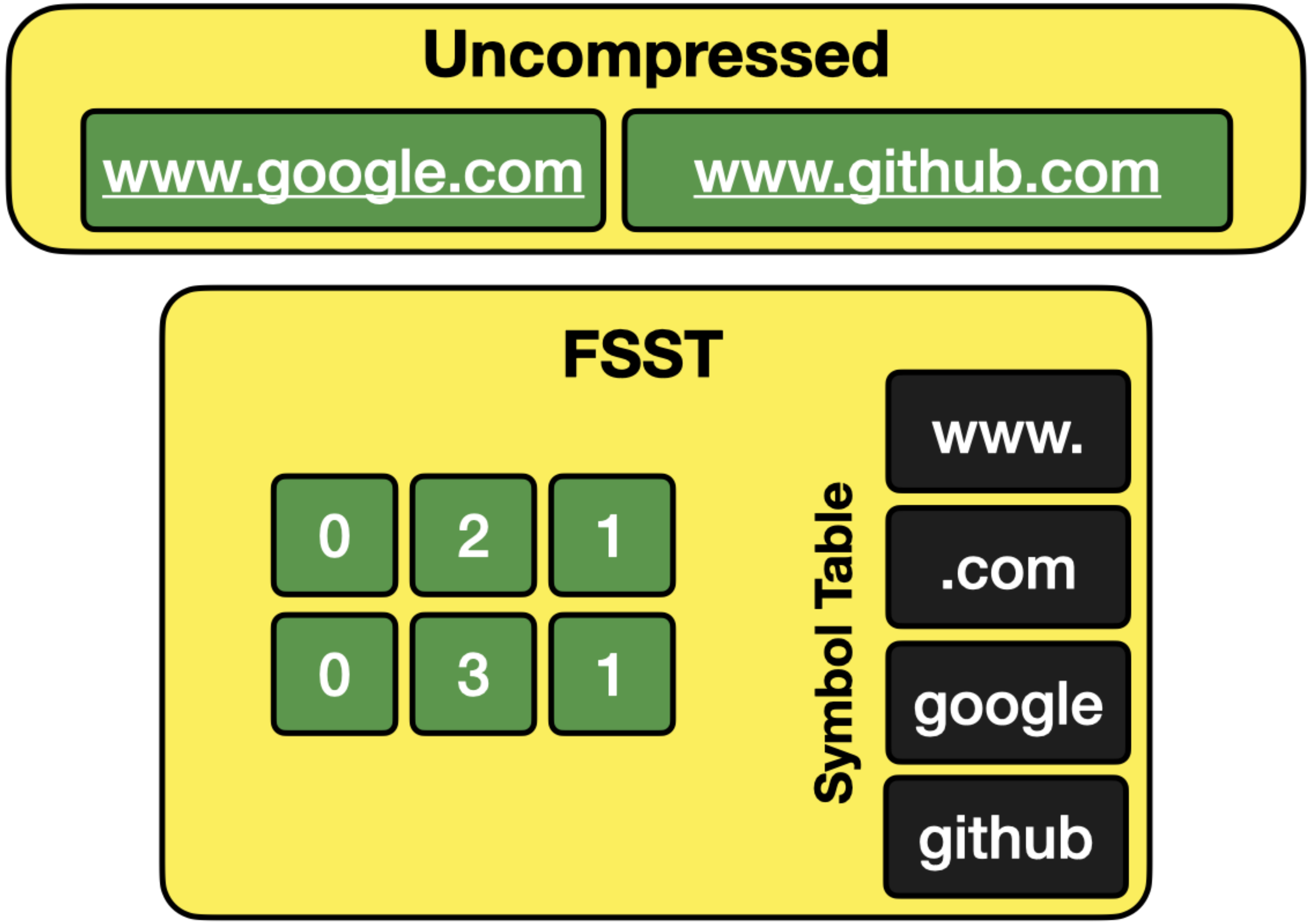
Source: https://duckdb.org/2022/10/28/lightweight-compression



Source: https://duckdb.org/2022/10/28/lightweight-compression



Source: https://duckdb.org/2022/10/28/lightweight-compression



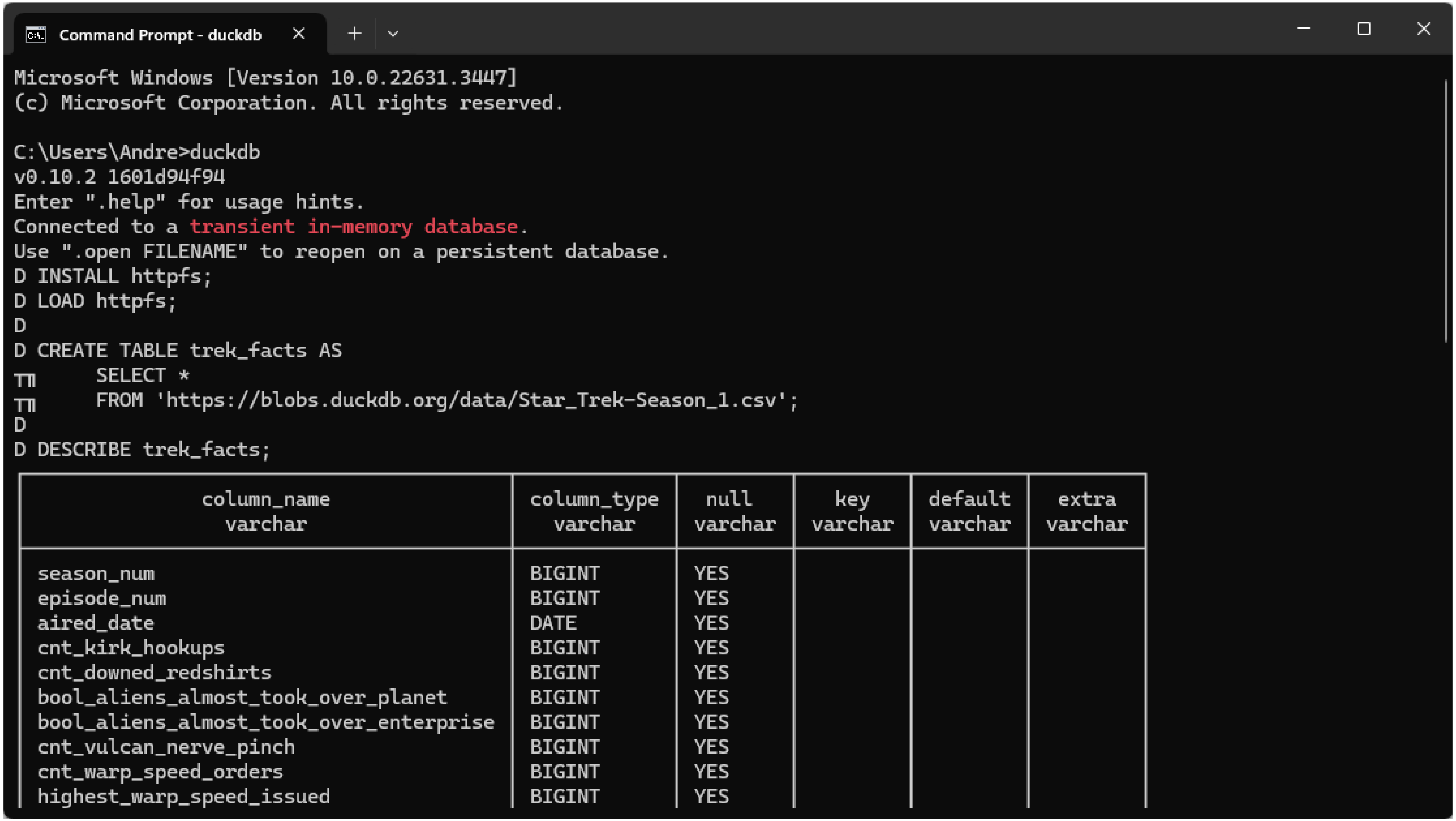
Source: https://duckdb.org/2022/10/28/lightweight-compression

- Chimp, based on Gorilla compression for floating points
- Patas takes that even further
- Compression does not apply to in-memory databases!!

Compression info via pragma_storage_info('<tablename>') function

row_group_id	***column_name***	***column_id***	***segment_type***	***count***	***compression***
4	**extra**	**13**	**FLOAT**	**65536**	**Chimp**
20	**tip_amount**	**15**	**FLOAT**	**65536**	**Chimp**
26	**pickup_latitude**	**6**	**VALIDITY**	**65536**	**Constant**
46	**tolls_amount**	**16**	**FLOAT**	**65536**	**RLE**
73	**store_and_fwd_flag**	**8**	**VALIDITY**	**65536**	**Uncompressed**
96	**total_amount**	**17**	**VALIDITY**	**65536**	**Constant**
111	**total_amount**	**17**	**VALIDITY**	**65536**	**Constant**
141	**pickup_at**	**1**	**TIMESTAMP**	**52224**	**BitPacking**
201	**pickup_longitude**	**5**	**VALIDITY**	**65536**	**Constant**
209	**passenger_count**	**3**	**TINYINT**	**65536**	**BitPacking**

Designed by Freepik



```
Microsoft Windows [Version 10.0.22631.3447]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Andre>duckdb
v0.10.2 1601d94f94
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
D INSTALL httpfs;
D LOAD httpfs;
D
D CREATE TABLE trek_facts AS
‣      SELECT *
‣      FROM 'https://blobs.duckdb.org/data/Star_Trek-Season_1.csv';
D
D DESCRIBE trek_facts;
┌────────────────────────────────────────┬─────────────┬─────────┬─────────┬─────────┬─────────┐
│              column_name               │ column_type │  null   │   key   │ default │  extra  │
│                varchar                 │   varchar   │ varchar │ varchar │ varchar │ varchar │
├────────────────────────────────────────┼─────────────┼─────────┼─────────┼─────────┼─────────┤
│ season_num                             │ BIGINT      │ YES     │         │         │         │
│ episode_num                            │ BIGINT      │ YES     │         │         │         │
│ aired_date                             │ DATE        │ YES     │         │         │         │
│ cnt_kirk_hookups                       │ BIGINT      │ YES     │         │         │         │
│ cnt_downed_redshirts                   │ BIGINT      │ YES     │         │         │         │
│ bool_aliens_almost_took_over_planet    │ BIGINT      │ YES     │         │         │         │
│ bool_aliens_almost_took_over_enterprise│ BIGINT      │ YES     │         │         │         │
│ cnt_vulcan_nerve_pinch                 │ BIGINT      │ YES     │         │         │         │
│ cnt_warp_speed_orders                  │ BIGINT      │ YES     │         │         │         │
│ highest_warp_speed_issued              │ BIGINT      │ YES     │         │         │         │
```

```sql
SELECT
        x_wing,
        proton_torpedoes,
        --targeting_computer
FROM luke_whats_wrong
GROUP BY x_wing, proton_torpedoes,
;
```

```sql
SELECT
	'Use the Force, Luke'[:13] AS sliced_quote_1,
	substr('I am your father', 1, 4) AS sliced_quote_2,
	substring('Obi-Wan Kenobi, you''re my only hope',17,100) AS
sliced_quote_3;
```

```sql
SELECT
	*
FROM (
	SELECT
		s1.tie_fighter,
		s2.tie_fighter
FROM squadron_one s1
JOIN squadron_two s2 ON 1=1 ) theyre_coming_in_too_fast;
```

tie_fighter	tie_fighter:1
green_one	green_two

```sql
SELECT
        'These are the voyages of the starship Enterprise...' AS intro,
        instr(intro, 'starship') AS starship_loc,
        substr(intro, starship_loc + len('starship') + 1) AS trimmed_intro;
```

```sql
SELECT * EXCLUDE (jar_jar_binks, midichlorians) FROM star_wars;
```

```sql
SELECT
        sw.* EXCLUDE (jar_jar_binks, midichlorians),
        ff.* EXCLUDE cancellation
FROM star_wars sw, firefly ff;
```

```sql
SELECT * REPLACE (
        movie_count+3 AS movie_count,
        show_count*1000 AS show_count
        )
FROM star_wars_owned_by_disney;
```

```sql
SELECT
        systems,
        planets,
        cities,
        cantinas,
        SUM(scum + villainy) AS total_scum_and_villainy
FROM star_wars_locations
GROUP BY ALL;

SELECT
        * EXCLUDE (cantinas, booths, scum, villainy),
        SUM(scum + villainy) AS total_scum_and_villainy
FROM star_wars_locations
GROUP BY ALL;
```

```sql
SELECT
        age,
        sum(civility) AS total_civility
FROM star_wars_universe
GROUP BY ALL
ORDER BY ALL;
```

It also supports DESC
And you can say NULLS FIRST or NULLS LAST

```sql
SELECT
        episode_num,
        COLUMNS('.*warp.*')
FROM trek_facts;
```

episode_num	cnt_warp_speed_orders	highest_warp_speed_issued
0	1	1
1	0	0
2	1	1
3	1	0
…	…	…
27	1	1
28	0	0
29	2	8

```sql
SELECT
        MAX(COLUMNS('.*warp.*'))
FROM trek_facts;
```

max(trek_facts.cnt_warp_speed_orders)	max(trek_facts.highest_warp_speed_issued)
5	8

```sql
SELECT
        episode_num,
        COLUMNS('.*warp.*’)
FROM trek_facts
WHERE COLUMNS('.*warp.*') >= 2;
```

episode_num	cnt_warp_speed_orders	highest_warp_speed_issued
14	3	7
17	2	7
18	2	8
29	2	8

```sql
SELECT
        MAX(COLUMNS(* EXCLUDE season_num))
FROM trek_facts;
```

max(trek_facts. episode_num)	max(trek_facts. aired_date)	max(trek_facts. cnt_kirk_hookups)	...	max(trek_facts. bool_enterprise_saved_the_day)
29	1967-04-13	2	...	1

```
SELECT
        MAX(COLUMNS(* REPLACE aired_date::timestamp AS aired_date))
FROM trek_facts;
```

max(trek_facts.season_num)	max(trek_facts.episode_num)	max(aired_date := CAST(aired_date AS TIMESTAMP))	...	max(trek_facts.bool_enterprise_saved_the_day)
1	29	1967-04-13 00:00:00	...	1

```
SELECT
        episode_num,
        COLUMNS(col -> col LIKE '%warp%’)
FROM trek_facts
WHERE COLUMNS(col -> col LIKE '%warp%') >= 2;
```

episode_num	cnt_warp_speed_orders	highest_warp_speed_issued
14	3	7
17	2	7
18	2	8
29	2	8

```sql
FROM my_table SELECT my_column;
```

```sql
FROM my_table;
```

```sql
COPY (FROM trek_facts) TO
'phaser_filled_facts.parquet';
```

(OLD)

```
SELECT
        concat(
                list_aggr(
                        string_split(
                                UPPER('Make it stop’),
                        ' ‘),
                'string_agg','.’),
        '.') AS oof;
```

```
oof
MAKE.IT.STOP.
```

```
SELECT
        ('Make it so’)
                .UPPER()
                .string_split(' ‘)
                .list_aggr('string_agg','.’)
                .concat('.') AS im_not_messing_around_number_one;
```

```
im_not_messing_around_number_one
--------------------------------
MAKE.IT.SO.
```

```
CREATE TABLE proverbs AS
SELECT
        'Revenge is a dish best served cold' AS klingon_proverb
UNION ALL BY NAME
SELECT
        'You will be assimilated' AS borg_proverb,
        'If winning is not important, why keep score?' AS klingon_proverb;
FROM proverbs;
```

klingon_proverb	borg_proverb
Revenge is a dish best served cold	NULL
If winning is not important, why keep score?	You will be assimilated

```sql
INSERT INTO proverbs BY NAME
        SELECT 'Resistance is futile' AS borg_proverb;
```

klingon_proverb	borg_proverb
Revenge is a dish best served cold	NULL
If winning is not important, why keep score?	You will be assimilated
NULL	Resistance is futile

item	year	count
phasers	2155	1035
phasers	2156	25039
phasers	2157	95000
photon torpedoes	2155	255
photon torpedoes	2156	17899
photon torpedoes	2157	87492

```
PIVOT purchases ON year
USING SUM(count)
GROUP BY item;
```

item	2155	2156	2157
phasers	1035	25039	95000
photon torpedoes	255	17899	87492

item	2155	2156	2157
phasers	1035	25039	95000
photon torpedoes	255	17899	87492

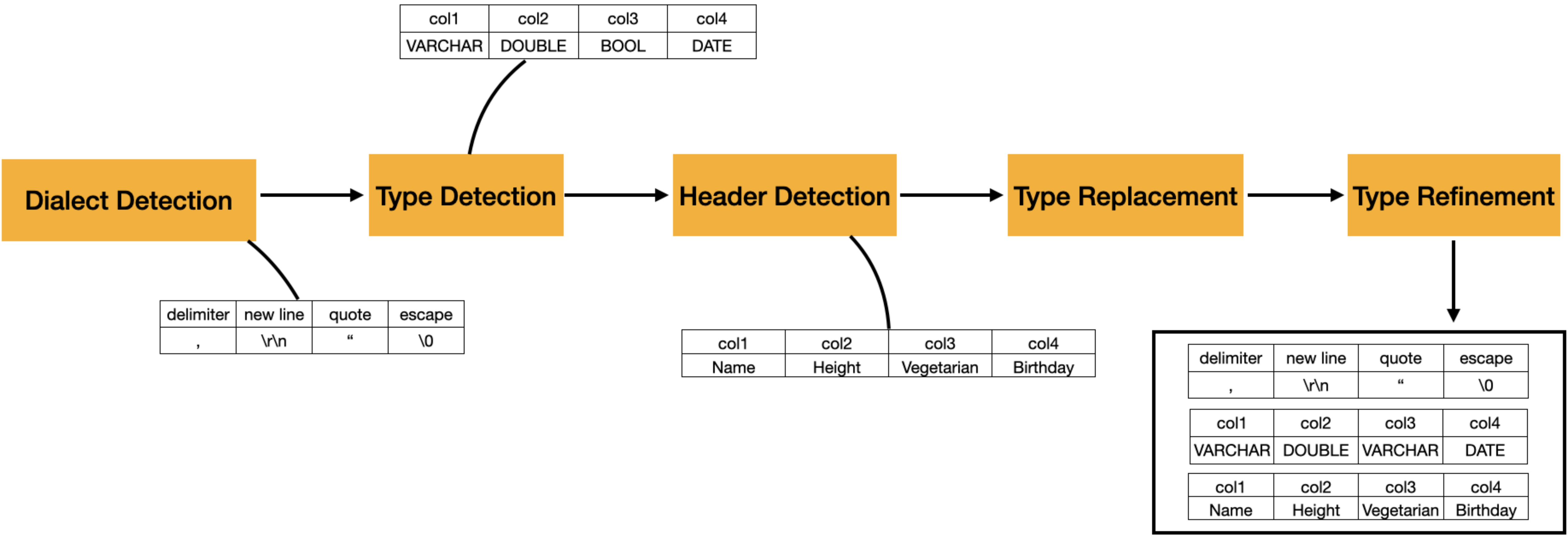
```
UNPIVOT pivoted_purchases
    ON COLUMNS(* EXCLUDE item)
    INTO
        NAME year
        VALUE count;
```

item	year	count
phasers	2155	1035
phasers	2156	25039
phasers	2157	95000
photon torpedoes	2155	255
photon torpedoes	2156	17899
photon torpedoes	2157	87492

And many many more!

- Number generator
- As-of joins
- Tpch data generator
- Json structs

CSV



col1	col2	col3	col4
VARCHAR	DOUBLE	BOOL	DATE

delimiter	new line	quote	escape
,	\r\n	“	\0

col1	col2	col3	col4
Name	Height	Vegetarian	Birthday

delimiter	new line	quote	escape
,	\r\n	“	\0

col1	col2	col3	col4
VARCHAR	DOUBLE	VARCHAR	DATE

col1	col2	col3	col4
Name	Height	Vegetarian	Birthday

```
Name, Height, Vegetarian, Birthday
"Pedro", 1.73, False, 30-07-92
... imagine 2048 consistent rows ...
"Mark", 1.72, N/A, 20-09-92
```

```
Conversion Error: CSV Error on Line: 5648
Original Line: Pedro,The 90s
Error when converting column "birth_date". date field value out of range: "The 90s", expected forma

Column date is being converted as type DATE
This type was auto-detected from the CSV file.
Possible solutions:
* Override the type for this column manually by setting the type explicitly, e.g. types={'birth_dat
* Set the sample size to a larger value to enable the auto-detection to scan more values, e.g. samp
* Use a COPY statement to automatically derive types from an existing table.

  file= people.csv
  delimiter = , (Auto-Detected)
  quote = " (Auto-Detected)
  escape = " (Auto-Detected)
  new_line = \r\n (Auto-Detected)
  header = true (Auto-Detected)
  skip_rows = 0 (Auto-Detected)
  date_format = (DD-MM-YYYY) (Auto-Detected)
  timestamp_format =  (Auto-Detected)
  null_padding=0
  sample_size=20480
  ignore_errors=false
  all_varchar=0
```

Name	Description	Type	Default
store_rejects	If set to true, any errors in the file will be skipped and stored in the default rejects temporary tables.	BOOLEAN	False
rejects_scan	Name of a temporary table where the information of the scan information of faulty CSV file are stored.	VARCHAR	reject_scans
rejects_table	Name of a temporary table where the information of the faulty lines of a CSV file are stored.	VARCHAR	reject_errors
rejects_limit	Upper limit on the number of faulty records from a CSV file that will be recorded in the rejects table. 0 is used when no limit should be applied.	BIGINT	0

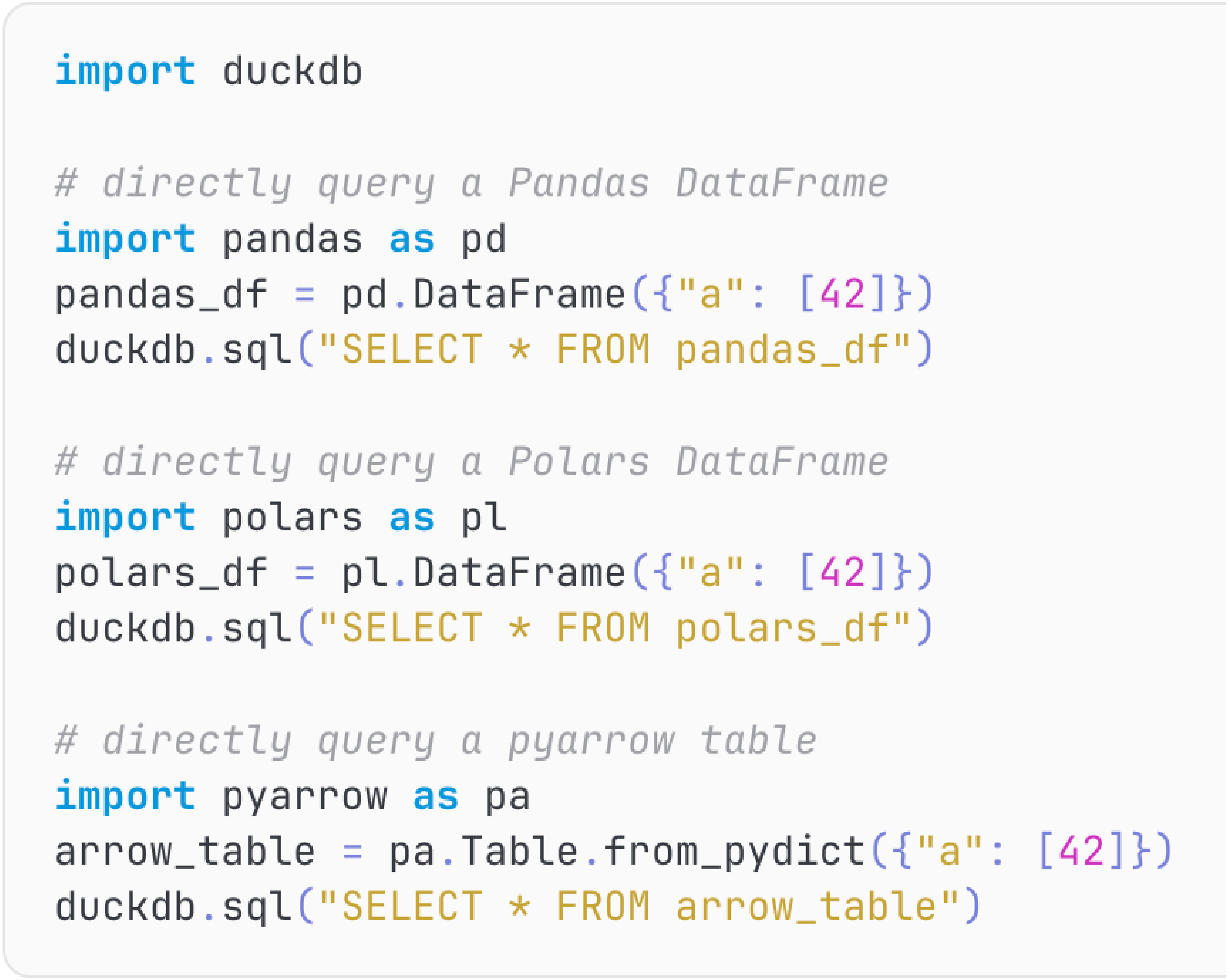
scan_id	file_id	file_path	delimiter	quote	escape	newline_delimiter
5	0	faulty.csv	,	”	”	\n

line	line_byte_position	byte_position	column_idx	column_name	error_type	csv_line	error_message
2	10	23	2	age	CAST	Oogie Boogie, three	Error when converting column “age”. Could not convert string “ three” to ‘INTEGER’

python™

```python
import duckdb
duckdb.sql("SELECT 42").show()
```

```python
import duckdb
duckdb.read_csv("example.csv")
duckdb.read_parquet("example.parquet")
duckdb.read_json("example.json")

duckdb.sql("SELECT * FROM 'example.csv'")
duckdb.sql("SELECT * FROM 'example.parquet'")
duckdb.sql("SELECT * FROM 'example.json'")
```



```python
import duckdb

# directly query a Pandas DataFrame
import pandas as pd
pandas_df = pd.DataFrame({"a": [42]})
duckdb.sql("SELECT * FROM pandas_df")

# directly query a Polars DataFrame
import polars as pl
polars_df = pl.DataFrame({"a": [42]})
duckdb.sql("SELECT * FROM polars_df")

# directly query a pyarrow table
import pyarrow as pa
arrow_table = pa.Table.from_pydict({"a": [42]})
duckdb.sql("SELECT * FROM arrow_table")
```

```python
import duckdb
duckdb.sql("SELECT 42").fetchall()   # Python objects
duckdb.sql("SELECT 42").df()         # Pandas DataFrame
duckdb.sql("SELECT 42").pl()         # Polars DataFrame
duckdb.sql("SELECT 42").arrow()      # Arrow Table
duckdb.sql("SELECT 42").fetchnumpy() # NumPy Arrays
```

```python
import duckdb
duckdb.sql("SELECT 42").write_parquet("out.parquet") # Write to a Parquet file
duckdb.sql("SELECT 42").write_csv("out.csv")         # Write to a CSV file
duckdb.sql("COPY (SELECT 42) TO 'out.parquet'")      # Copy to a Parquet file
```

dbt