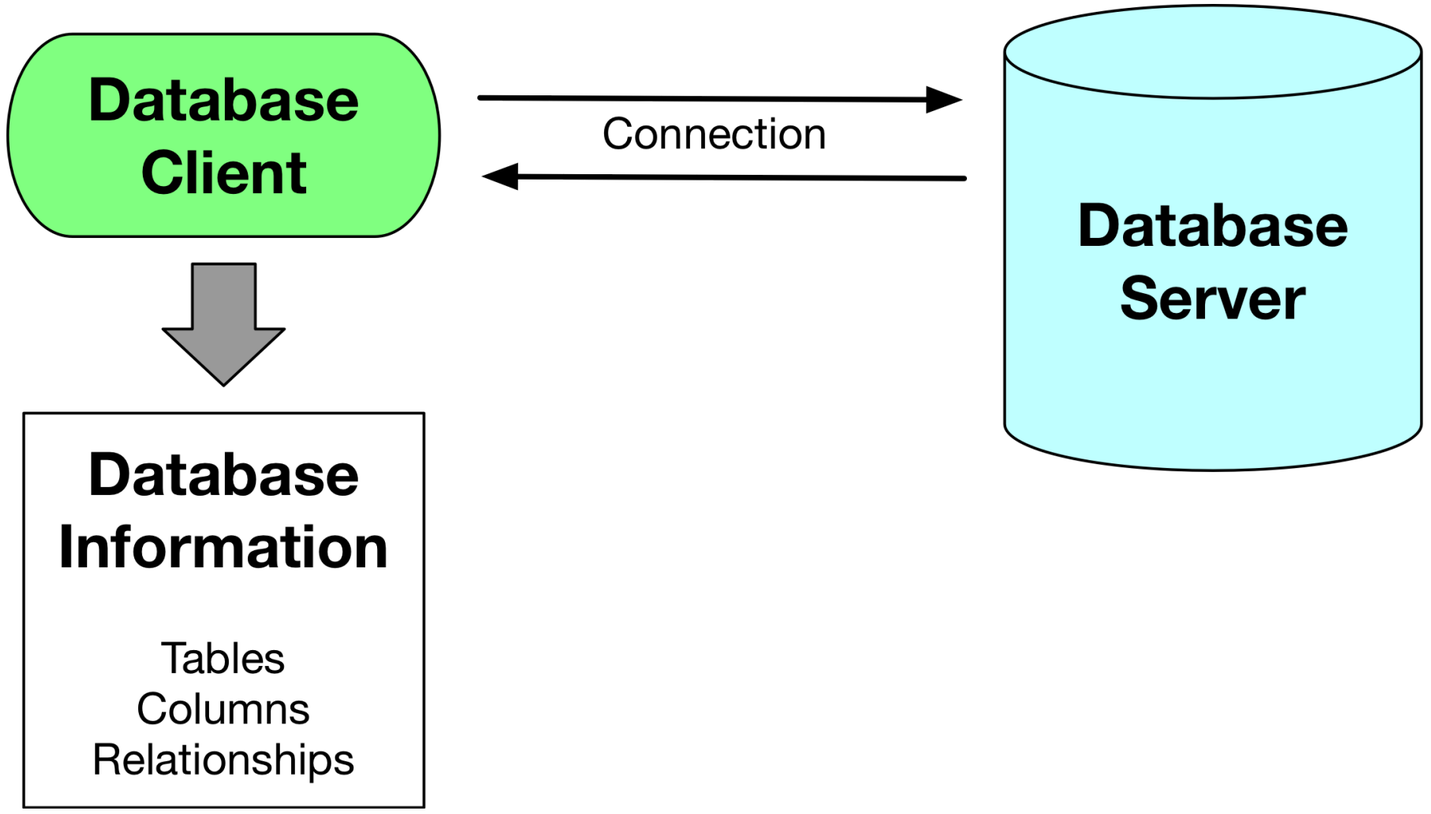
SQL FOR EXPLORATORY DATA ANALYSIS

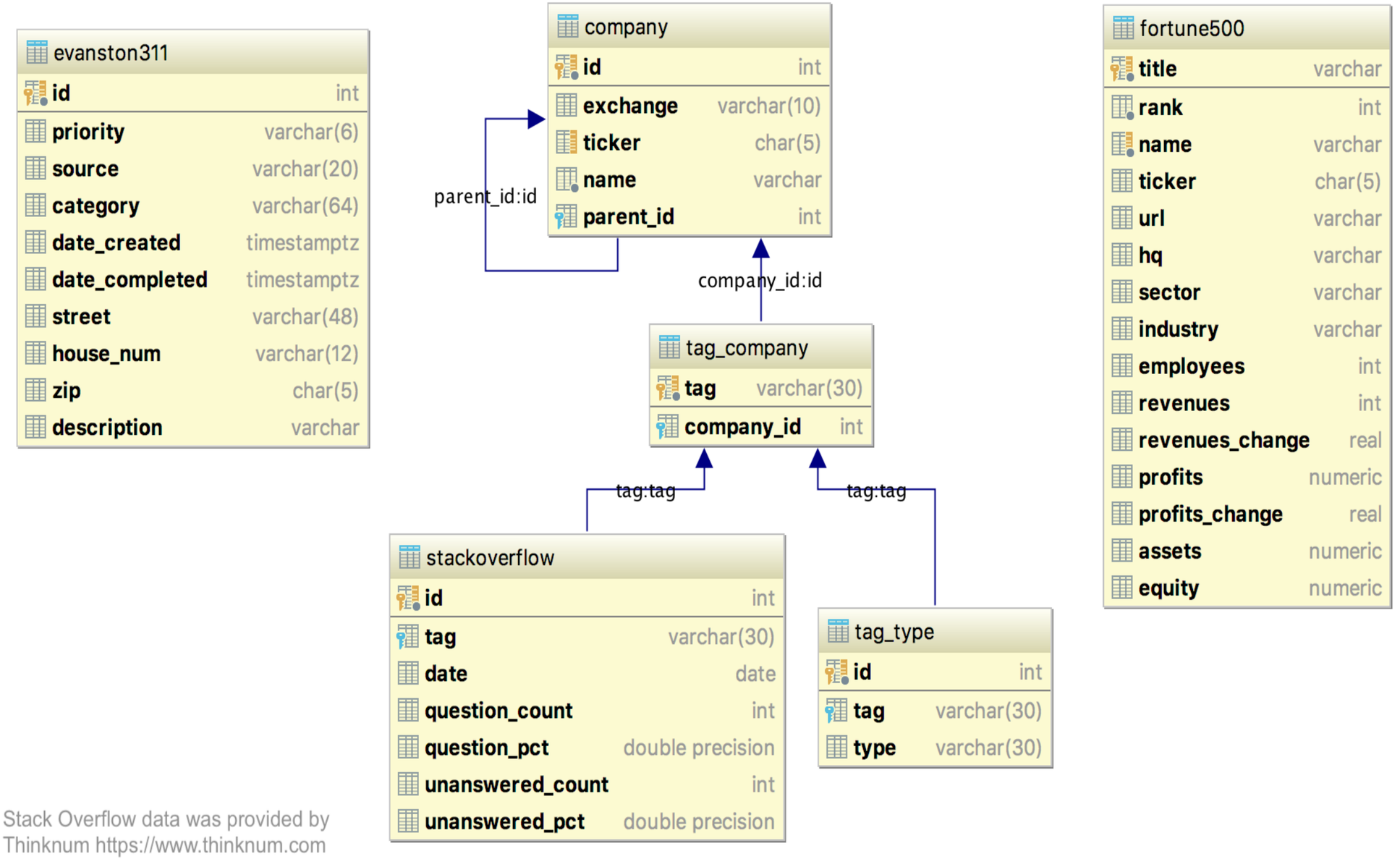
What's in the database?

Christina Maimone
Data Scientist

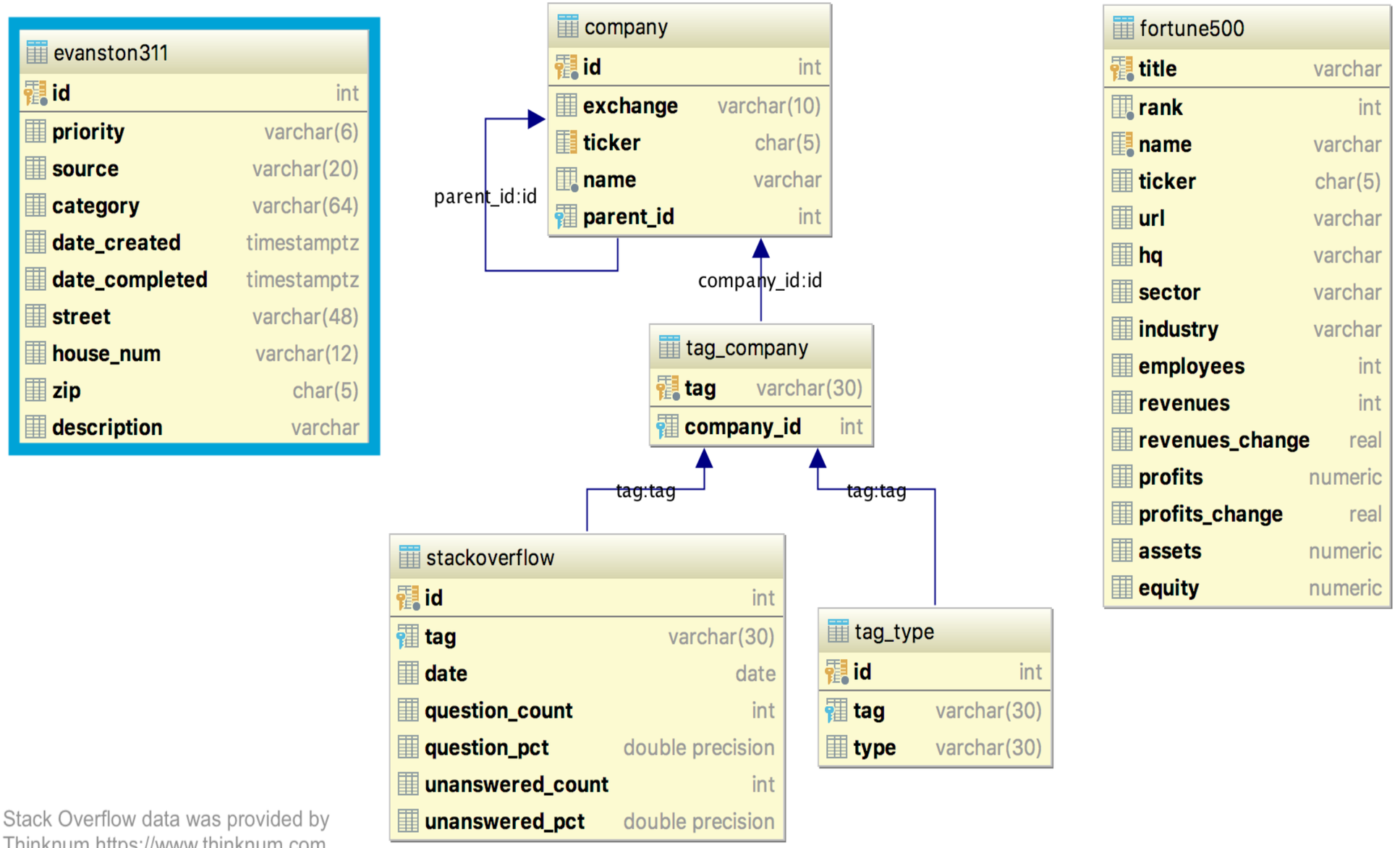
PostgreSQL

Database client

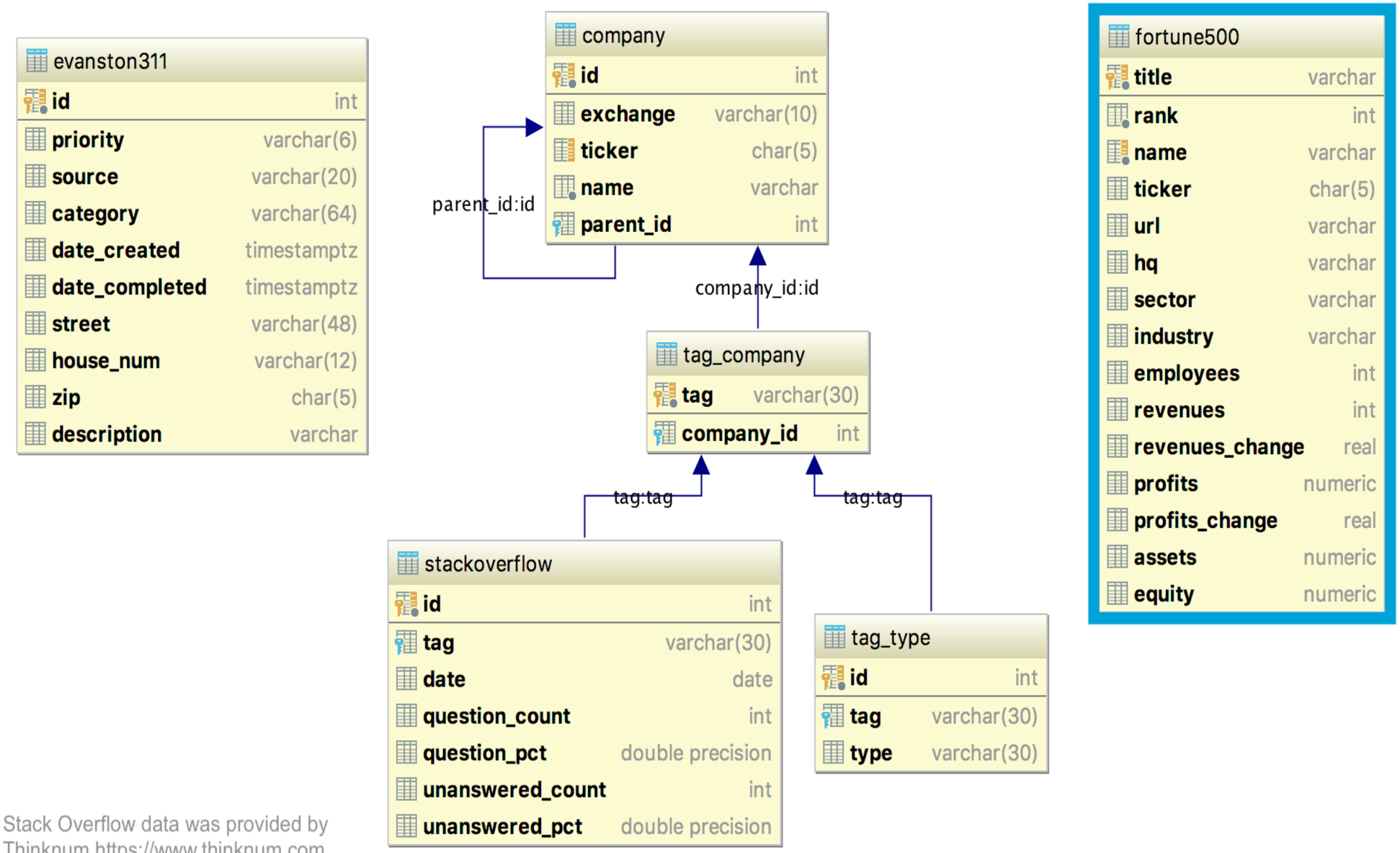




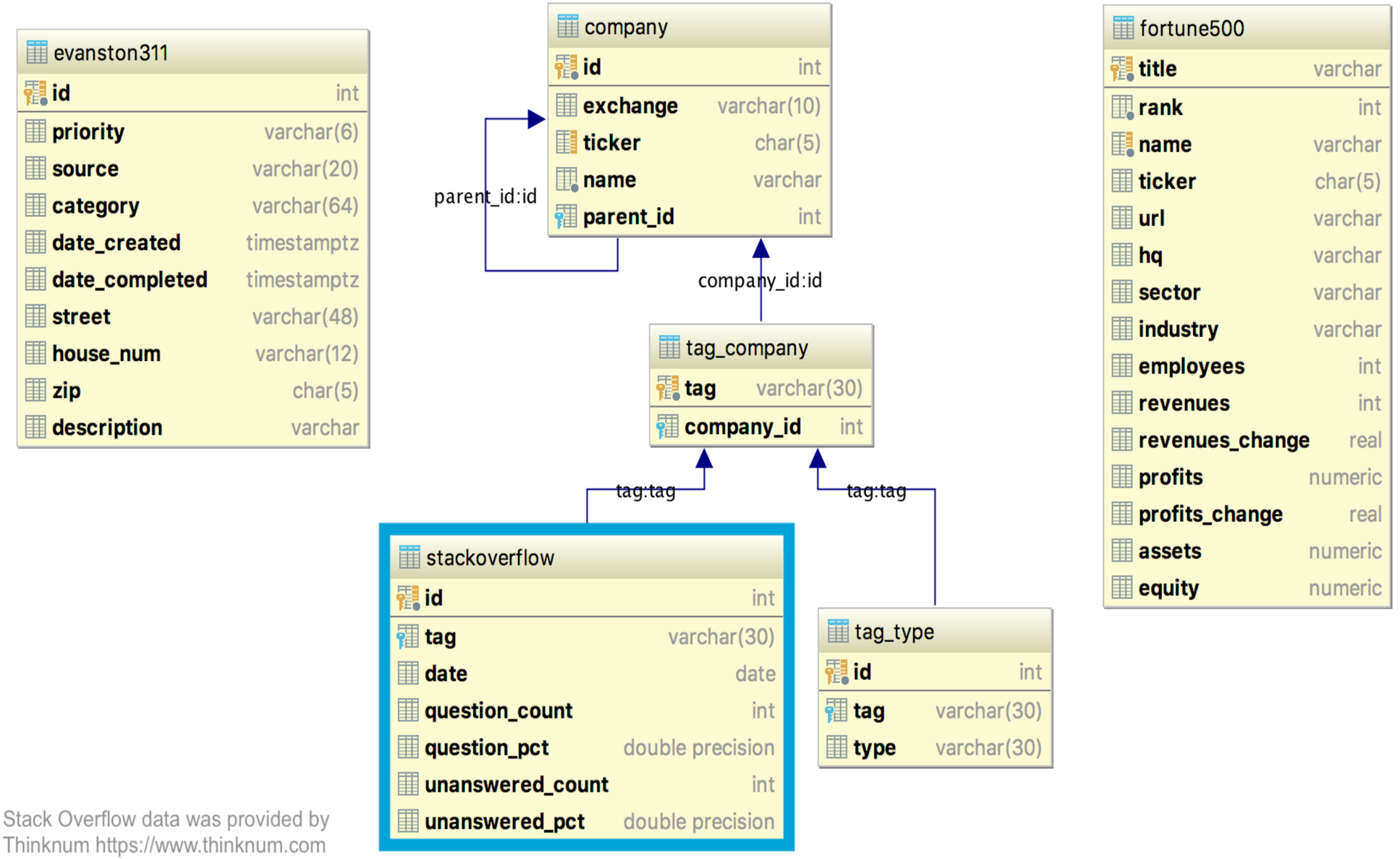
Stack Overflow data was provided by Thinknum https://www.thinknum.com



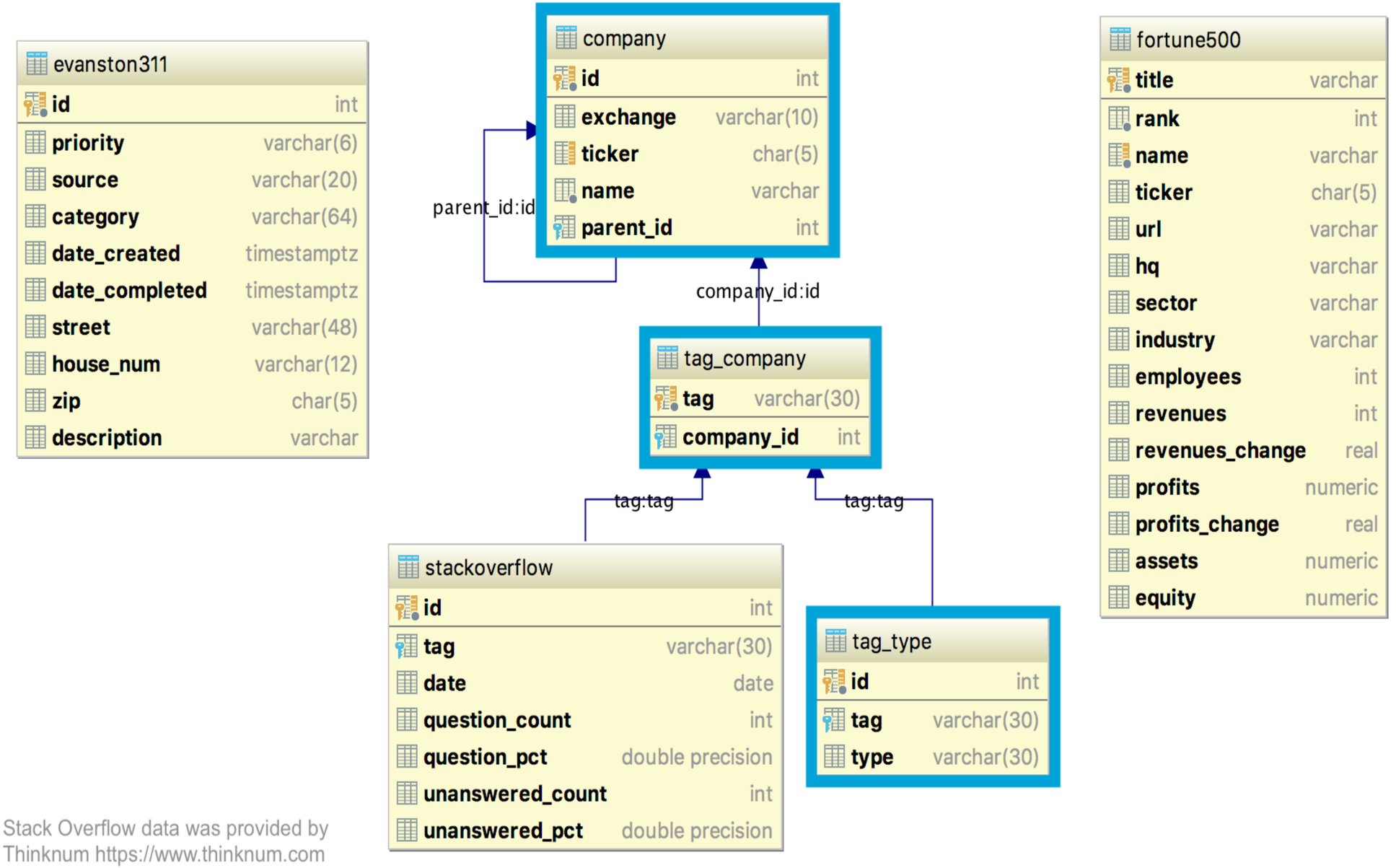
Stack Overflow data was provided by Thinknum https://www.thinknum.com



Stack Overflow data was provided by Thinknum https://www.thinknum.com



Stack Overflow data was provided by Thinknum https://www.thinknum.com



Stack Overflow data was provided by Thinknum https://www.thinknum.com

Select a few rows

```
SELECT *
  FROM company
 LIMIT 5;
```

```
 id | exchange | ticker |          name           | parent_id
----+----------+--------+-------------------------+-----------
  1 | nasdaq   | PYPL   | PayPal Holdings, Inc. |
  2 | nasdaq   | AMZN   | Amazon.com, Inc.      |
  3 | nasdaq   | MSFT   | Microsoft Corporation |
  4 | nasdaq   | MDB    | MongoDB Inc.          |
  5 | nasdaq   | DBX    | Dropbox, Inc.         |
(5 rows)
```

A few reminders

Code	Note
`NULL`	missing

A few reminders

Code	Note
`NULL`	missing
`IS NULL`, `IS NOT NULL`	don't use `= NULL`

A few reminders

Code	Note
`NULL`	missing
`IS NULL`, `IS NOT NULL`	don't use `= NULL`
`count(*)`	number of rows

A few reminders

Code	Note
`NULL`	missing
`IS NULL`, `IS NOT NULL`	don't use `= NULL`
`count(*)`	number of rows
`count(column_name)`	number of non-`NULL` values

A few reminders

Code	Note
`NULL`	missing
`IS NULL`, `IS NOT NULL`	don't use `= NULL`
`count(*)`	number of rows
`count(column_name)`	number of non-`NULL` values
`count(DISTINCT column_name)`	number of different non-`NULL` values

A few reminders

Code	Note
`NULL`	missing
`IS NULL`, `IS NOT NULL`	don't use `= NULL`
`count(*)`	number of rows
`count(column_name)`	number of non-`NULL` values
`count(DISTINCT column_name)`	number of different non-`NULL` values
`SELECT DISTINCT column_name ...`	distinct values, including `NULL`

SQL FOR EXPLORATORY DATA ANALYSIS

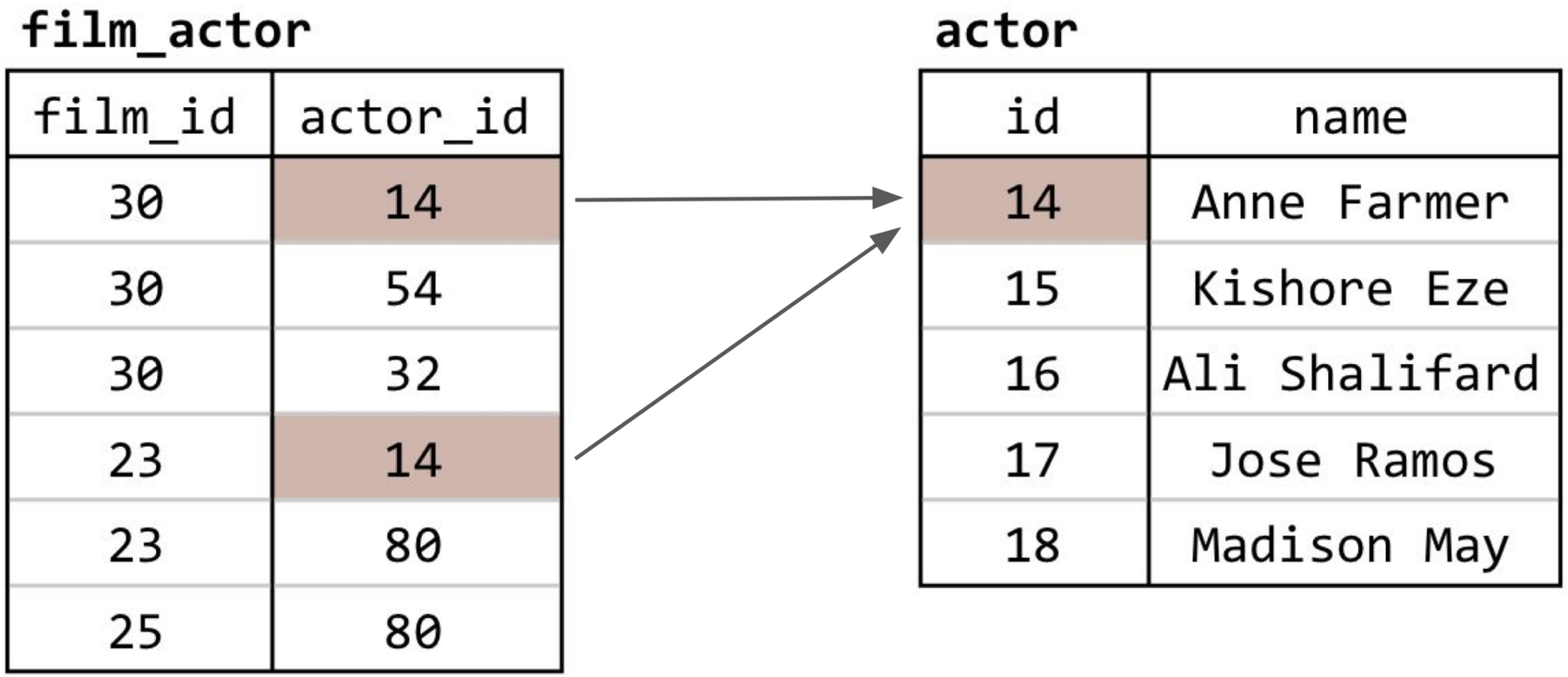
Let's start exploring

SQL FOR EXPLORATORY DATA ANALYSIS

The keys to the database

Christina Maimone
Data Scientist

Foreign keys

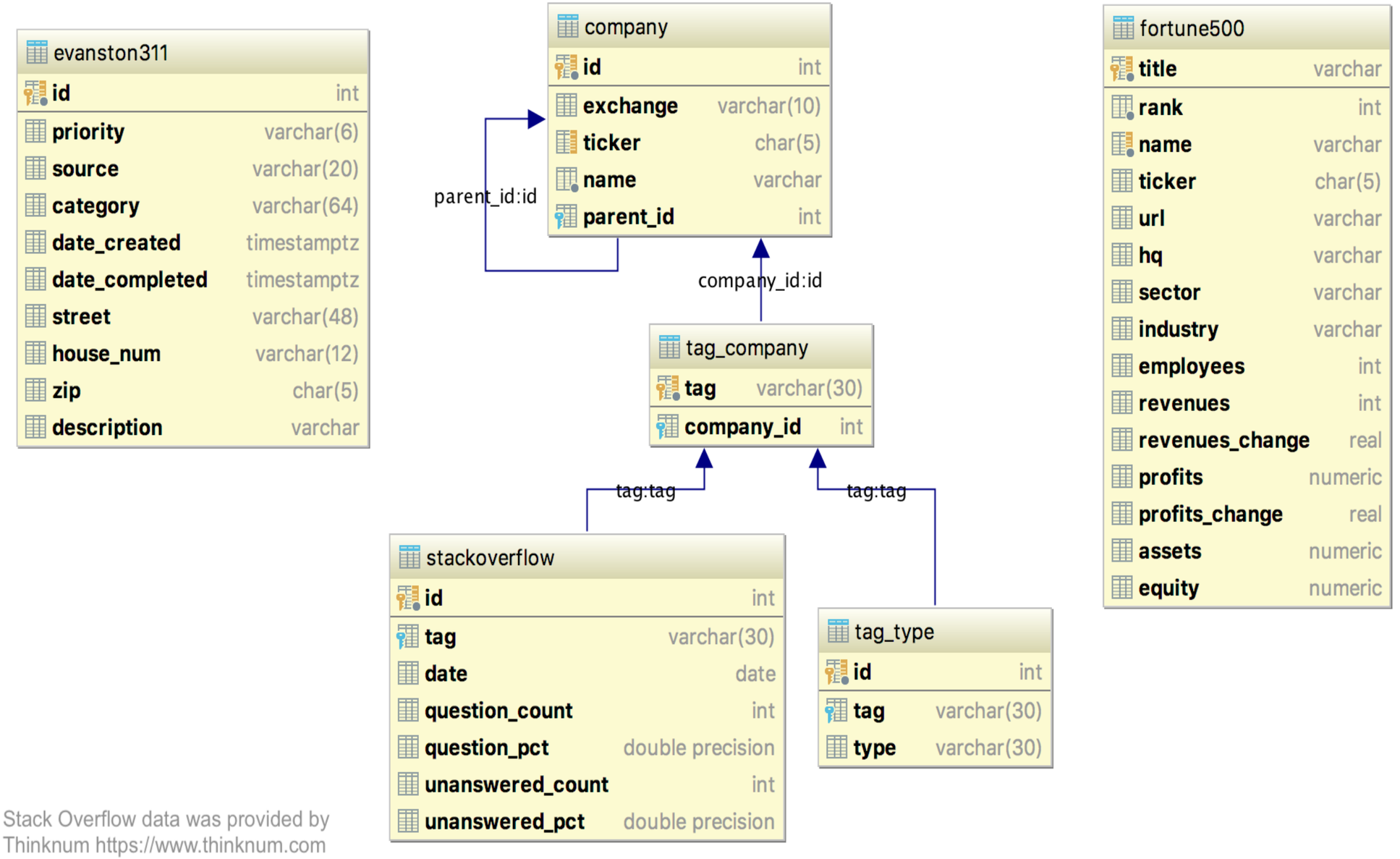


Foreign keys

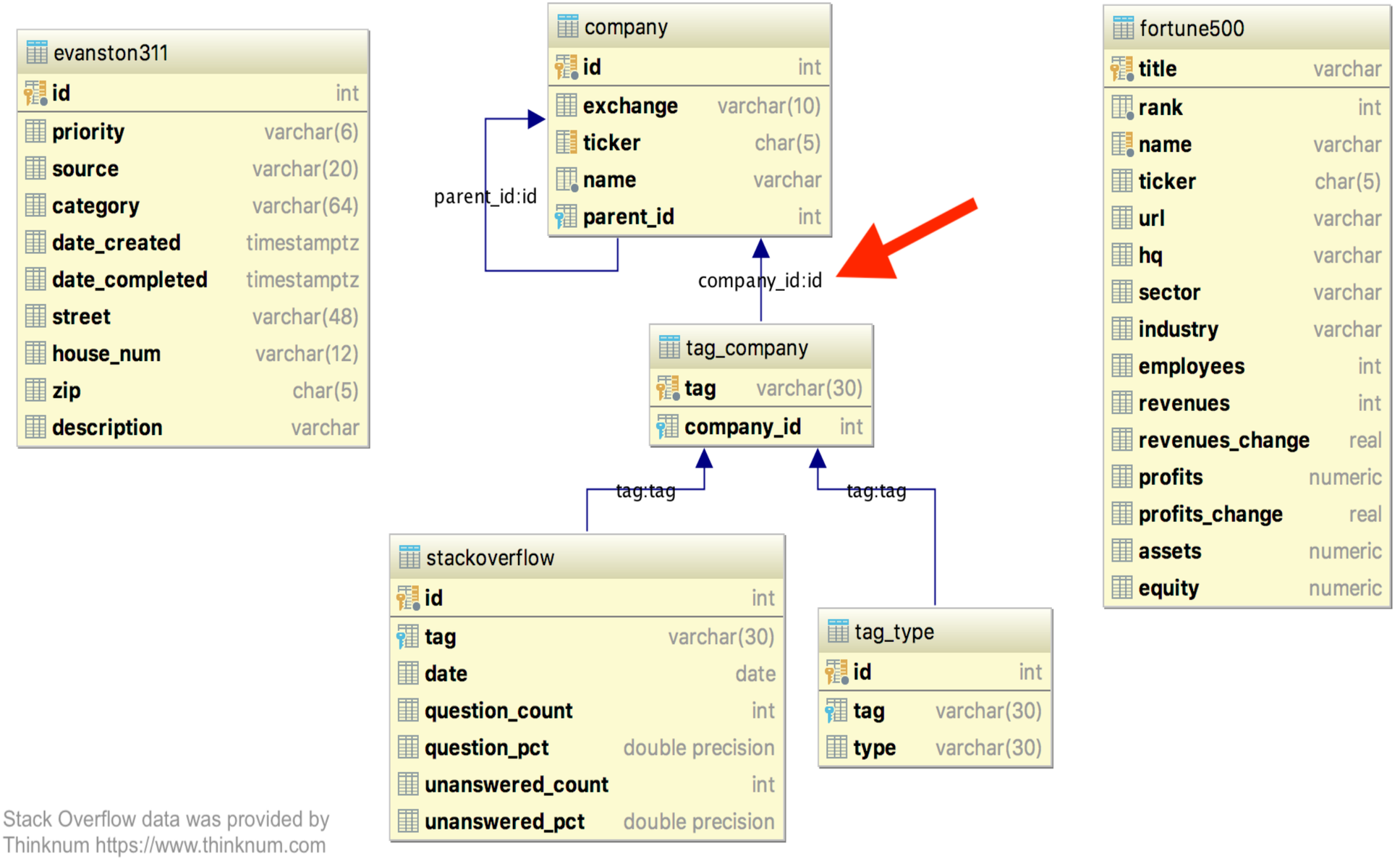
- Reference another row
  - In a different table or the same table
  - Via a unique ID

    >> Primary key column containing unique, non-NULL values

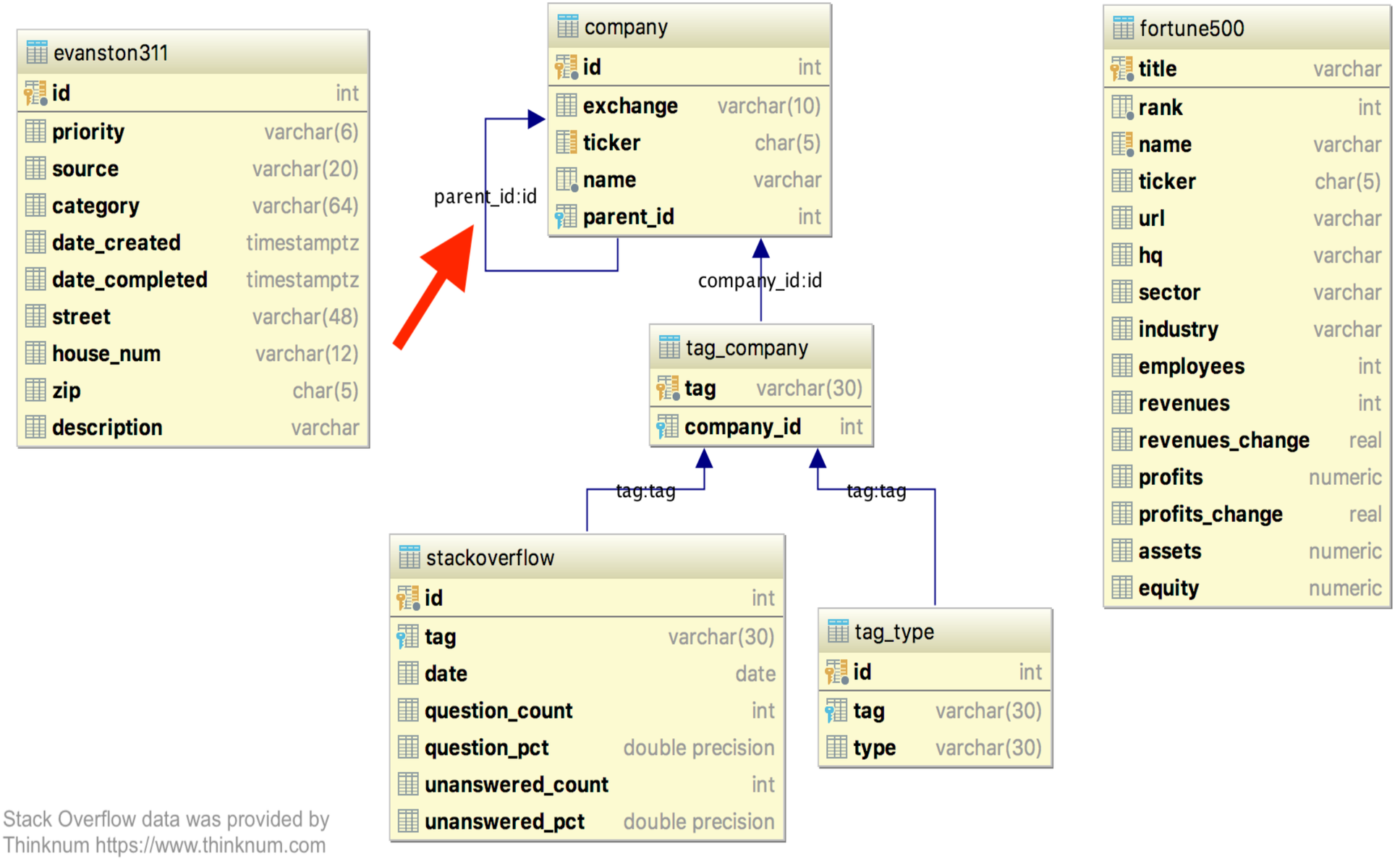
- Values restricted to values in referenced column OR `NULL`



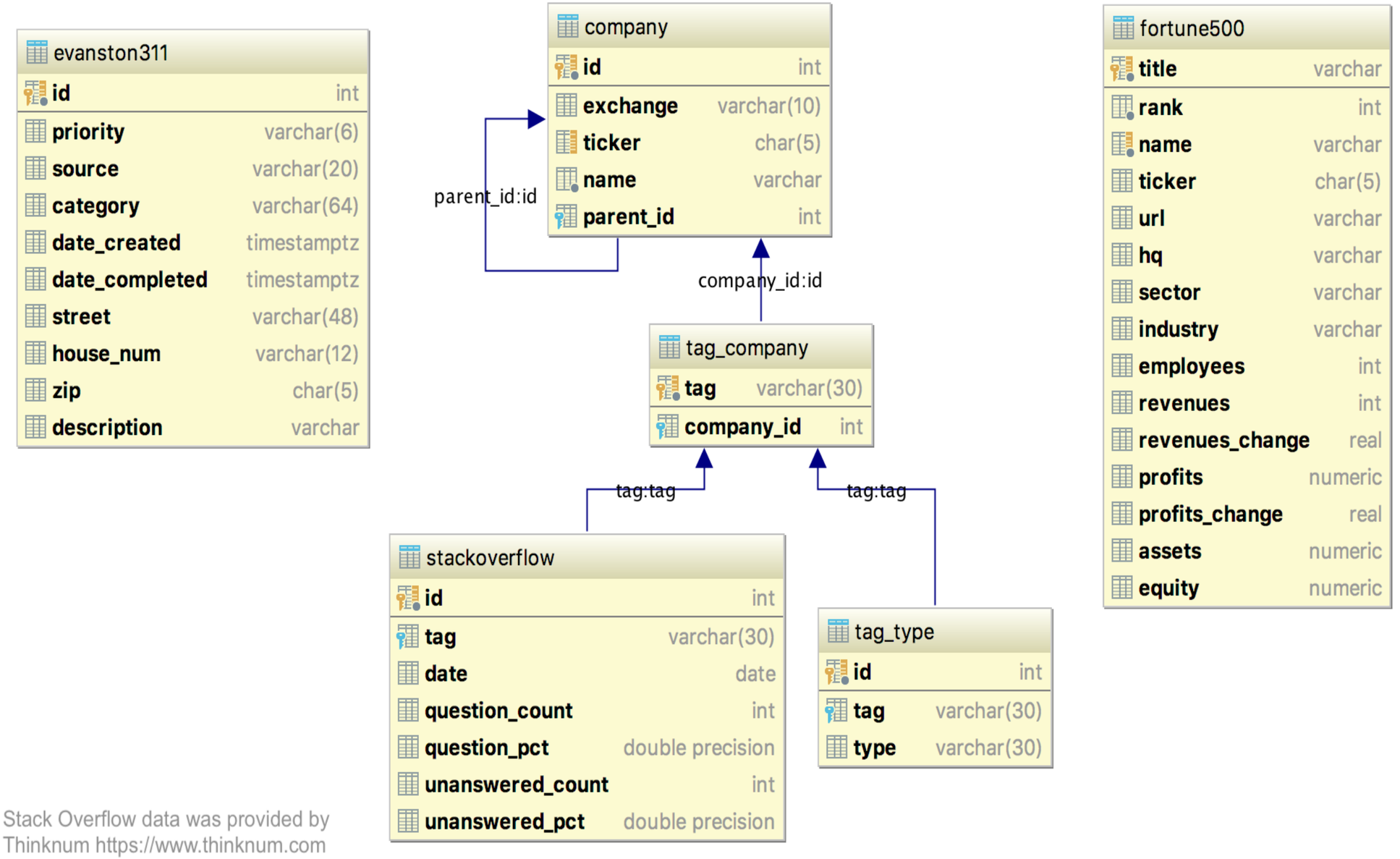
Stack Overflow data was provided by Thinknum https://www.thinknum.com



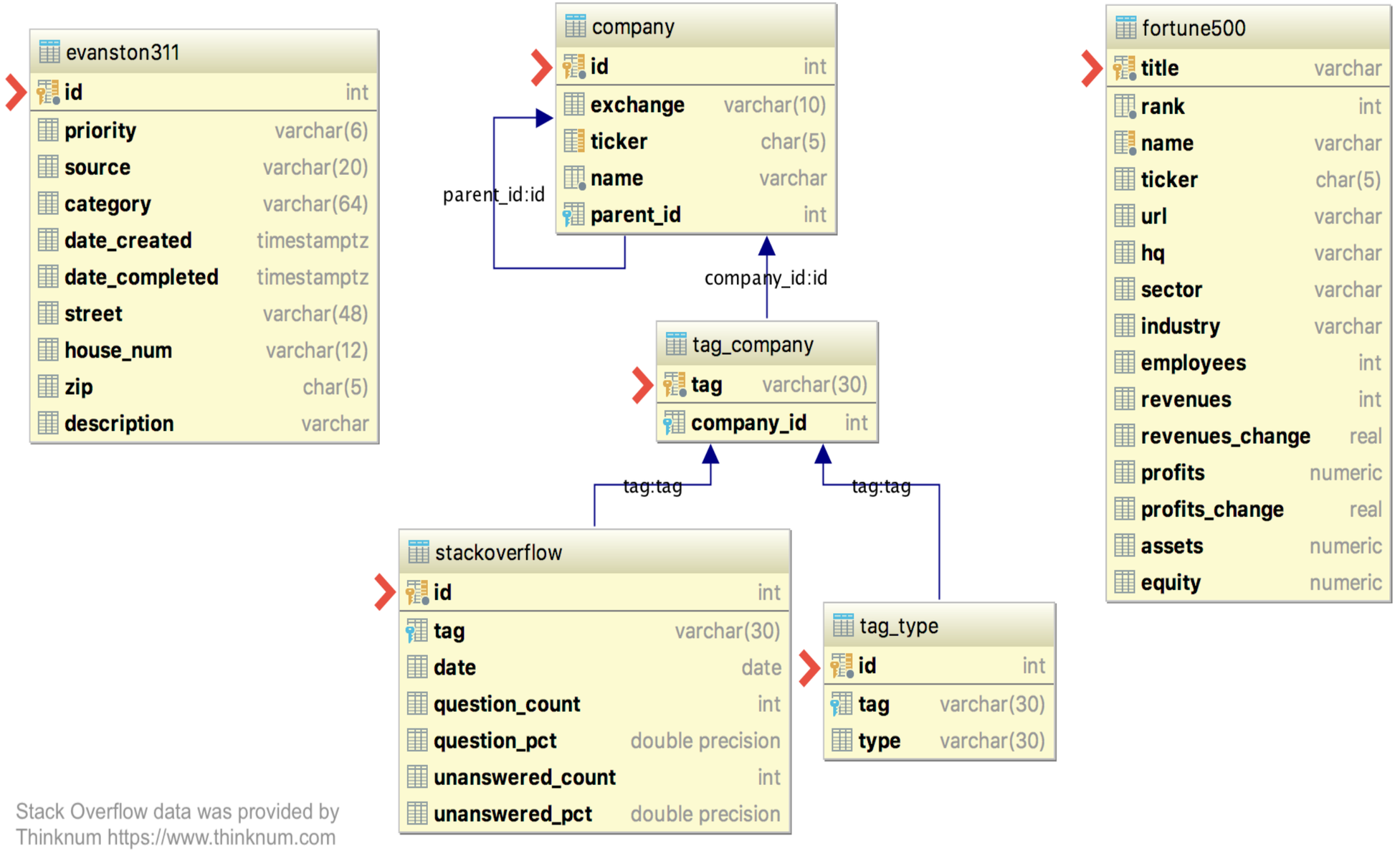
Stack Overflow data was provided by Thinknum https://www.thinknum.com



Stack Overflow data was provided by Thinknum https://www.thinknum.com



Stack Overflow data was provided by Thinknum https://www.thinknum.com



Stack Overflow data was provided by Thinknum https://www.thinknum.com

Coalesce function

```
coalesce(value_1, value_2 [, ...])
```

- Operates row by row
- Returns first non-`NULL` value

Coalesce function

```
SELECT *
  FROM prices;
```

```
 column_1 | column_2
----------+----------
          |       10
          |
       22 |
        3 |        4
```

```
SELECT coalesce(column_1, column_2)
  FROM prices;
```

```
 coalesce
----------
       10

       22
        3
```

SQL FOR EXPLORATORY DATA ANALYSIS

Time to keep exploring!

SQL FOR EXPLORATORY DATA ANALYSIS

Column Types and Constraints

Christina Maimone
Data Scientist

Column constraints

- **Foreign key**: value that exists in the referenced column, or NULL
- **Primary key**: unique, not `NULL`
- **Unique**: values must all be different except for `NULL`
- **Not null**: `NULL` not allowed: must have a value
- **Check constraints**: conditions on the values
  - `column1 > 0`
  - `columnA > columnB`

Data types

Common

- Numeric
- Character
- Date/Time
- Boolean

Special

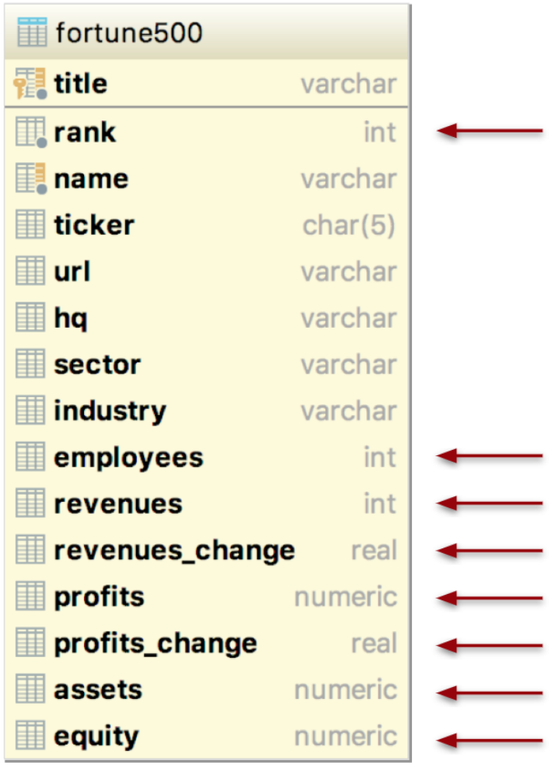
- Arrays
- Monetary
- Binary
- Geometric
- Network Address
- XML
- JSON
- and more!

Numeric types: PostgreSQL documentation

Table 8-2. Numeric Types

Name	Storage Size	Description	Range
`smallint`	2 bytes	small-range integer	-32768 to +32767
`integer`	4 bytes	typical choice for integer	-2147483648 to +2147483647
`bigint`	8 bytes	large-range integer	-9223372036854775808 to +9223372036854775807
`decimal`	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
`numeric`	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
`real`	4 bytes	variable-precision, inexact	6 decimal digits precision
`double precision`	8 bytes	variable-precision, inexact	15 decimal digits precision
`smallserial`	2 bytes	small autoincrementing integer	1 to 32767
`serial`	4 bytes	autoincrementing integer	1 to 2147483647
`bigserial`	8 bytes	large autoincrementing integer	1 to 9223372036854775807

Types in entity relationship diagrams



Casting with CAST()

Format

```
-- With the CAST function
SELECT CAST (value AS new_type);
```

Examples

```
-- Cast 3.7 as an integer
SELECT CAST (3.7 AS integer);
```

```
4
```

```
-- Cast a column called total as an integer
SELECT CAST (total AS integer)
  FROM prices;
```

Casting with ::

Format

```
-- With :: notation
SELECT value::new_type;
```

Examples

```
-- Cast 3.7 as an integer
SELECT 3.7::integer;
```

```
-- Cast a column called total as an integer
SELECT total::integer
  FROM prices;
```

SQL FOR EXPLORATORY DATA ANALYSIS

Time to practice!