dbt-score

continuous integration for dbt metadata

EuroPython 2025

🔗 dbt-score.picnic.tech

Matthieu
Caneill

🇫🇷 🥐

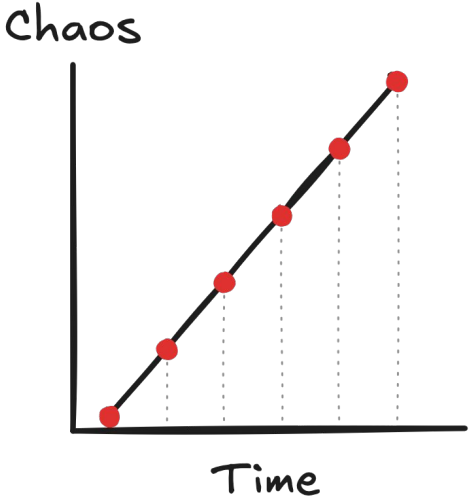
Picnic
Online
supermarket

🇳🇱🇩🇪🇫🇷

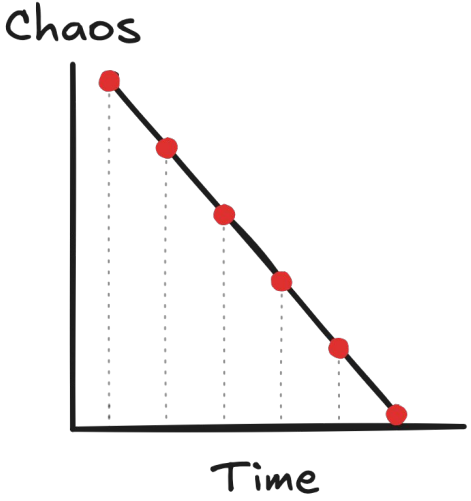
Jochem
van Dooren

🇳🇱 🧀

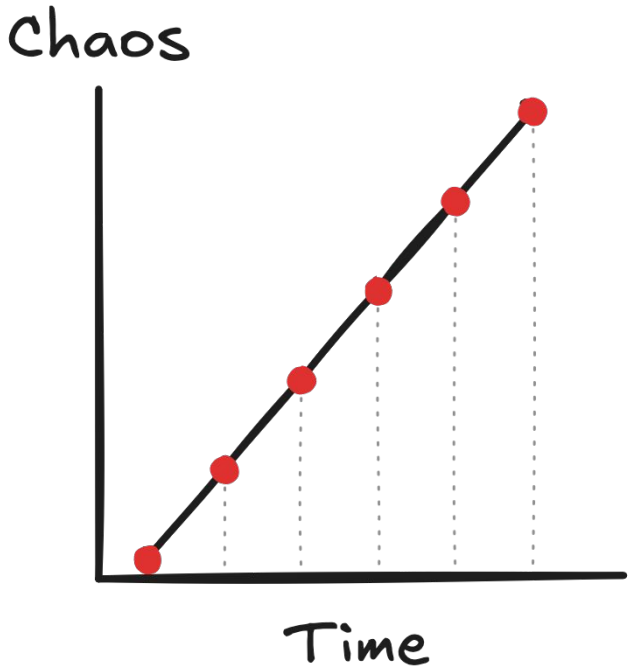
Part 1: Metadata engineering

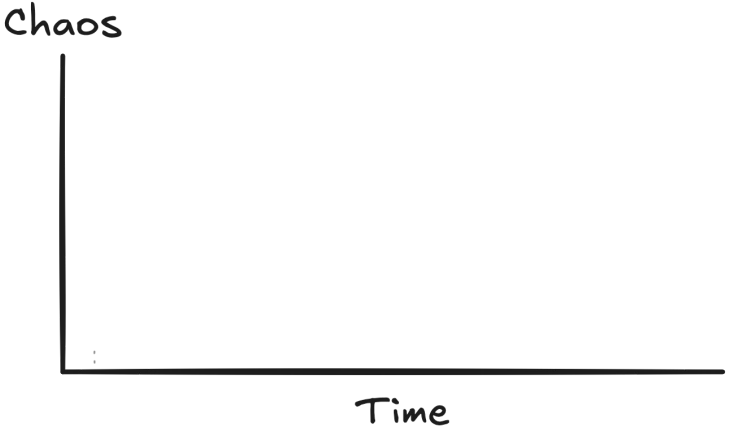


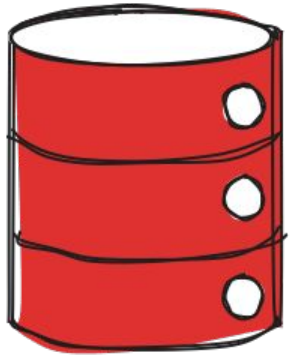
Part 2: dbt-score



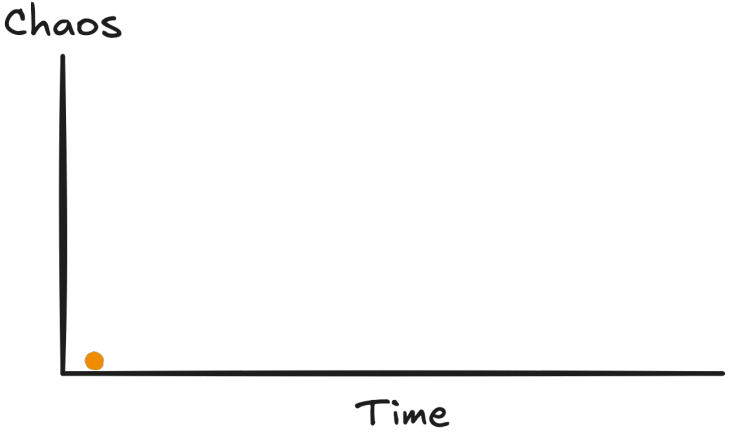
Part 1: Metadata engineering

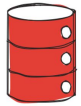



Chaos
Time

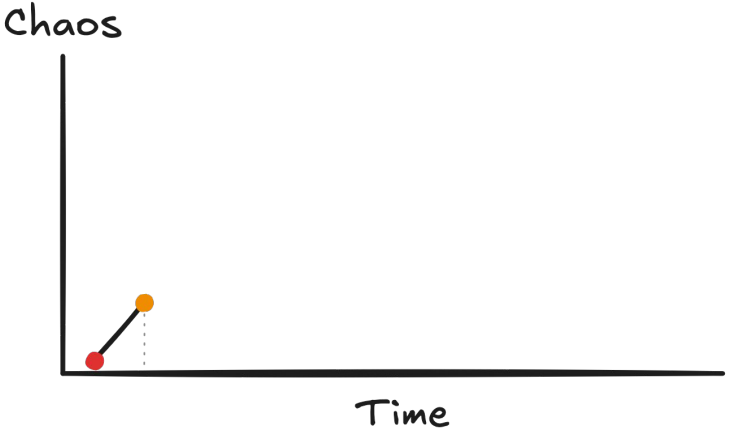


```
create table pizzas (
  id integer,
  name varchar,
  price decimal
)
```

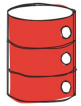




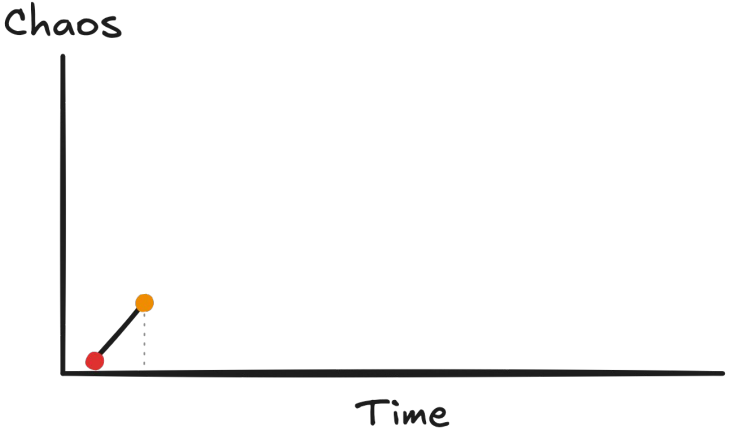
```
create table pizzas (
  id integer,
  name varchar,
  price decimal
)
```



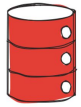

```
select
  id,
  'Pizza ' || name as name,
  10 + 3 * rand() as price
from recipes
where type = 'pizza'
```



```
create table pizzas (
  id integer,
  name varchar,
  price decimal
)
```

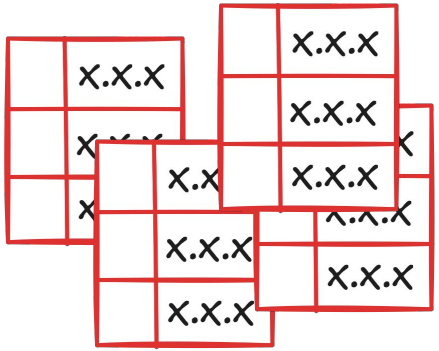


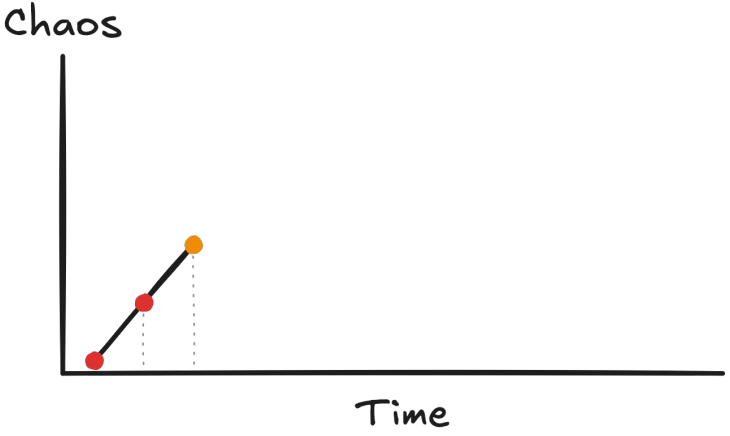

```yaml
models:
- name: pizzas
  columns:
  - name: id
    data_type: int
    constraints:
    - type: primary_key
  - name: name
    data_type: varchar
  - name: price
    data_type: decimal
```



```
create table pizzas (
  id integer,
  name varchar,
  price decimal
)
```

```
models:
- name: pizzas
  columns:
  - name: id
    data_type: int
    constraints:
    - type: primary_key
  - name: name
    data_type: varchar
  - name: price
    data_type: decimal
```



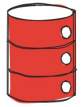



```
models:
- name: pizzas
  columns:
  - name: id
    data_type:
    constraint
```

```
models:
- name: customers
  columns:
```

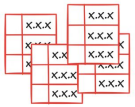
```
models:
- name: deliveries
  columns:
  - name: id
    data_type: int
  - name: customer_id
    data_type: int
    constraints:
    - type: foreign_key
      to: ref('customers')
  - name: delivery_ts
    data_type: timestamp
```

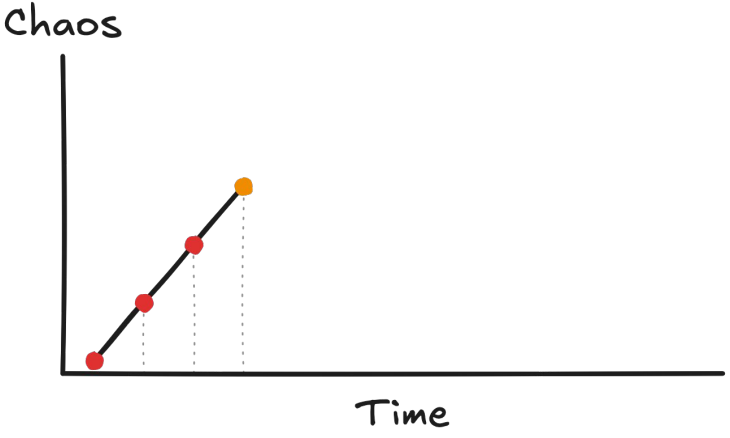
```
models:
- name: weather
  columns:
  - name: id
    data_type: int
  - name: location
    data_type: varchar
  - name: rain_millimeters
    data_type: int
```



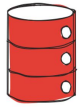
```
create table pizzas (
  id integer,
  name varchar,
  price decimal
)
```

```
models:
- name: pizzas
  columns:
  - name: id
    data_type: int
    constraints:
    - type: primary_key
  - name: name
    data_type: varchar
  - name: price
    data_type: decimal
```



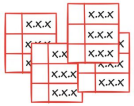


```
models:
- name: pizzas
  columns: [...]
  table_format: iceberg
  catalog: pizza_eng
  dashboards:
  - name: pizza_popularity
    chart: pie
    group_by: pizza_type
  freshness: daily
```

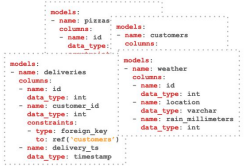


```
create table pizzas (
  id integer,
  name varchar,
  price decimal
)
```

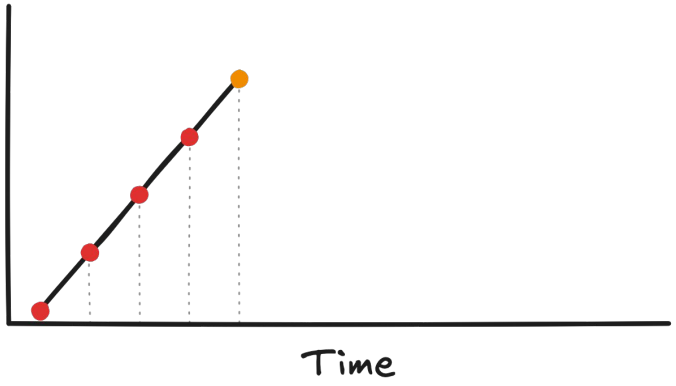
```
models:
- name: pizzas
  columns:
  - name: id
    data_type: int
    constraints:
    - type: primary_key
  - name: name
    data_type: varchar
  - name: price
    data_type: decimal
```



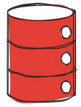
```
models:
- name: pizzas
  columns: [...]
  table_format: iceberg
  catalog: pizza_eng
  dashboards:
  - name: pizza_popularity
    chart: pie
    group_by: pizza_type
  freshness: daily
```



Chaos

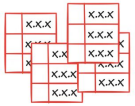


```
models:
- name: items
  columns:
  [...]
  - name: item_kind
    type: varchar
    data_tests:
    - name: allowed_values
      values:
      - 'pizza'
      - 'ice-cream'
      - 'coffee'
```



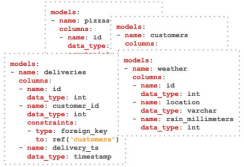
```
create table pizzas (
  id integer,
  name varchar,
  price decimal
)
```

```
models:
- name: pizzas
  columns:
  - name: id
    data_type: int
    constraints:
    - type: primary_key
  - name: name
    data_type: varchar
  - name: price
    data_type: decimal
```

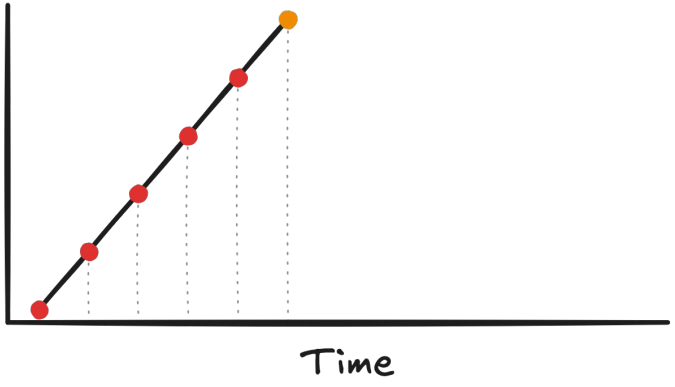


```
models:
- name: pizzas
  columns: [...]
  table_format: iceberg
  catalog: pizza_eng
  dashboards:
  - name: pizza_popularity
    chart: pie
    group_by: pizza_type
  freshness: daily
```

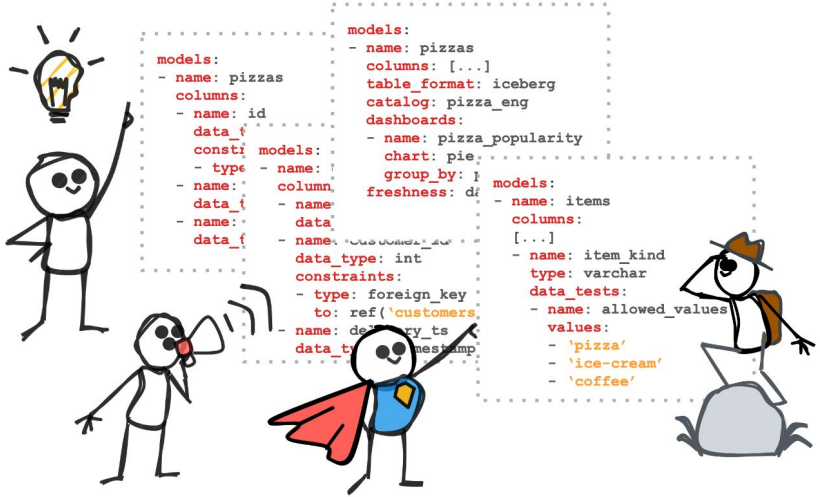
```
models:
- name: items
  columns:
  [...]
  - name: item_kind
    type: varchar
    data_tests:
    - name: allowed_values
      values:
      - 'pizza'
      - 'ice-cream'
      - 'coffee'
```

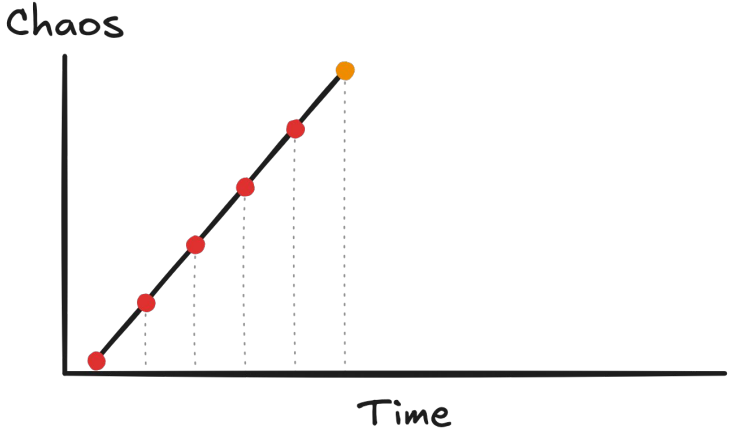


Chaos




models:
- name: pizzas
columns:
- name: id
- name:
- name:
models:
- name: pizzas
columns: [...]
table_format: iceberg
catalog: pizza_eng
dashboards:
- name: pizza_popularity
chart: pie
group_by:
freshness:
models:
- name:
column
- name
data
- name
data_type: int
constraints:
- type: foreign_key
to: ref('customers
- name:
models:
- name: items
columns:
[...]
- name: item_kind
type: varchar
data_tests:
- name: allowed_values
values:
- 'pizza'
- 'ice-cream'
- 'coffee'






```yaml
models:
- name: items
  description: all items sold
  config:
    grants:
      select: [ceo, all]
  meta:
    owner: null
  columns:
  [...]
  - name: id
    type: int
    description:
  - name: item_kind
    type: varchar
    data_tests:
    - name: allowed_values
      values:
      - 'pizza'
```

capitalization

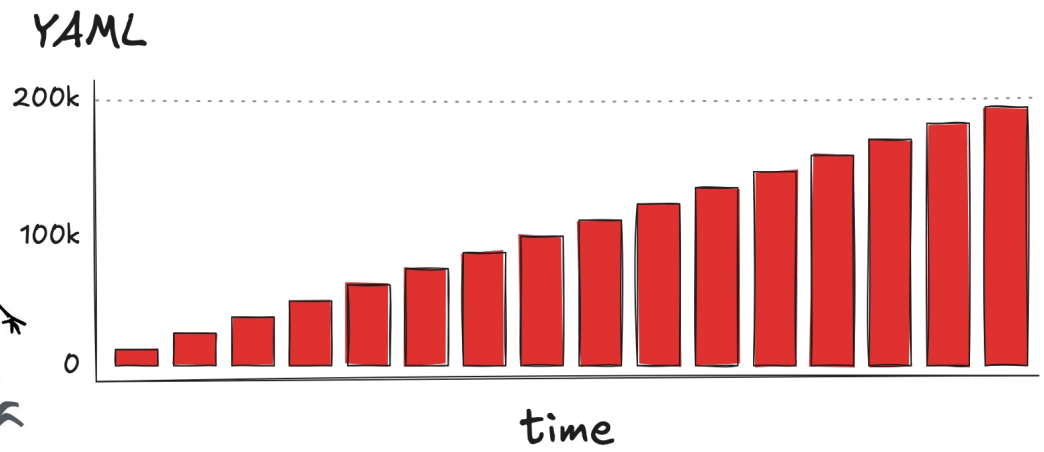
data leak

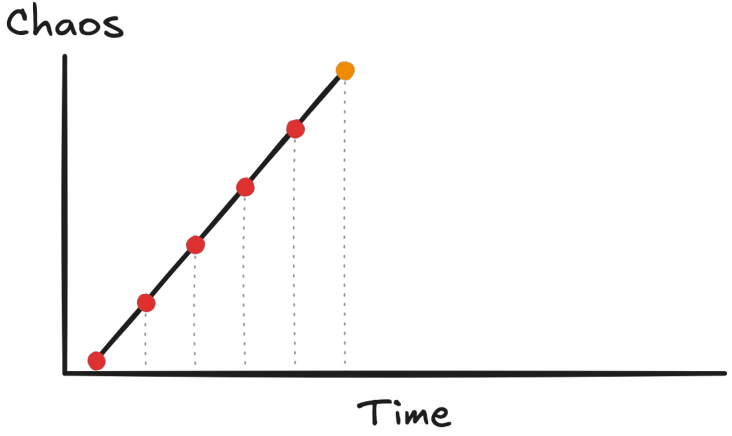
invalid owner

no dashboard

PK constraint?

suspicious test





no dashboard

ambiguous type

consistency

PK constraint?

pineapple pizza

data leak

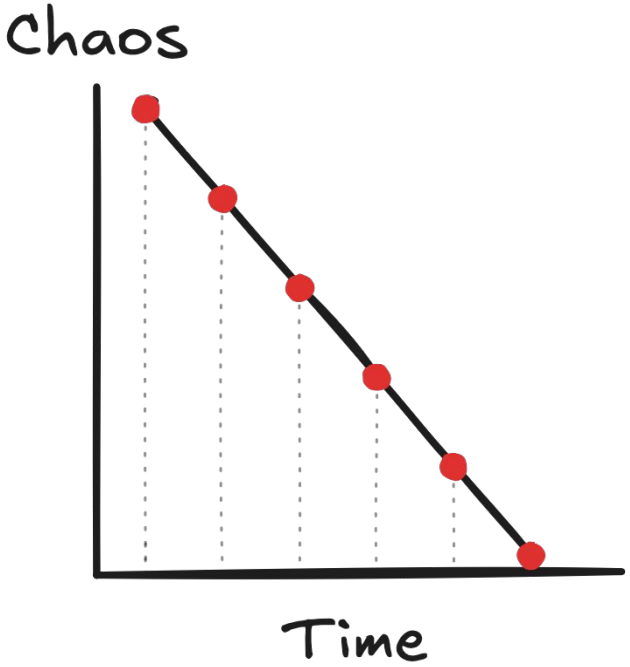
mismatch

incorrect role

invalid owner

suspicious test

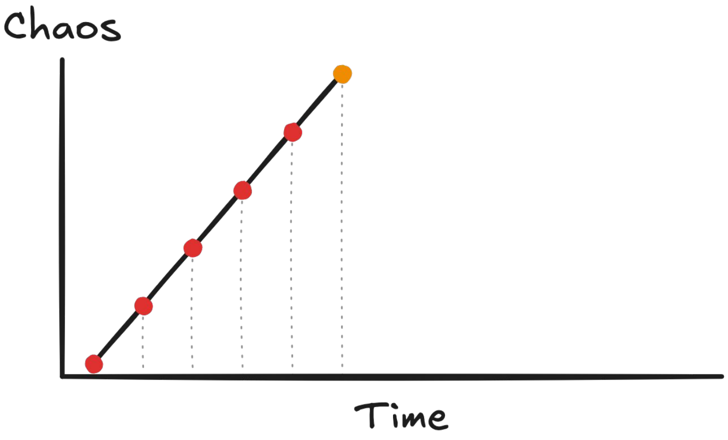
Part 2:
dbt-score



> pip install dbt-score

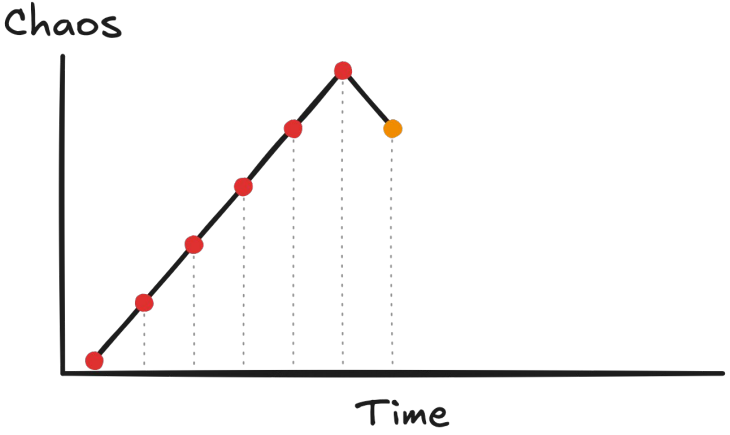
> dbt-score --help

```
Usage: dbt-score [OPTIONS] COMMAND [ARGS]...

       __ __     __
  ____/ // /_   / /_        _____ _____ ____   _____ ___
 / __  // __ \ / __/______ / ___// ___// __ \ / ___// _ \
/ /_/ // /_/ // /_ /_____/(__  )/ /__ / /_/ // /   /  __/
\__,_//_.___/ \__/       /____/ \___/ \____//_/    \___/

Options:
 --version   Show the version and exit.
 -h, --help  Show this message and exit.

Commands:
 lint  Lint dbt metadata.
 list  Display rules list.
```



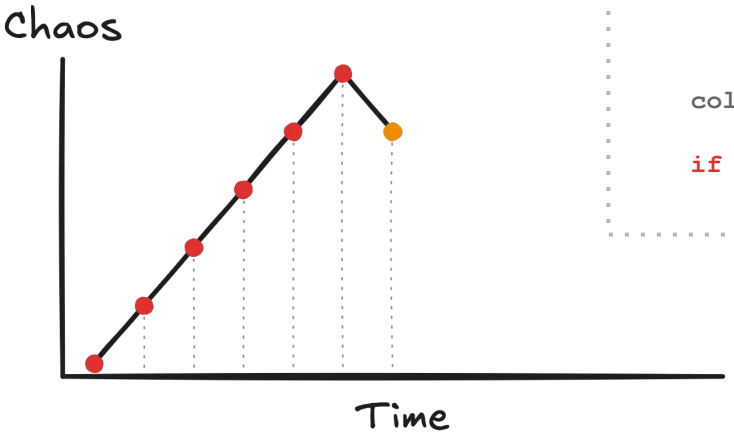
```
> dbt-score lint -s pizzerias
```

🥈 M: pizzerias (score: 8.0)
WARN (medium) columns_have_description: Columns lack a description: id.
WARN (medium) has_owner: Model lacks an owner.

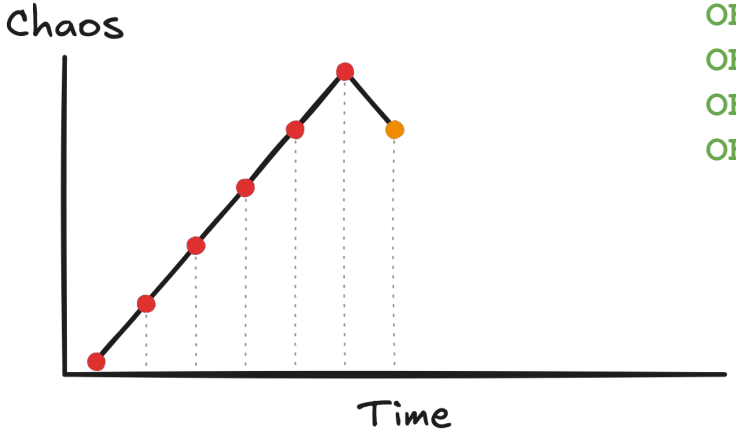



```yaml
models:
- name: pizzerias
  description: all the good pizzerias ⚠️
  config:
    materialized: incremental
    grants:
      select: [pizza_ceo, all] ⚠️
  meta:
    owner: null ⚠️
  columns:
  - name: id ⚠️
    data_type: int
    data_tests:
    - not_null
  - name: location
    description: Geographical location
    data_type: varchar
  - name: rating
    description: Average rating
    data_type: decimal ⚠️
```

```python
@rule
def is_not_capitalized(model: Model) -> RuleViolation | None:
    """Descriptions must be capitalized."""
    if model.description and not model.description.istitle():
        return RuleViolation("Description is not capitalized")
```




```python
@rule(severity=Severity.LOW)
def columns_have_tests(model: Model) -> RuleViolation | None:
    """Model has columns with tests."""
    missing = []
    for column in model.columns:
        if not getattr(column, "data_tests", None):
            missing += [column.name]

    column_string = ', '.join(missing)}"

    if missing:
        return RuleViolation(f"Columns are missing tests: {column_string}")
```



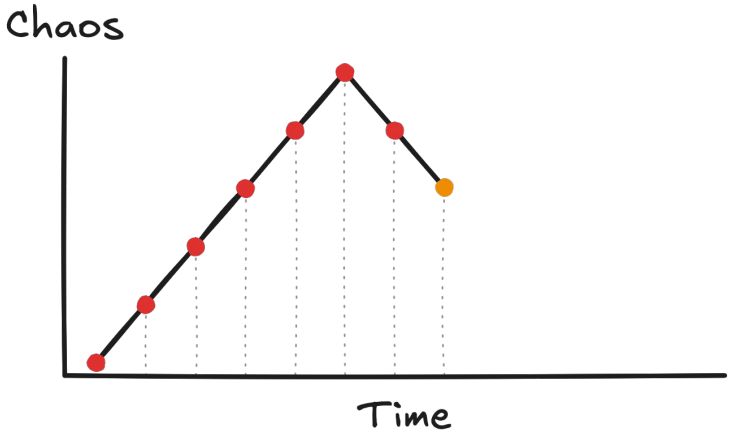

```
> dbt-score lint -s pizzerias --show-all

🥉M: pizzerias (score: 7.0)
WARN (medium) columns_have_tests: Model has
columns without tests: location, rating.
WARN (medium) is_not_capitalized: Description is
not capitalized.
WARN (medium) columns_have_description: Columns
lack a description: id.
WARN (medium) has_owner: Model lacks an owner.
OK has_description
OK has_uniqueness_test
OK single_column_uniqueness_at_column_level
OK single_pk_defined_at_column_level
OK sql_has_reasonable_number_of_lines
```

```python
@rule(severity=Severity.LOW)
def has_dashboar
    """Model has
    if
```

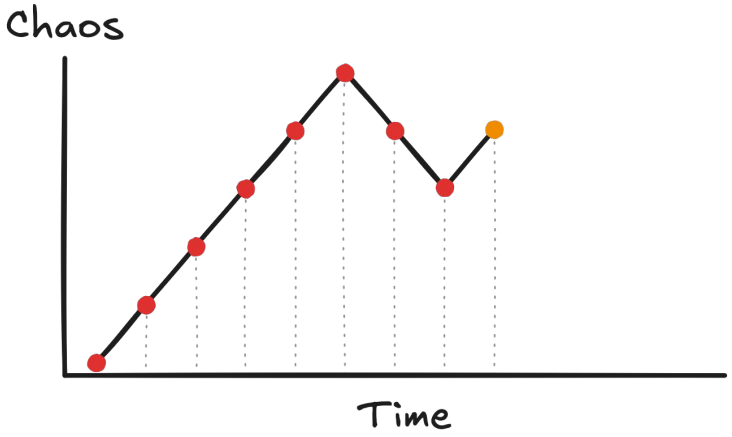
```python
@rule
def no
```

```python
@rule
def max_number_of_columns(model: Model) -> RuleViolation | None:
    """Model does not have more than 20 columns."""
    if len(model.columns) > 20:
```

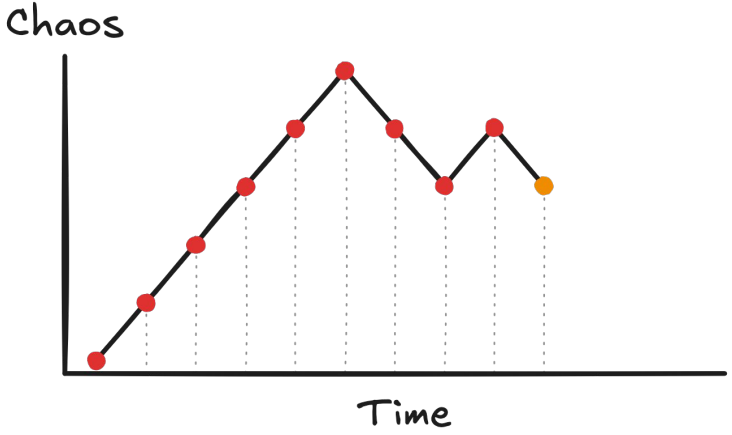
```python
@rule
def max_sql_size(model: Model) -> RuleViolation | None:
    """Model SQL size does not exceed 1000 characters."""
    if len(model.raw_code) > 1000:
        return RuleViolation("Model SQL size exceeds 1000 characters.")
```



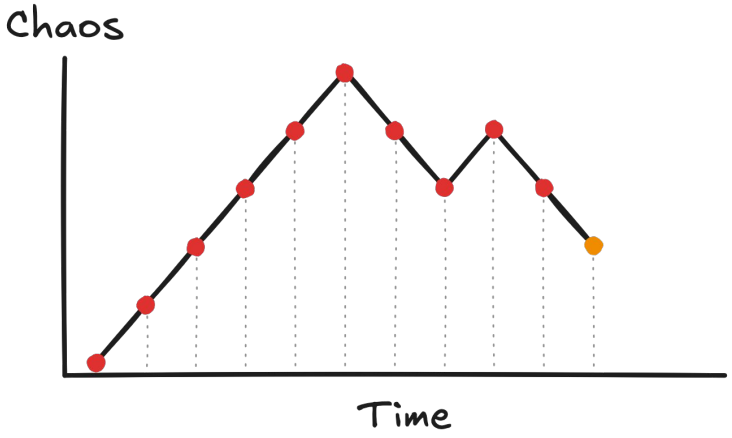

```
> dbt-score lint

Project score: 10.0 🥇
```




```yaml
models:
- name: ice_cream
  description: ice cream shops
  config:
    materialized: incremental
  meta:
    owner: data-noob
  columns:
  - name: id
    data_type: int
  - name: location
    data_type: varchar
  - name: city
    data_type: varchar
  - name: yearly_sales
    data_type: varchar
```




```yaml
name: dbt-score CI

on:
  pull_request:
    branches: [master]
  push:
    branches: [master]

jobs:
  dbt-score:
    runs-on: ubuntu-latest
    steps:
    - name: Checkout code
      uses: actions/checkout@v4
    - name: Set up PDM
      uses: pdm-project/setup-pdm@v4
      with:
        python-version: 3.13
    - name: Install dependencies
      run: pdm sync -d
    - name: Run dbt-score
      run: dbt-score lint
```



✅ Fix dbt-score warnings

17 today ⏱ 1m 19s

ci #501: Pull Request #114 by data-god

❌ Improve pizzas performance

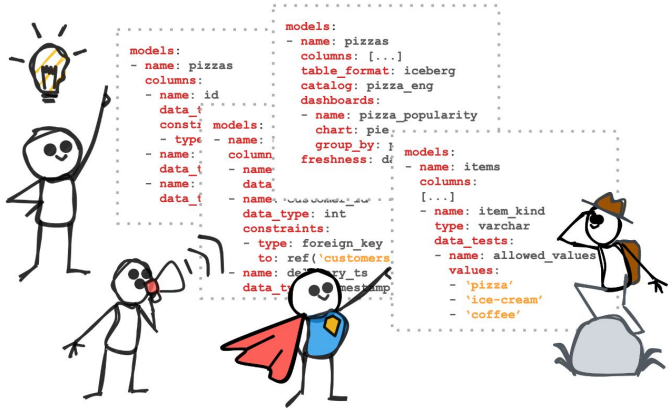
17 yesterday ⏱ 1m 30s

ci #500: Pull Request #113 by data-engineer

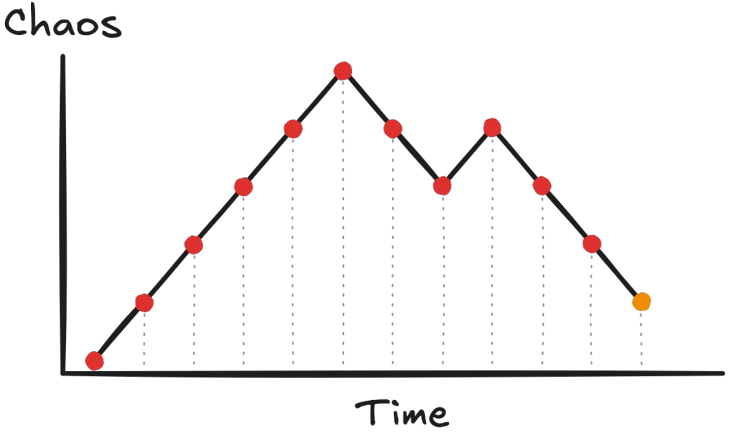
❌ Calculate ice cream aggregates

17 yesterday ⏱ 1m 22s

ci #499: Pull Request #112 by data-engineer




```python
@rule(severity=Severity.LOW)
def has_dashboa
    """Model has
    if
@rule
def no_
@rule
def max_number_of_columns(model: Model) -> RuleViolation | None:
    """Model does not have more than 20 columns."""
    if len(model.columns) > 20:
@rule
def max_sql_size(model: Model) -> RuleViolation | None:
    """Model SQL size does not exceed 1000 characters."""
    if len(model.raw_code) > 1000:
        return RuleViolation("Model SQL size exceeds 1000 characters.")
```



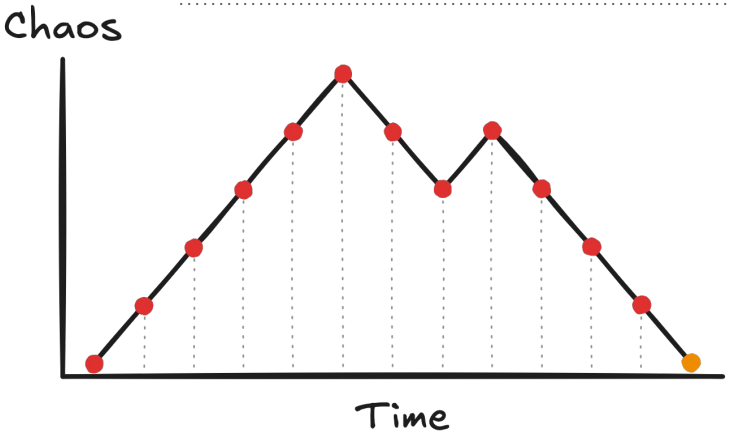

```yaml
name: dbt-score CI

on:
  pull_request:
    branches: [master]
  push:
    branches: [master]

jobs:
  dbt-score:
    runs-on: ubuntu-latest
    steps:
    - name: Checkout code
      uses: actions/checkout@v4
    - name: Set up PDM
      uses: pdm-project/setup-pdm@v4
      with:
        python-version: 3.13
    - name: Install dependencies
      run: pdm sync -d
    - name: Run dbt-score
      run: dbt-score lint
```

✅ Fix dbt-score warnings
today 1m 19s
ci #501: Pull Request #114 by data-god

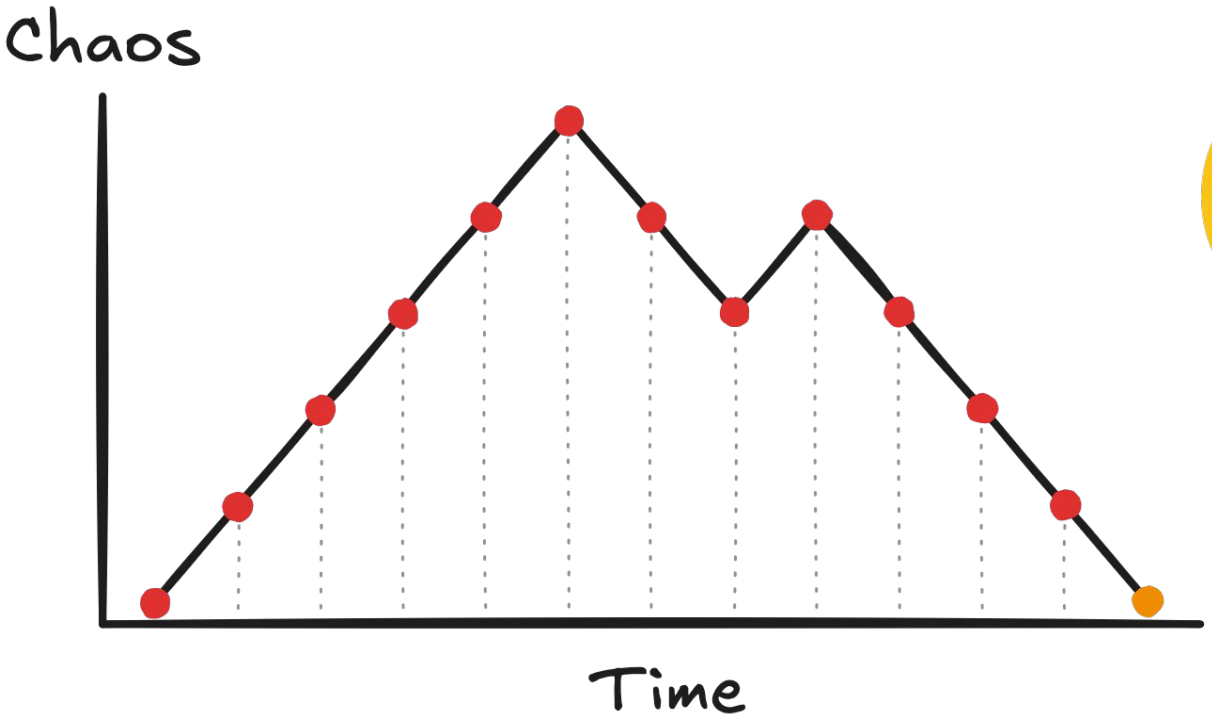
❌ Improve pizzas performance
yesterday 1m 30s
ci #500: Pull Request #113 by data-engineer

❌ Calculate ice cream aggregates
yesterday 1m 22s
ci #499: Pull Request #112 by data-engineer

```toml
[tool.dbt-score]
rule_namespaces = ["company_rules", "project_rules"]
disabled_rules = ["columns_have_description"]
fail_project_under = 7.5
fail_any_item_under = 6.0

[tool.dbt-score.badges]
first.threshold = 10.0
first.icon = "🥇"
second.threshold = 8.0
second.icon = "🥈"
third.threshold = 6.0
third.icon = "🥉"
wip.icon = "🏗️"
```




```toml
[tool.dbt-score]
rule_namespaces = ["project_rules"]
fail_project_under = 8.0
fail_any_item_under = 8.0

[tool.dbt-score.badges]
first.threshold = 10.0
first.icon = "😍"
second.threshold = 9.0
second.icon = "🤩"
third.threshold = 8.0
third.icon = "🙂"
wip.icon = "🤔"
```


Chaos
Time


1

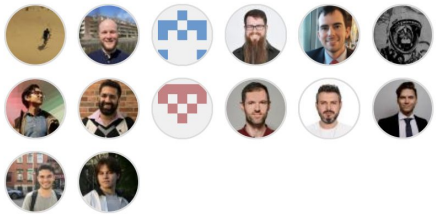
Links

https://dbt-score.picnic.tech

https://github.com/PicnicSupermarket/dbt-score

Credits

dbt-score authors



+ 5 contributors

Picnic