# Data type conversions

CLEANING DATA IN POSTGRESQL DATABASES


**Darryl Reeves, Ph.D.**
Industry Assistant Professor, New York University

datacamp

# Type conversion (an example)

| camis | name | score | inspection_type | ... |
|---|---|---|---|---|
| ... | ... | ... | ... | ... |
| 41659848 | LA BRISA DEL CIBAO | 20 | Cycle Inspection / Initial Inspection | ... |
| 40961447 | MESON SEVILLA RESTAURANT | 50 | Cycle Inspection / Initial Inspection | ... |
| 50063071 | WA BAR | 15 | Cycle Inspection / Initial Inspection | ... |
| 50095871 | ALPHONSO'S PIZZERIA & TRATTORIA | 10 | Cycle Inspection / Initial Inspection | ... |
| 41104041 | THE SPARROW TAVERN | 13 | Cycle Inspection / Initial Inspection | ... |
| ... | ... | ... | ... | ... |

# Determining column types

```
SELECT
    column_name,
    data_type
FROM
    information_schema.columns
WHERE
    table_name = 'restaurant_inspection';
```

```
column_name      | data_type
-----------------+-----------
camis            | bigint
name             | text
boro             | text
building         | text
street           | text
zip_code         | smallint
...              | ...
```

# Determining column types

```
SELECT
    column_name,
    data_type
FROM
    information_schema.columns
WHERE
    table_name = 'restaurant_inspection' AND
    column_name = 'camis';
```

```
 column_name | data_type
-------------+-----------
 camis       | bigint
```

# Conversion with CASE

- Type conversion with a `CASE` clause
- Grades are given as `A`, `B`, and `C`
- Conversion: `A` = 3, `B` = 2, `C` = 1

```
SELECT
  boro,
  AVG(grade_points)
FROM (
  SELECT
    *,
    CASE
      WHEN grade = 'A' then 3
      WHEN grade = 'B' then 2
      WHEN grade = 'C' then 1
    END AS grade_points
  FROM
    restaurant_inspection
  ) sub
GROUP BY boro;
```

# Conversion with CASE

```
SELECT
  boro,
  AVG(grade_points)
FROM (
  SELECT
    *,
    CASE
      WHEN grade = 'A' then 3
      WHEN grade = 'B' then 2
      WHEN grade = 'C' then 1
    END AS grade_points
  FROM
    restaurant_inspection
  ) sub
GROUP BY boro;
```

```
boro          |        avg
--------------+-------------------
Brooklyn      | 2.7641196013289037
Bronx         | 2.7685589519650655
Manhattan     | 2.7678381256656017
Queens        | 2.7803571428571429
Staten Island | 2.8068181818181818
```

# Conversion with CAST

```
camis    | diff
---------+------
...      |   ...
50080214 |   87
50059239 |   74
50086316 |   74
41637438 |   71
41667902 |   64
50067622 |   61
50017111 |   60
50017056 |   60
50045240 |   59
50002403 |   59
...      |   ...
```

```sql
SELECT
  camis,
  MAX(score) - MIN(score) AS diff
FROM
  restaurant_inspection
WHERE
  score IS NOT NULL
GROUP BY
  camis
ORDER BY
  diff DESC;
```

# Conversion with CAST()

`CAST( value AS type)`

```
SELECT
  camis,
  MAX(CAST(score AS int)) - MIN(CAST(score AS int)) AS diff
FROM
  restaurant_inspection
WHERE
  score IS NOT NULL
GROUP BY
  camis
ORDER BY
  diff DESC
```

# Conversion with double colon (::)

`value::type`

```
SELECT
  camis,
  MAX(score::int) - MIN(score::int) AS diff
FROM
  restaurant_inspection
WHERE
  score IS NOT NULL
GROUP BY
  camis
ORDER BY
  diff DESC
```

# Let's practice!

CLEANING DATA IN POSTGRESQL DATABASES

# Date parsing and formatting

CLEANING DATA IN POSTGRESQL DATABASES


**Darryl Reeves, Ph.D.**
Industry Assistant Professor, New York University

datacamp

# Parsing dates with the DATE() function

| camis | name | inspection_date | grade_date | ... |
|---|---|---|---|---|
| ... | ... | ... | ... | ... |
| 50034992 | EMPANADAS MONUMENTAL | 06/21/2019 | 06/21/2019 | ... |
| 50095871 | ALPHONSO'S PIZZERIA | 01/16/2020 | 01/16/2020 | ... |
| 41104041 | THE SPARROW TAVERN | 09/17/2019 | 09/17/2019 | ... |
| 50016937 | BURGER KING | 09/14/2018 | 09/14/2018 | ... |
| 50033304 | ASTORIA PIZZA | 12/18/2019 | 12/18/2019 | ... |
| ... | ... | ... | ... | ... |

- `DATE` functionality unavailable for `TEXT` column
  - Checking date ranges
  - Extracting date components
  - Calculating interval between dates
- `DATE(string_date)`
  - Converts `string_date` to `DATE` values
  - `DATE('2019-12-01')` → `DATE` value

# Parsing dates with the DATE() function

```
SELECT
  camis,
  name,
  DATE(inspection_date) AS inspection_date,
  DATE(grade_date) AS grade_date
FROM
  restaurant_inspection;
```

```
 camis    |              name                | inspection_date | grade_date | ...
----------+----------------------------------+-----------------+------------+-----
...       | ...                              | ...             | ...        | ...
50034992  | EMPANADAS MONUMENTAL             | 2019-06-21      | 2019-06-21 | ...
50095871  | ALPHONSO'S PIZZERIA & TRATTORIA  | 2020-01-16      | 2020-01-16 | ...
41104041  | THE SPARROW TAVERN               | 2019-09-17      | 2019-09-17 | ...
50016937  | BURGER KING                      | 2018-09-14      | 2018-09-14 | ...
50033304  | ASTORIA PIZZA                    | 2019-12-18      | 2019-12-18 | ...
...       | ...                              | ...             | ...        | ...
```

# Parsing dates with the TO_DATE() function

- `TO_DATE(date_string, format_string)` → `DATE` value
- `DATE('Wednesday, June 10th, 2014')` → ERROR
- `TO_DATE('Wednesday, June 10th, 2014', 'Day, Month DDth, YYYY')` → `DATE` value

# The NULLIF() expression

```
camis   |              name                | inspection_date | grade_date | ...
--------+----------------------------------+-----------------+------------+-----
...     | ...                              | ...             | ...        | ...
41659848 | LA BRISA DEL CIBAO              | 2018-01-30      | -          | ...
40961447 | MESON SEVILLA RESTAURANT        | 2019-03-19      | -          | ...
50063071 | WA BAR                          | 2018-05-23      | -          | ...
50034992 | EMPANADAS MONUMENTAL            | 2019-06-21      | 2019-06-21 | ...
50095871 | ALPHONSO'S PIZZERIA & TRATTORIA | 2020-01-16      | 2020-01-16 | ...
...     | ...                              | ...             | ...        | ...
```

`NULLIF(value1, value2)`

```
SELECT
  NULLIF(grade_date, '-')
FROM
  restaurant_inspection;
```

# Displaying dates with the TO_CHAR() function

- Default date format:
  - YYYY-MM-DD (ex. 2012-04-03)
- `TO_CHAR('2012-04-03', YYYY-DD-MM)`
- `TO_CHAR(date_value, format_string)` → string value

```
SELECT
  camis,
  name,
  TO_CHAR(
        inspection_date::date,
        'MM/DD/YY'
  ) AS inspection_date
FROM
  restaurant_inspection;
```

| camis | name | inspection_date | ... |
|---|---|---|---|
| ... | ... | ... | ... |
| 41659848 | LA BRISA DEL CIBAO | 01/30/2018 | ... |
| 40961447 | MESON SEVILLA RESTAURANT | 03/19/2019 | ... |
| 50063071 | WA BAR | 05/23/2018 | ... |
| 50034992 | EMPANADAS MONUMENTAL | 06/21/2019 | ... |
| 50095871 | ALPHONSO'S PIZZERIA & TRATTORIA | 01/16/2020 | ... |
| ... | ... | ... | ... |

| camis | name | inspection_date | ... |
|---|---|---|---|
| ... | ... | ... | ... |
| 41659848 | LA BRISA DEL CIBAO | 01/30/20 | ... |
| 40961447 | MESON SEVILLA RESTAURANT | 03/19/20 | ... |
| 50063071 | WA BAR | 05/23/20 | ... |
| 50034992 | EMPANADAS MONUMENTAL | 06/21/20 | ... |
| 50095871 | ALPHONSO'S PIZZERIA & TRATTORIA | 01/16/20 | ... |
| ... | ... | ... | ... |

# Date format patterns

- `TO_DATE(date_string, format)`
- `TO_CHAR(date_value, format)`

# Date format patterns with TO_DATE()

`YYYY`

```
TO_DATE('2012', 'YYYY')
```

```
DATE
```

`MM`

```
TO_DATE('09/2012', 'MM/YYYY')
```

```
DATE
```

`DD`

```
TO_DATE('09/03/2012', 'MM/DD/YYYY')
```

```
DATE
```

`Day`

```
TO_DATE('Sunday, the 10th', 'Day, the DDth')
```

```
DATE
```

[1] https://www.postgresql.org/docs/12/functions-formatting.html

# Date format patterns with TO_CHAR()

`YYYY`

```
TO_CHAR('2012-09-03'::date, 'YYYY')
```

```
2012
```

`MM`

```
TO_CHAR('09/03/2012'::date, 'MM/YYYY')
```

```
09/2012
```

`DD`

```
TO_CHAR('09/03/2012'::date, 'MM/DD/YYYY')
```

```
09/03/2012
```

`Day`

```
TO_CHAR('09/03/2012'::date, 'Day, the DDth')
```

```
Monday, the 03rd
```

[1] https://www.postgresql.org/docs/12/functions-formatting.html

# Let's practice!

CLEANING DATA IN POSTGRESQL DATABASES

# Timestamp parsing and formatting

## CLEANING DATA IN POSTGRESQL DATABASES


**Darryl Reeves, Ph.D.**
Industry Assistant Professor, New York University

datacamp

# PostgreSQL timestamps

| camis | name | inspection_datetime | inspection_type | ... |
|---|---|---|---|---|
| ... | ... | ... | ... | ... |
| 50000458 | BEVERLEY PIZZA & CAFE | 2019-07-08 14:26 | Cycle Inspection / Initial Inspection | ... |
| 50002521 | JADE PALACE | 2018-05-14 12:35 | Cycle Inspection / Initial Inspection | ... |
| 40389732 | GIANDO | 2017-07-10 13:39 | Cycle Inspection / Re-inspection | ... |
| 50044246 | FLEET BAKERY | 2019-10-29 15:40 | Cycle Inspection / Re-inspection | ... |
| 50038120 | SHUN WON FLUSHING | 2018-07-17 16:20 | Cycle Inspection / Re-inspection | ... |
| ... | ... | ... | ... | ... |

`inspection_datetime` : `TIMESTAMP` column

# Parsing timestamps with TO_TIMESTAMP()

- Convert strings to `TIMESTAMP`
- `TO_TIMESTAMP(ts_string, format_string)` → `TIMESTAMP`

```
SELECT
  camis,
  name,
  TO_TIMESTAMP(inspection_datetime, 'YYYY-MM-DD HH24:MI'),
  inspection_type
FROM
  restaurant_inspection;
```

# Timestamp string format patterns

- `TO_TIMESTAMP(ts_string, format)`
- `TO_CHAR(ts_value, format)`
- `TO_DATE()` patterns (`YYYY`, `MM`, `Day`, ...) usable

# Timestamp string format patterns

- `TO_TIMESTAMP(ts_string, format)`

| Pattern | TO_TIMESTAMP() Example |
|---|---|
| `HH24` | `TO_TIMESTAMP('23', 'HH24')` → `TIMESTAMP` |
| `HH12` | `TO_TIMESTAMP('01', 'HH12')` → `TIMESTAMP` |
| `MI` | `TO_TIMESTAMP('18:13', 'HH24:MI')` → `TIMESTAMP` |
| `SS` | `TO_TIMESTAMP('33:20', 'MI:SS')` → `TIMESTAMP` |
| `PM` or `AM` | `TO_TIMESTAMP('5:35AM', 'HH12:MIPM')` → `TIMESTAMP` |

[1] https://www.postgresql.org/docs/12/functions-formatting.html

# The EXTRACT() function

```
EXTRACT(time_unit FROM time_value)
```

- `time_value` - `DATE` or `TIMESTAMP`

# The EXTRACT() function

```
SELECT
  camis,
  name,
  inspection_datetime,
  EXTRACT('year' FROM inspection_datetime) AS year,
  inspection_type
FROM
  restaurant_inspection;
```

```
 camis    |         name          |    inspection_datetime     | year |            inspection_type            | ...
----------+-----------------------+----------------------------+------+---------------------------------------+-----
 ...      | ...                   | ...                        | ...  | ...                                   | ...
 50000458 | BEVERLEY PIZZA & CAFE | 2019-07-08 06:37:46.658905 | 2019 | Cycle Inspection / Initial Inspection | ...
 50002521 | JADE PALACE           | 2018-05-14 03:47:24.474573 | 2018 | Cycle Inspection / Initial Inspection | ...
 40389732 | GIANDO                | 2017-07-10 03:59:12.864428 | 2017 | Cycle Inspection / Re-inspection      | ...
 50044246 | FLEET BAKERY          | 2019-10-29 02:06:33.614964 | 2019 | Cycle Inspection / Re-inspection      | ...
 50038120 | SHUN WON FLUSHING     | 2018-07-17 01:15:04.15666  | 2018 | Cycle Inspection / Re-inspection      | ...
 ...      | ...                   | ...                        | ...  | ...                                   | ...
```

# Time unit options for EXTRACT()

| Time Unit | EXTRACT() Example |
|---|---|
| `year` | `EXTRACT('year' FROM '2020-07-20 16:42:21'::timestamp)` → 2020 |
| `month` | `EXTRACT('month' FROM '2020-07-20 16:42:21'::timestamp)` → 7 |
| `day` | `EXTRACT('day' FROM '2020-07-20 16:42:21'::timestamp)` → 20 |
| `hour` | `EXTRACT('hour' FROM '2020-07-20 16:42:21'::timestamp)` → 16 |
| `minute` | `EXTRACT('minute' FROM '2020-07-20 16:42:21'::timestamp)` → 42 |
| `second` | `EXTRACT('second' FROM '2020-07-20 16:42:21'::timestamp)` → 21 |

[1] https://www.postgresql.org/docs/current/functions-datetime.html

# Let's practice!

CLEANING DATA IN POSTGRESQL DATABASES