Introduction to data cleaning

CLEANING DATA IN POSTGRESQL DATABASES

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

datacamp

Why is cleaning data important?

- Data is messy
- Before it can be analyzed, often needs cleaning
- Helpful to utilize column type constraints
- Course focuses on when defensive approaches not available/applicable

Cleaning string data

- String data: abundant, flexible, and often messy

```
name                            | grade |            inspection_type                | census_tract |
--------------------------------+-------+-------------------------------------------+--------------+--
...                             | ...   | ...                                       | ...          |
EMPANADAS MONUMENTAL            | B     | Cycle Inspection  /  Re-inspection        | 26900        |
ALPHONSO'S PIZZERIA & TRATTORIA | A     | Cycle Inspection  /  Initial Inspection   | 202          |
THE SPARROW TAVERN              | A     | Cycle Inspection  /  Initial Inspection   | 12500        |
BURGER KING                     | A     | Cycle Inspection  /  Re-inspection        | 86400        |
ASTORIA PIZZA                   | B     | Cycle Inspection  /  Re-inspection        | 6300         |
...                             | ...   | ...                                       | ...          |
```

Cleaning string data

name	grade	inspection_type	census_tract
...	...	...	...
EMPANADAS MONUMENTAL	B	Cycle Inspection / Re-inspection	26900
ALPHONSO'S PIZZERIA & TRATTORIA	A	Cycle Inspection / Initial Inspection	202
THE SPARROW TAVERN	A	Cycle Inspection / Initial Inspection	12500
BURGER KING	A	Cycle Inspection / Re-inspection	86400
ASTORIA PIZZA	B	Cycle Inspection / Re-inspection	6300
...	...	...	...

1. Restrict capitalization in `name`
2. Remove extra divider space in `inspection_type`
3. Make `census_tract` values have a uniform length

Cleaning string data

name	grade	inspection_type	census_tract	...
...	...	...	...	...
EMPANADAS MONUMENTAL	B	Cycle Inspection / Re-inspection	26900	...
ALPHONSO'S PIZZERIA & TRATTORIA	A	Cycle Inspection / Initial Inspection	202	...
THE SPARROW TAVERN	A	Cycle Inspection / Initial Inspection	12500	...
BURGER KING	A	Cycle Inspection / Re-inspection	86400	...
ASTORIA PIZZA	B	Cycle Inspection / Re-inspection	6300	...
...	...	...	...	...

name	grade	inspection_type	census_tract	...
...	...	...	...	...
Empanadas Monumental	B	Cycle Inspection / Re-inspection	026900	...
Alphonso'S Pizzeria & Trattoria	A	Cycle Inspection / Initial Inspection	000202	...
The Sparrow Tavern	A	Cycle Inspection / Initial Inspection	012500	...
Burger King	A	Cycle Inspection / Re-inspection	086400	...
Astoria Pizza	B	Cycle Inspection / Re-inspection	006300	...
...	...	...	...	...

Using the INITCAP() function

`INITCAP(input_string)` - fixing capitalization

```
SELECT INITCAP('HELLO FRIEND!');
```

```
Hello Friend!
```

Using the REPLACE() function

`REPLACE(input_string, to_replace, replacement)` - replacing one text value with another

```
SELECT REPLACE('180 Main Street', 'Street', 'St');
```

```
180 Main St
```

Using the LPAD() function

`LPAD(input_string, length [, fill_value])` - prepending text values to a string

```
SELECT LPAD('123', 7, 'X');
```

```
XXXX123
```

Building the string cleaning query

```
SELECT
  INITCAP(name) as name,
  grade,
  REPLACE(inspection_type, '  /  ', ' / ') as inspection_type,
  LPAD(census_tract, 6, '0') as census_tract
FROM
  restaurant_inspection;
```

```
name                            | grade |           inspection_type              | census_tract | ...
--------------------------------+-------+----------------------------------------+--------------+-----
...                             | ...   | ...                                    | ...          | ...
Empanadas Monumental            | B     | Cycle Inspection / Re-inspection       | 026900       | ...
Alphonso'S Pizzeria & Trattoria | A     | Cycle Inspection / Initial Inspection  | 000202       | ...
The Sparrow Tavern              | A     | Cycle Inspection / Initial Inspection  | 012500       | ...
Burger King                     | A     | Cycle Inspection / Re-inspection       | 086400       | ...
Astoria Pizza                   | B     | Cycle Inspection / Re-inspection       | 006300       | ...
...                             | ...   | ...                                    | ...          | ...
```

Let's practice!

CLEANING DATA IN POSTGRESQL DATABASES

Pattern matching

CLEANING DATA IN POSTGRESQL DATABASES

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

datacamp

Identifying patterns: an example

camis	name	inspection_date	score	nta	...
...	...	...	...	...	...
41659848	LA BRISA DEL CIBAO	01/30/2018	20	QN26	...
40961447	MESON SEVILLA RESTAURANT	03/19/2019	50	MN15	...
50063071	WA BAR	05/23/2018	15	MN17	...
50034992	EMPANADAS MONUMENTAL	06/21/2019	17	MN35	...
50095871	ALPHONSO'S PIZZERIA & TRATTORIA	01/16/2020	10	MN28	...
41104041	THE SPARROW TAVERN	09/17/2019	13	QN72	...
50016937	BURGER KING	09/14/2018	12	QN55	...
50066469	DARBAR'S CHICKEN & RIBS	08/07/2017	11	QN55	...
41195691	F & J PINE RESTAURANT	05/02/2019	26	BX49	...
50015706	EL RINCONCITO DE LOS SABORES	12/18/2019	20	QN35	...
...	...	...	..	...	...

[1] https://www1.nyc.gov/site/planning/data-maps/open-data/dwn-nynta.page

Identifying patterns: an example

Valid Prefixes to NTA Code

- MN - Manhattan
- BK - Brooklyn
- BX - Bronx
- QN - Queens
- SI - Staten Island

Identifying patterns: an example

camis	name	inspection_date	score	nta	...
...	...	...	...	...	...
50058910	HUNGER PANG	06/15/2017	13	BK42	...
40376029	MOMS LUNCHEONETTE	03/14/2020	13	QN544	...
50019128	IKI MODERN JAPANESE CUISINE	07/13/2017	10	QN22	...
50000458	BEVERLEY PIZZA & CAFE	07/08/2019	12	BK41	...
50002521	JADE PALACE	05/14/2018	11	BX13	...
...	...	...	...	...	...

The LIKE operator

```
SELECT * FROM restaurant_inspection WHERE nta LIKE 'QN544';
```

```
SELECT * FROM restaurant_inspection WHERE nta = 'QN544';
```

The LIKE Operator

Pattern Matching Characters

- `%` - matches any sequence of zero or more characters
- `_` (underscore) - matches a single character

```
SELECT
*
FROM
restaurant_inspection
WHERE nta LIKE 'QN%';
```

```
SELECT
*
FROM
restaurant_inspection
WHERE nta LIKE 'QN%'
AND nta NOT LIKE 'QN__';
```

Regular expressions (REs)

- Pattern matching with `LIKE` is limited
- More specific patterns can be useful
- Regular Expressions (REs) enable more expressive pattern matching

The SIMILAR TO operator

- Provides additional pattern matching functionality

```
SELECT
  camis, name, inspection_date, score, nta
FROM
  restaurant_inspection
WHERE nta SIMILAR TO 'QN%' AND nta NOT SIMILAR TO 'QN__';
```

Basics of REs

Metacharacter	Usage	Example RE	Example Match
`\d`	matches a digit (0-9)	`\d\d\d`	'345'
`?`	matches 0 or 1 of previous character	`x\d?`	'x5'
`+`	matches one or more of previous character	`\d+`	'10'
`*`	matches any character 0 or more times	`\d*`	'3081'
`[]`	matches any character inside of the brackets	`[a-z]`	'f'

Using REs with SIMILAR TO

```
SELECT
    camis, name, inspection_date, score, nta
FROM
    restaurant_inspection
WHERE nta SIMILAR TO 'QN%' AND nta NOT SIMILAR TO 'QN__';
```

```
SELECT
    camis, name, inspection_date, score, nta
FROM
    restaurant_inspection
WHERE
    nta NOT SIMILAR TO '[A-Z][A-Z]\d\d';
```

```
camis    |        name        | inspection_date | score | nta
---------+--------------------+-----------------+-------+------
41659848 | LA BRISA DEL CIBAO | 01/30/2018      | 20    | Q26
41104041 | THE SPARROW TAVERN | 09/17/2019      | 13    | QN723
```

Let's practice!

CLEANING DATA IN POSTGRESQL DATABASES

Matching similar strings

CLEANING DATA IN POSTGRESQL DATABASES

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

datacamp

Similar strings (example)

- Delivery address: `121 Fontainebleau Drive`
- `Fountainbleau` , `Fontainbleu` , `Fontainblue`

The Soundex algorithm

- Words represented by sound
- Encodes words using 4 characters
- `Fountainbleau` , `Fontainbleu` , `Fontainblue` → `F535`

SOUNDEX() in PostgreSQL

- Available through `fuzzystrmatch` module

```
CREATE EXTENSION fuzzystrmatch;
```

`SOUNDEX(input_string)` → 4 character code

```
SELECT
  SOUNDEX('Fountainbleau') as sd1,
  SOUNDEX('Fontainebleau') as sd2,
  SOUNDEX('Fontaineblue') as sd3;
```

```
 sd1  | sd2  | sd3
------+------+------
F535 | F535 | F535
```

The DIFFERENCE() function

`DIFFERENCE(string1, string2)` → 0, 1, 2, 3, or 4

```
SELECT SOUNDEX('pair') AS sd_pair, SOUNDEX('pear') AS sd_pear;
```

```
 sd_pair | sd_pear
---------+---------
 P600    | P600
```

```
SELECT DIFFERENCE('pair', 'pear') AS diff;
```

```
 diff
------
    4
```

The DIFFERENCE() function

```
SELECT SOUNDEX('bow') AS sd_bow, SOUNDEX('bough') AS sd_bough;
```

```
 sd_bout | sd_bought
---------+-----------
 B300    | B230
```

```
SELECT DIFFERENCE('bout', 'bought') AS diff
```

```
 diff
------
    2
```

Using DIFFERENCE()

```
name                   |   boro    | building |            street
-----------------------+-----------+----------+-------------------------------
ATOMIC WINGS           | Manhattan | 2090     | FREDERICK DOUGLASS BOULEVARD
BARAKA BUFFET          | Manhattan | 2546     | FREDERICK DOUGLASS BOULEVARD
CHOCOLAT               | Manhattan | 2217     | FREDERICK DOUGLASS BOULEVARD
ESO                    | Manhattan | 2906     | FREDERICK DOUGLASS BOULEVARD
HOP HOUSE HARLEM       | Manhattan | 2224     | FREDERICK DOUGLASS BOULEVARD
HOT POT UNDER DE' TREE | Manhattan | 2839     | FREDERICK DOUGLAS BOULEVARD
LIDO                   | Manhattan | 2168     | FREDERICK DOUGLAS BOULEVARD
MESS HALL              | Manhattan | 2194     | FRDRCK DGLS BLVD
VINATERIA              | Manhattan | 2211     | FREDERICK DOUGLAS BOULEVARD
```

Using DIFFERENCE()

```
name                    |   boro    | building |            street             | sd_street
 ----------------------+-----------+----------+-------------------------------+-----------
 ATOMIC WINGS           | Manhattan | 2090     | FREDERICK DOUGLASS BOULEVARD | F636
 BARAKA BUFFET          | Manhattan | 2546     | FREDERICK DOUGLASS BOULEVARD | F636
 CHOCOLAT               | Manhattan | 2217     | FREDERICK DOUGLASS BOULEVARD | F636
 ESO                    | Manhattan | 2906     | FREDERICK DOUGLASS BOULEVARD | F636
 HOP HOUSE HARLEM       | Manhattan | 2224     | FREDERICK DOUGLASS BOULEVARD | F636
 HOT POT UNDER DE' TREE | Manhattan | 2839     | FREDERICK DOUGLAS BOULEVARD  | F636
 LIDO                   | Manhattan | 2168     | FREDERICK DOUGLAS BOULEVARD  | F636
 MESS HALL              | Manhattan | 2194     | FRDRCK DGLS BLVD             | F636
 VINATERIA              | Manhattan | 2211     | FREDERICK DOUGLAS BOULEVARD  | F636
```

```
SELECT
  name, boro, building, street
FROM
  restaurant_inspections
WHERE
  DIFFERENCE(street, 'Frederick Douglass Boulevard') = 4;
```

Updating the recordings

```
UPDATE
    table_name
SET
    column_name = value
WHERE
    condition
```

```
UPDATE
    restaurant_inspection
SET
    street = 'Frederick Douglass Boulevard'
WHERE
    DIFFERENCE(street, 'Frederick Douglass Boulevard') = 4;
```

```
UPDATE 10
```

Let's practice!

CLEANING DATA IN POSTGRESQL DATABASES

Handling missing data

CLEANING DATA IN POSTGRESQL DATABASES

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

datacamp

Missing data (an example)

```
... |       name       | score |                inspection_type                 | ...
----+--------------------+-------+------------------------------------------------+-----
... | ...                | ...   | ...                                            | ...
... | SCHNIPPERS         | 27    | Cycle Inspection / Initial Inspection          | ...
... | ATOMIC WINGS       |       | Administrative Miscellaneous / Re-inspection   | ...
... | WING LING          | 44    | Cycle Inspection / Initial Inspection          | ...
... | JUAN VALDEZ CAFE   | 24    | Cycle Inspection / Initial Inspection          | ...
... | FULTON GRAND       | 22    | Cycle Inspection / Initial Inspection          | ...
... | ...                | ...   | ...                                            | ...
```

Representations for missing values:

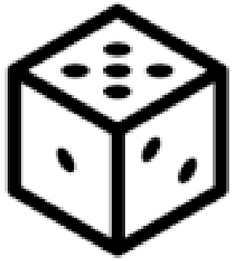
- `NULL` (general)
- `''` - empty string (used for string columns)

Causes of missing data

What causes missing data? 

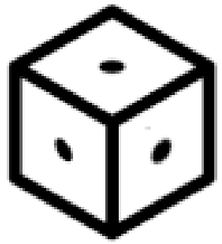
-  human error
-  systematic issues

Types of missing data



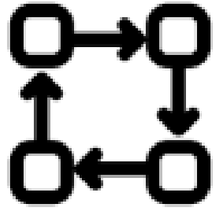
Missing Completely at Random

(MCAR)



Missing at Random

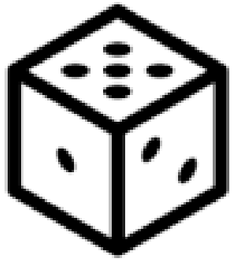
(MAR)



Missing Not at Random

(MNAR)

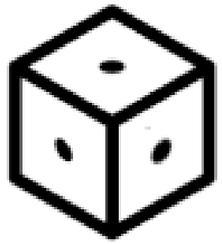
Types of missing data



Missing Completely at Random

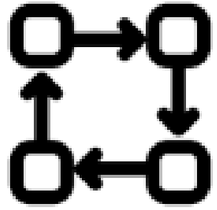
(MCAR)

No systematic relationship between missing data and other values



Missing at Random

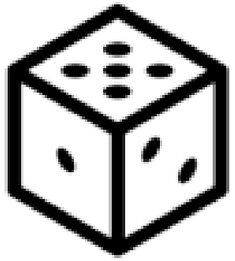
(MAR)



Missing Not at Random

(MNAR)

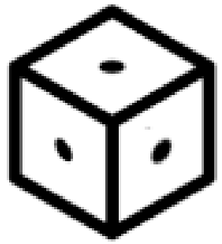
Types of missing data



Missing Completely at Random

(MCAR)

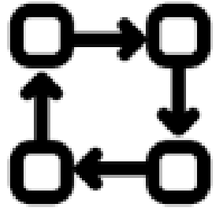
No systematic relationship between missing data and other values



Missing at Random

(MAR)

Systematic relationship between missing data and other observed values



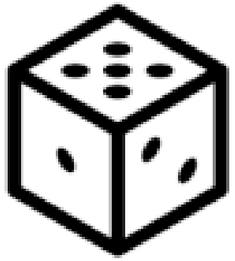
Missing Not at Random

(MNAR)

Types of missing data

```
... |       name        | score |                 inspection_type                  | ...
----+-------------------+-------+--------------------------------------------------+-----
... | ...               | ...   | ...                                              | ...
... | SCHNIPPERS        | 27    | Cycle Inspection / Initial Inspection            | ...
... | ATOMIC WINGS      |       | Administrative Miscellaneous / Re-inspection     | ...
... | WING LING         | 44    | Cycle Inspection / Initial Inspection            | ...
... | JUAN VALDEZ CAFE  | 24    | Cycle Inspection / Initial Inspection            | ...
... | FULTON GRAND      | 22    | Cycle Inspection / Initial Inspection            | ...
... | ...               | ...   | ...                                              | ...
```

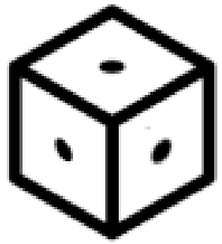
Types of missing data



Missing Completely at Random

(MCAR)

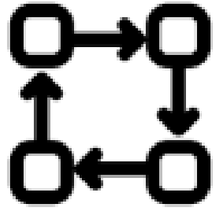
No systematic relationship between missing data and other values



Missing at Random

(MAR)

Systematic relationship between missing data and other observed values



Missing Not at Random

(MNAR)

Systematic relationship between missing data and unobserved values

Identifying missing data

```
SELECT
  *
FROM
  restaurant_inspection
WHERE
  score IS NULL;
```

```
SELECT
  COUNT(*)
FROM
  restaurant_inspection
WHERE
  score IS NULL;
```

Identifying missing data

```
SELECT
  inspection_type,
  COUNT(*) as count
FROM
  restaurant_inspection
WHERE
  score IS NULL
GROUP BY
  inspection_type
ORDER BY
  count DESC;
```

```
                inspection_type                 | count
------------------------------------------------+-------
Administrative Miscellaneous / Initial Inspection |   104
Smoke-Free Air Act / Initial Inspection           |    29
Calorie Posting / Initial Inspection              |    22
Administrative Miscellaneous / Re-inspection      |    22
Trans Fat / Initial Inspection                    |    18
Smoke-Free Air Act / Re-inspection                |     7
Trans Fat / Re-inspection                         |     3
```

Rectifying missing data

- Best option: locate and add missing values
  - May not be feasible
  - May not be worthwhile
- Provide a value (average, median, etc)
- Exclude records

Replacing missing values with COALESCE()

`COALESCE(arg1, [arg2, ...])`

```
SELECT
  name,
  COALESCE(score, -1),
  inspection_type
FROM
  restaurant_inspection;
```

Replacing missing values with COALESCE()

```
... |       name        | score |                  inspection_type                  | ...
----+-------------------+-------+---------------------------------------------------+-----
... | ...               | ...   | ...                                               | ...
... | SCHNIPPERS        | 27    | Cycle Inspection / Initial Inspection             | ...
... | ATOMIC WINGS      | -1    | Administrative Miscellaneous / Re-inspection      | ...
... | WING LING         | 44    | Cycle Inspection / Initial Inspection             | ...
... | JUAN VALDEZ CAFE  | 24    | Cycle Inspection / Initial Inspection             | ...
... | FULTON GRAND      | 22    | Cycle Inspection / Initial Inspection             | ...
... | ...               | ...   | ...                                               | ...
```

Let's practice!

CLEANING DATA IN POSTGRESQL DATABASES

Handling duplicated data

CLEANING DATA IN POSTGRESQL DATABASES

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

datacamp

Duplicate data

- Database should not store duplicate records
- Wastes storage resources
- Potentially distorts analysis

Detecting duplicated data

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

Detecting duplicated data

```
SELECT
  camis
FROM
  restaurant_inspection
GROUP BY
  camis
HAVING
  COUNT(*) > 1;
```

```
579
```

Detecting duplicated data

```
SELECT
  camis,
  name,
  boro
FROM
  restaurant_inspection
GROUP BY
  camis, name, boro
HAVING
  COUNT(*) > 1;
```

```
579
```

```
SELECT
  camis,
  name,
  boro,
  inspection_date
FROM
  restaurant_inspection
GROUP BY
  camis, name, boro, inspection_date
HAVING
  COUNT(*) > 1;
```

```
83
```

Detecting duplicated data

```
SELECT
  camis,
  name,
  boro,
  inspection_date,
  violation_code
FROM
  restaurant_inspection
GROUP BY
  camis, name, boro, inspection_date, violation_code
HAVING
  COUNT(*) > 1;
```

```
0
```

Detecting duplicated data

```
camis    |           name             |   boro    | inspection_date | violation_code | .
---------+----------------------------+-----------+-----------------+----------------+--
...      | ...                        | ...       | ...             | ...            | .
41659848 | LA BRISA DEL CIBAO         | Queens    | 01/30/2018      | 04L            | .
40961447 | MESON SEVILLA RESTAURANT   | Manhattan | 03/19/2019      | 10F            | .
41630358 | FAY DA BAKERY              | Queens    | 03/07/2019      | 06E            | .
41659848 | LA BRISA DEL CIBAO         | Queens    | 01/30/2018      | 04L            | .
...      | ...                        | ...       | ...             | ...            | .
```

The ROW_NUMBER() function

`ROW_NUMBER() OVER()`

```
ROW_NUMBER() OVER(
  PARTITION BY
      col1, col2, ...
  ORDER BY
      colA, colB, ...
)
```

```
camis    |           name            |   boro    | inspection_date | violation_code | row_number | ...
---------+---------------------------+-----------+-----------------+----------------+------------+----
...      | ...                       | ...       | ...             | ...            | ...        | ...
41659848 | LA BRISA DEL CIBAO        | Queens    | 01/30/2018      | 04L            | 1          | ...
41659848 | LA BRISA DEL CIBAO        | Queens    | 01/30/2018      | 04L            | 2          | ...
40961447 | MESON SEVILLA RESTAURANT  | Manhattan | 03/19/2019      | 10F            | 1          | ...
41630358 | FAY DA BAKERY             | Queens    | 03/07/2019      | 06E            | 1          | ...
...      | ...                       | ...       | ...             | ...            | ...        |...
```

Enumerating duplicate rows

```
SELECT
  camis,
  name,
  boro,
  inspection_date,
  violation_code,
  ROW_NUMBER() OVER(
      PARTITION BY
        camis,
        name,
        boro,
        inspection_date,
        violation_code
  ) - 1 AS duplicate
FROM
  restaurant_inspection;
```

camis	name	boro	inspection_date	violation_code	duplicate
...	...	...	...	...	...
40961447	MESON SEVILLA RESTAURANT	Manhattan	03/19/2019	10F	0
41630358	FAY DA BAKERY	Queens	03/07/2019	06E	0
41630358	FAY DA BAKERY	Queens	03/07/2019	06E	1
41630358	FAY DA BAKERY	Queens	03/07/2019	06E	2
41659848	LA BRISA DEL CIBAO	Queens	01/30/2018	04L	0
41659848	LA BRISA DEL CIBAO	Queens	01/30/2018	04L	1
...	...	...	...	...	...

Enumerating duplicate rows

```
SELECT
  camis, name, boro, inspection_date, violation_code,
  ROW_NUMBER() OVER(
      PARTITION BY camis, name, boro, inspection_date, violation_code
  ) - 1 AS duplicate
FROM
  restaurant_inspection;
```

```
camis    |           name           |   boro    | inspection_date | violation_code | duplicate  | ...
---------+--------------------------+-----------+-----------------+----------------+------------+----
 ...     | ...                      | ...       | ...             | ...            | ...        | ...
40961447 | MESON SEVILLA RESTAURANT | Manhattan | 03/19/2019      | 10F            | 0          | ...
41630358 | FAY DA BAKERY            | Queens    | 03/07/2019      | 06E            | 0          | ...
41630358 | FAY DA BAKERY            | Queens    | 03/07/2019      | 06E            | 1          | ...
41630358 | FAY DA BAKERY            | Queens    | 03/07/2019      | 06E            | 2          | ...
41659848 | LA BRISA DEL CIBAO       | Queens    | 01/30/2018      | 04L            | 0          | ...
41659848 | LA BRISA DEL CIBAO       | Queens    | 01/30/2018      | 04L            | 1          | ...
 ...     | ...                      | ...       | ...             | ...            | ...        | ...
```

Resolving impartial duplicates

Impartial duplicate - column values are duplicated with ambiguity where values differ

camis	name	inspection_date	violation_code	score	...
...	...	...	...	...	...
50038736	DON NICO'S	03/29/2018	09B	26	...
50038736	DON NICO'S	03/29/2018	09B	18	...
50033304	ASTORIA PIZZA	12/18/2019	02B	16	...
50081658	IRVING FARMS	12/13/2018	06F	9	...
50033733	ICHIBANTEI	02/12/2019	10B	12	...
...	...	...	...	...	...

Resolving impartial duplicates

Compute replacement from aggregate function (`AVERAGE()` , `MIN()` , `MAX()` , etc.)

```
SELECT
    camis,
    name,
    inspection_date,
    violation_code,
    AVG(score) AS score
FROM
    restaurant_inspection
GROUP BY
    camis,
    name,
    inspection_date,
    violation_code
HAVING
    COUNT(*) > 1;
```

Resolving impartial duplicates

```
camis    |      name        | inspection_date | violation_code | score | ...
---------+------------------+-----------------+----------------+-------+-----
...      | ...              | ...             | ...            | ...   | ...
50038736 | DON NICO'S       | 03/29/2018      | 09B            | 22.0  | ...
...      | ...              | ...             | ...            | ...   | ...
```

Let's practice!

CLEANING DATA IN POSTGRESQL DATABASES

Detecting invalid values

CLEANING DATA IN POSTGRESQL DATABASES

Darryl Reeves, Ph.D.
Assistant Professor, Long Island University - Brooklyn

datacamp

Invalid data values

camis	name	inspection_date	score	inspection_type	...
...	...	...	...	...	...
41659848	LA BRISA DEL CIBAO	01/30/2018	20	Cycle Inspection / Initial Inspection	...
40961447	MESON SEVILLA RESTAURANT	03/19/2019	50	Cycle Inspection / Initial Inspection	...
50063071	WA BAR	05/23/2018	15	Cycle Inspection / Initial Inspection	...
50034992	EMPANADAS MONUMENTAL	06/21/2019	17	Cycle Inspection / Re-inspection	...
50095871	ALPHONSO'S PIZZERIA & TRATTORIA	01/16/2020	10	Cycle Inspection / Initial Inspection	...
...	...	...	...	...	...

camis	name	inspection_date	score	inspection_type	...
...	...	...	...	...	...
41104041	THE SPARROW TAVERN	09/17/2019	13	Cycle Inspection / Initial Inspection	...
50016937	BURGER KING	09/14/2018	12	Cycle Inspection / Re-inspection	...
50066469	DARBAR'S CHICKEN & RIBS	08/07/2017	11	Pre-permit (Operational) / Reopening Inspection	...
41195691	F & J PINE RESTAURANT	05/02/2019	26	Cycle Inspection / Initial Inspection	...
50015706	EL RINCONCITO DE LOS SABORES	12/18/2019	A	Cycle Inspection / Initial Inspection	...
...	...	...	...	...	...

Handling invalid data with pattern matching

```
SELECT
  camis,
  name,
  inspection_date,
  score
FROM
  restaurant_inspection
WHERE
  score NOT SIMILAR TO '\d+';
```

Handling invalid data with pattern matching

- Query only restricts non-digit characters
- No restriction on length of value

```
SELECT
  camis,
  name,
  inspection_date,
  score
FROM
  restaurant_inspection
WHERE
  score NOT SIMILAR TO '\d{1}' AND
  score NOT SIMILAR TO '\d{2}' AND
  score NOT SIMILAR TO '\d{3}';
```

Using type constraints

- Column contains integer values
- Column should not allow non-integers

```
ALTER TABLE restaurant_inspection
ALTER COLUMN score TYPE SMALLINT USING score::smallint;
```

- `SMALLINT` : values from -32,768 to 32,767
- `USING` clause specifies conversion of previous values

Review: Basics of Regular Expressions

Metacharacter	Usage	Example RE	Example Match
`\d`	matches a digit (0-9)	`\d\d\d`	'345'
`?`	matches 0 or 1 of previous character	`x\d?`	'x5'
`+`	matches one or more of previous character	`\d+`	'10'
`*`	matches any character 0 or more times	`\d*`	'3081'
`[]`	matches any character inside of the brackets	`[a-z]`	'f'

Type constraints enable range constraints

```
ALTER TABLE restaurant_inspection
ALTER COLUMN score TYPE SMALLINT USING score::smallint;

SELECT
  camis,
  name,
  inspection_date,
  score
FROM
  restaurant_inspection
WHERE
  score < 0;
```

Type constraints enable range constraints

```
ALTER TABLE restaurant_inspection
ALTER COLUMN score TYPE SMALLINT USING score::smallint;

SELECT
  camis,
  name,
  inspection_date,
  score
FROM
  restaurant_inspection
WHERE
  score <= -1;
```

Type constraints enable range constraints

```
ALTER TABLE restaurant_inspection
ALTER COLUMN score TYPE SMALLINT USING score::smallint;

SELECT
  camis,
  name,
  inspection_date,
  score
FROM
  restaurant_inspection
WHERE
  score < 0 OR
  score > 100;
```

Type constraints enable range constraints

```
ALTER TABLE restaurant_inspection
ALTER COLUMN score TYPE SMALLINT USING score::smallint;

SELECT
  camis,
  name,
  inspection_date,
  score
FROM
  restaurant_inspection
WHERE
  score < 0 OR
  score >= 101;
```

The BETWEEN operator

```
SELECT
  camis, name, inspection_date, score
FROM
  restaurant_inspection
WHERE
  score NOT BETWEEN 0 AND 100;
```

```
 camis    |           name            | inspection_date | score
----------+---------------------------+-----------------+-------
...       | ...                       | ...             |   ...
41702543  | TROPICAL GRILL            | 05/14/2018      |   109
50074058  | PAD THAI                  | 08/01/2018      |   101
50085349  | DON CHILE MEXICAN GRILL   | 12/04/2018      |   124
50092932  | ENERGY JUICE BAR          | 06/24/2019      |   102
41702543  | TROPICAL GRILL            | 05/14/2018      |   109
50034653  | KAI FAN ASIAN CUISINE     | 12/06/2019      |    -1
...       | ...                       | ...             |   ...
```

Let's practice!

CLEANING DATA IN POSTGRESQL DATABASES

Detecting inconsistent data

CLEANING DATA IN POSTGRESQL DATABASES

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

datacamp

Inconsistent data

- Certain restaurant inspection rules
- `score` corresponds to number of violations 

```
camis    |             name                |  score | ...
---------+---------------------------------+--------+-----
...      | ...                             | ...    | ...
41659848 | LA BRISA DEL CIBAO              | 20     | ...
40961447 | MESON SEVILLA RESTAURANT        | 50     | ...
50063071 | WA BAR                          | 15     | ...
...      | ...                             | ...    | ...
```

- `A` (0 to 13), `B` (14 to 27), `C` (28+)
- Scenarios for grades:
  - `A` on initial inspection
  - Re-inspection with `A`, `B`, or `C`

[1] https://www1.nyc.gov/assets/doh/downloads/pdf/rii/restaurant-grading-faq.pdf

Checking rules with SQL

- Interdependent can introduce inconsistency
- Rules can be encoded in SQL
- `A` given for score from 0 to 13

```
SELECT
    camis,
    grade,
    grade_date,
    score,
    inspection_type
FROM
    restaurant_inspection
WHERE
    grade = 'A' AND
    score NOT BETWEEN 0 AND 13;
```

```
0
```

Checking rules with SQL

- `B` given for score from 14 to 27

```
SELECT
    camis,
    grade,
    grade_date,
    score,
    inspection_type
FROM
    restaurant_inspection
WHERE
    grade = 'B' AND
    score NOT BETWEEN 14 AND 27;
```

```
 camis    | grade | grade_date | score |         inspection_type
----------+-------+------------+-------+---------------------------------
 50034653 | B     | 12/06/2019 | -1    | Cycle Inspection / Re-inspection
```

Checking rules with SQL

```
SELECT
    camis, grade, grade_date, score, inspection_type FROM
    restaurant_inspection
WHERE
    (grade = 'A' OR grade = 'B' OR grade = 'C') AND
    inspection_type LIKE '%Reopening%';
```

```
camis    | grade | grade_date | score |                 inspection_type                  | ...
---------+-------+------------+-------+--------------------------------------------------+-----
...      | ...   | ...        | ...   | ...                                              | ...
50005784 | C     | 05/29/2019 | 14    | Cycle Inspection / Reopening Inspection          | ...
50091190 | C     | 07/12/2019 | 7     | Pre-permit (Operational) / Reopening Inspection  | ...
40395023 | C     | 09/13/2019 | 8     | Cycle Inspection / Reopening Inspection          | ...
50037770 | C     | 10/26/2018 | 11    | Cycle Inspection / Reopening Inspection          | ...
50036406 | C     | 07/10/2018 | 20    | Cycle Inspection / Reopening Inspection          | ...
...      | ...   | ...        | ...   | ...                                              | ...
```

[1] https://www1.nyc.gov/assets/doh/downloads/pdf/rii/restaurant-grading-faq.pdf

Data cleaning insights

- Diversity of approaches
- Careful thought required
- Domain knowledge is key
  - Which values are valid
  - Reasons for duplication
  - Appropriate fill-in values

Let's practice!

CLEANING DATA IN POSTGRESQL DATABASES

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
...      |   ...
```

```
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('month' 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

Combining columns

CLEANING DATA IN POSTGRESQL DATABASES

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

datacamp

Combining columns (an example)

Concatenation

```
name                          |  boro   | building |      street      | zip_code | ...
------------------------------+---------+----------+------------------+----------+-----
...                           | ...     | ...      | ...              | ...      | ...
DARBAR'S CHICKEN & RIBS       | Queens  | 12609    | LIBERTY AVE      | 11419    | ...
F & J PINE RESTAURANT         | Bronx   | 1913     | BRONXDALE AVENUE | 10462    | ...
EL RINCONCITO DE LOS SABORES  | Queens  | 13933    | 89TH AVE         | 11435    | ...
DON NICO'S                    | Queens  | 9014     | 161ST ST         | 11432    | ...
ASTORIA PIZZA                 | Queens  | 3204B    | 30TH AVE         | 11102    | ...
...                           | ...     | ...      | ...              | ...      | ...
```

```
Restaurant Name
Street Address
Boro, NY Zipcode
```

Joining values with CONCAT()

- `CONCAT(string1 [, string2, string3, ...])`
- `CONCAT('data', 'cleaning', 'is', 'fun')` → `datacleaningisfun`
- `CONCAT('data', ' ', 'cleaning', ' ', 'is', ' ', 'fun')` → `data cleaning is fun`

Joining values with CONCAT()

```
SELECT
  CONCAT(
      name, E'\n',
      building, ' ', street, E'\n',
      boro, ', NY ', zip_code
  ) AS mailing_address
FROM
  restaurant_inspection;
```

```
mailing_address
---------------------------
DARBAR'S CHICKEN & RIBS     +
12609 LIBERTY AVE           +
Queens, NY 11419
F & J PINE RESTAURANT       +
1913 BRONXDALE AVENUE       +
Bronx, NY 10462
EL RINCONCITO DE LOS SABORES+
13933 89TH AVE              +
Queens, NY 11435
DON NICO'S                  +
9014 161ST ST               +
Queens, NY 11432
ASTORIA PIZZA               +
3204B 30TH AVE              +
Queens, NY 11102
```

Joining values with CONCAT()

```
name                    |  boro     | building |      street                | zip_code | ...
------------------------+-----------+----------+----------------------------+----------+-----
...                     | ...       | ...      | ...                        | ...      | ...
IRVING FARMS            | Queens    |          | CENTRAL TERMINAL BUILDING  | 11371    | ...
DON PEPIS DELICATESSEN  | Manhattan |          | AMTRAK LEVEL               | 10001    | ...
DUNKIN'                 | Queens    |          | CENTRAL TERMINAL BLDG      | 11371    | ...
                        | Queens    | 17111    | JAMAICA AVE                | 11432    | ...
                        | Brooklyn  | 1489     | FULTON STREET              | 11216    | ...
...                     | ...       | ...      | ...                        | ...      | ...
```

Joining values with CONCAT()

```
SELECT
  CONCAT(
      name, E'\n',
      building, ' ', street, E'\n',
      boro, ', NY ', zip_code
  ) AS mailing_address
FROM
  restaurant_inspection;
```

```
     mailing_address
---------------------------
IRVING FARMS              +
 CENTRAL TERMINAL BUILDING+
Queens, NY 11371
DON PEPIS DELICATESSEN    +
 AMTRAK LEVEL             +
Manhattan, NY 10001
DUNKIN'                   +
 CENTRAL TERMINAL BLDG    +
Queens, NY 11371
                          +
17111 JAMAICA AVE         +
Queens, NY 11432
                          +
1489 FULTON STREET        +
Brooklyn, NY 11216
```

Joining values with ||

- `string1 || string2 [ || string3 || ...]`

```
SELECT 'data' || ' ' || 'cleaning' || ' ' || 'is' || ' ' || 'fun';
```

```
data cleaning is fun
```

`NULL` valued arguments → `NULL` value

```
SELECT
  name || E'\n' ||
  building || ' ' || street || E'\n'
  || boro || ', NY ' || zip_code AS mailing_address
FROM
  restaurant_inspection
```

Joining values with ||

```
name              |    mailing_address
------------------+-----------------------
SCHNIPPERS        | SCHNIPPERS           +
                  | 570 LEXINGTON AVENUE +
                  | Manhattan, NY 10022
ATOMIC WINGS      |
WING LING         | WING LING            +
                  | 159B EAST  170 STREET+
                  | Bronx, NY 10452
JUAN VALDEZ CAFE  | JUAN VALDEZ CAFE     +
                  | 140 EAST   57 STREET +
                  | Manhattan, NY 10022
FULTON GRAND      | FULTON GRAND         +
                  | 1011 FULTON STREET   +
                  | Brooklyn, NY 11238
```

Let's practice!

CLEANING DATA IN POSTGRESQL DATABASES

Splitting column data

CLEANING DATA IN POSTGRESQL DATABASES

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

datacamp

Splitting columns

camis	inspection_date	violation	...
...	...	...	...
50038736	03/29/2018	09B Thawing procedures	...
50033304	12/18/2019	02B Hot food item not held at or above 140° ...	...
50081658	12/13/2018	06F Wiping cloths soiled or not stored in sa...	...
50033733	02/12/2019	10B Plumbing not properly installed or maint...	...
40559634	08/22/2017	04N Filth flies or food/refuse/sewage-associ...	...
...	...	...	...

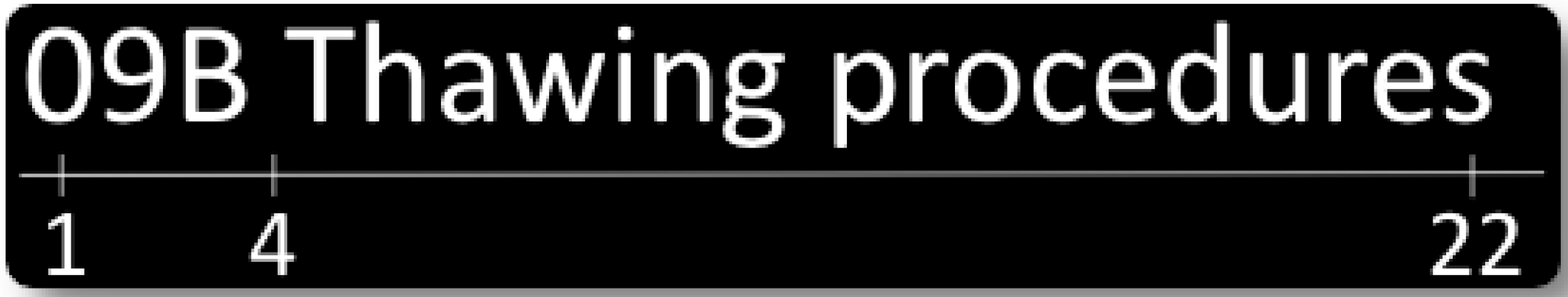
Finding substring starting position with STRPOS()

`STRPOS(source_string, search_string)`


```
SELECT
  STRPOS('09B Thawing procedures', ' ');
```

```
4
```

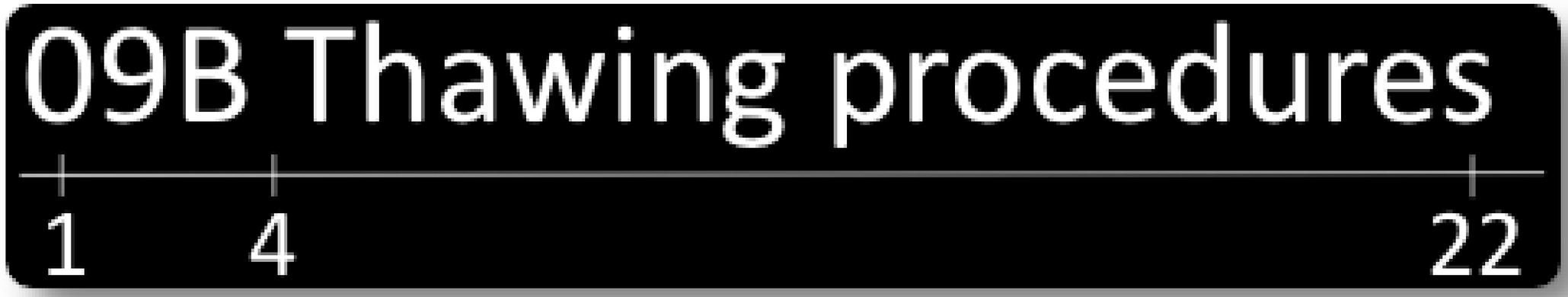
Finding substring starting position with STRPOS()




```
SELECT
  STRPOS('09B Thawing procedures', '?');
```

```
0
```

Finding substring starting position with STRPOS()




```
SELECT
  STRPOS('09B Thawing procedures', ' ');
```

```
4
```

Extracting a substring using SUBSTRING()

```
SUBSTRING(source_string FROM start_pos FOR num_chars)
```

Extracting a substring using SUBSTRING()

`SUBSTRING('Homerun' FROM 1 FOR 4)` → `Home`


```
SELECT
  SUBSTRING(
      '09B Thawing procedures'
      FROM 1
      FOR STRPOS('09B Thawing procedures', ' ') - 1
  );
```

```
09B
```

Extracting a substring using SUBSTRING()

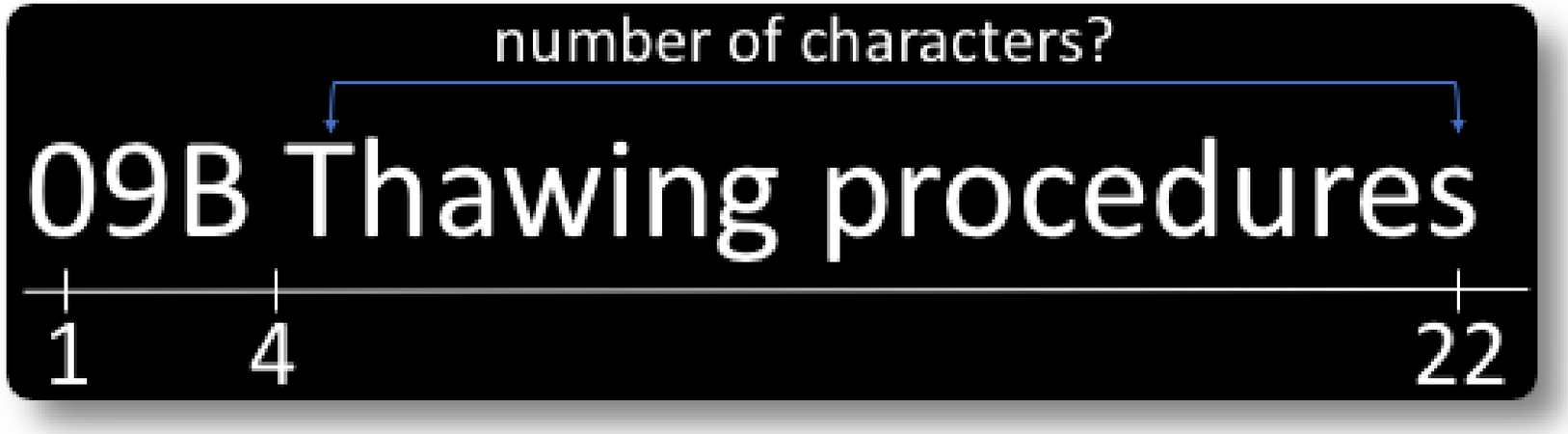
Requirements:

- Violation description start position
- Number of characters in the description

```
SELECT
  STRPOS('09B Thawing procedures', ' ') + 1;
```

```
5
```

Calculating the length of a string with LENGTH()



`LENGTH(string)` → `INTEGER`

```
SELECT LENGTH('hello')
```

```
5
```

Calculating the length of a string with LENGTH()

`LENGTH('09B Thawing procedures')` → `22`

`STRPOS('09B Thawing procedures', ' ')` → `4`

`LENGTH('09B Thawing procedures') - STRPOS('09B Thawing procedures', ' ')` → `18`

`LENGTH('Thawing procedures')` → `18`

Calculating the length of a string with LENGTH()

```
SELECT
    LENGTH('09B Thawing procedures') -
    STRPOS('09B Thawing procedures', ' ');
```

```
18
```

Putting the pieces together

```
SELECT
    SUBSTRING(
      '09B Thawing procedures'
      FROM
        STRPOS('09B Thawing procedures', ' ')
        + 1
      FOR
        LENGTH('09B Thawing procedures')
        - STRPOS('09B Thawing procedures', ' ')
    );
```

```
Thawing procedures
```

Splitting the violation column

```
SELECT
    camis,
    inspection_date,

    SUBSTRING(
      violation
      FROM 1
      FOR STRPOS(violation, ' ') - 1
    ) AS violation_code,

    SUBSTRING(
      violation
      FROM STRPOS(violation, ' ') + 1
      FOR LENGTH(violation) - STRPOS(violation, ' ')
    ) AS violation_description
FROM
    restaurant_inspection;
```

Splitting the violation column

camis	inspection_date	violation	...
...	...	...	...
50038736	03/29/2018	09B Thawing procedures	...
50033304	12/18/2019	02B Hot food item not held at or above 140° ...	...
50081658	12/13/2018	06F Wiping cloths soiled or not stored in sa...	...
50033733	02/12/2019	10B Plumbing not properly installed or maint...	...
40559634	08/22/2017	04N Filth flies or food/refuse/sewage-associ...	...
...	...	...	...

Splitting the violation column

camis	inspection_date	violation_code	violation_description	...
...	...	...	...	...
50038736	03/29/2018	09B	Thawing procedures	...
50033304	12/18/2019	02B	Hot food item not held at or above 140° ...	...
50081658	12/13/2018	06F	Wiping cloths soiled or not stored in sa...	...
50033733	02/12/2019	10B	Plumbing not properly installed or maint...	...
40559634	08/22/2017	04N	Filth flies or food/refuse/sewage-associ...	...
...	...	...	...	...

Let's practice!

CLEANING DATA IN POSTGRESQL DATABASES

Splitting data with delimiters

CLEANING DATA IN POSTGRESQL DATABASES

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

datacamp

Splitting data into columns

```
camis    |         name          |               inspection_type                | ...
---------+-----------------------+----------------------------------------------+-----
...      | ...                   | ...                                          | ...
50084922 | JUICE POINT           | Cycle Inspection / Re-inspection             | ...
50075375 | ATOMIC WINGS          | Administrative Miscellaneous / Re-inspection | ...
50048685 | KENNEDY FRIED CHICKEN | Cycle Inspection / Re-inspection             | ...
50058910 | HUNGER PANG           | Pre-permit (Operational) / Re-inspection     | ...
50047834 | SUBWAY                | Smoke-Free Air Act / Re-inspection           | ...
...      | ...                   | ...                                          | ...
```

Value delimiter: `' / '`

```
sub_inspection_type          | count
-----------------------------+-------
Reopening Inspection         |    56
Re-inspection                |  1333
Initial Inspection           |  3488
Second Compliance Inspection |     2
Compliance Inspection        |    27
```

Splitting data into columns

```
camis    |         name          |               inspection_type                | ...
---------+-----------------------+----------------------------------------------+-----
...      | ...                   | ...                                          | ...
50084922 | JUICE POINT           | Cycle Inspection / Re-inspection             | ...
50075375 | ATOMIC WINGS          | Administrative Miscellaneous / Re-inspection | ...
50048685 | KENNEDY FRIED CHICKEN | Cycle Inspection / Re-inspection             | ...
50058910 | HUNGER PANG           | Pre-permit (Operational) / Re-inspection     | ...
50047834 | SUBWAY                | Smoke-Free Air Act / Re-inspection           | ...
...      | ...                   | ...                                          | ...
```

```
 camis   |         name          |     main_inspection_type     | sub_inspection_type | ...
---------+-----------------------+------------------------------+---------------------+-----
...      | ...                   | ...                          | ...                 | ...
50084922 | JUICE POINT           | Cycle Inspection             | Re-inspection       | ...
50075375 | ATOMIC WINGS          | Administrative Miscellaneous | Re-inspection       | ...
50048685 | KENNEDY FRIED CHICKEN | Cycle Inspection             | Re-inspection       | ...
50058910 | HUNGER PANG           | Pre-permit (Operational)     | Re-inspection       | ...
50047834 | SUBWAY                | Smoke-Free Air Act           | Re-inspection       | ...
...      | ...                   | ...                          | ...                 | ...
```

Splitting strings using SPLIT_PART()

- `SPLIT_PART(source_string, delimiter_string, part_number)`

```
SELECT
  SPLIT_PART('Cycle Inspection / Re-inspection', ' / ', 1);
```

```
Cycle Inspection
```

```
SELECT
  SPLIT_PART('Cycle Inspection / Re-inspection', ' / ', 2);
```

```
Re-inspection
```

Splitting strings using SPLIT_PART()

```
SELECT
  camis,
  name,
  SPLIT_PART(inspection_type, ' / ', 1) AS main_inspection_type,
  SPLIT_PART(inspection_type, ' / ', 2) AS sub_inspection_type
FROM
  restaurant_inspection;
```

```
camis    |         name           |       main_inspection_type    | sub_inspection_type | ...
---------+------------------------+-------------------------------+---------------------+-----
...      | ...                    | ...                           | ...                 | ...
50084922 | JUICE POINT            | Cycle Inspection              | Re-inspection       | ...
50075375 | ATOMIC WINGS           | Administrative Miscellaneous | Re-inspection       | ...
50048685 | KENNEDY FRIED CHICKEN | Cycle Inspection              | Re-inspection       | ...
50058910 | HUNGER PANG            | Pre-permit (Operational)      | Re-inspection       | ...
50047834 | SUBWAY                 | Smoke-Free Air Act            | Re-inspection       | ...
...      | ...                    | ...                           | ...                 | ...
```

Splitting data into rows

```
 camis   |         name          | cuisine_description | ...
---------+-----------------------+---------------------+-----
...      | ...                   | ...                 | ...
50066768 | FIRST LAMB SHABU      | Chinese             | ...
41450971 | GIOVANNI'S RESTAURANT | Pizza/Italian       | ...
41628459 | KFC                   | Chicken             | ...
50043003 | BANGIA                | Korean              | ...
41418978 | BAGEL EXPRESS III     | Bagels/Pretzels     | ...
...      | ...                   | ...                 | ...
```

```
camis   |         name          | cuisine_description | ...
-------+-----------------------+---------------------+-----
...      | ...                   | ...                 | ...
50066768 | FIRST LAMB SHABU      | Chinese             | ...
41450971 | GIOVANNI'S RESTAURANT | Pizza               | ...
41450971 | GIOVANNI'S RESTAURANT | Italian             | ...
41628459 | KFC                   | Chicken             | ...
50043003 | BANGIA                | Korean              | ...
41418978 | BAGEL EXPRESS III     | Bagels              | ...
41418978 | BAGEL EXPRESS III     | Pretzels            | ...
...      | ...                   | ...                 | ...
```

Splitting data with REGEXP_SPLIT_TO_TABLE()

`REGEXP_SPLIT_TO_TABLE(source, pattern)`

```
SELECT REGEXP_SPLIT_TO_TABLE('Pizza/Italian', '/');
```

```
Pizza
Italian
```

Splitting data with REGEXP_SPLIT_TO_TABLE()

```
SELECT
  camis,
  name,
  REGEXP_SPLIT_TO_TABLE(cuisine_description, '/') AS cuisine_description,
  ...
FROM
  restaurant_inspection;
```

```
camis    |         name          | cuisine_description | ...
---------+-----------------------+---------------------+-----
...      | ...                   | ...                 | ...
50066768 | FIRST LAMB SHABU      | Chinese             | ...
41450971 | GIOVANNI'S RESTAURANT | Pizza               | ...
41450971 | GIOVANNI'S RESTAURANT | Italian             | ...
41628459 | KFC                   | Chicken             | ...
50043003 | BANGIA                | Korean              | ...
41418978 | BAGEL EXPRESS III     | Bagels              | ...
41418978 | BAGEL EXPRESS III     | Pretzels            | ...
...      | ...                   | ...                 | ...
```

Enumerating the resulting rows

```
cuisine_num |  camis   |         name          | cuisine_description | ...
------------+----------+-----------------------+---------------------+-----
...         | ...      | ...                   | ...                 | ...
1           | 41418978 | BAGEL EXPRESS III     | Bagels              | ...
2           | 41418978 | BAGEL EXPRESS III     | Pretzels            | ...
1           | 41450971 | GIOVANNI'S RESTAURANT | Pizza               | ...
2           | 41450971 | GIOVANNI'S RESTAURANT | Italian             | ...
1           | 41628459 | KFC                   | Chicken             | ...
1           | 50043003 | BANGIA                | Korean              | ...
1           | 50066768 | FIRST LAMB SHABU      | Chinese             | ...
...         | ...      | ...                   | ...                 | ...
```

`ROW_NUMBER() OVER()`

`PARTITION BY col1, col2, ...`

`ORDER BY colA, colB, ...`

Enumerating the resulting rows

```
SELECT
  ROW_NUMBER() OVER (
    PARTITION BY
      -- group columns for numbering
      camis,
      name
    ORDER BY
      -- set ordering of results
      camis,
      name
  ) AS cuisine_num,
  *
FROM (
  SELECT
    camis,
    name,
    REGEXP_SPLIT_TO_TABLE(cuisine_description, '/')
      AS cuisine_description
FROM
  restaurant_inspection;
```

```
cuisine_num |  camis   |         name          | cuisine_description | ...
------------+----------+-----------------------+---------------------+-----
...         | ...      | ...                   | ...                 | ...
1           | 41418978 | BAGEL EXPRESS III     | Bagels              | ...
2           | 41418978 | BAGEL EXPRESS III     | Pretzels            | ...
1           | 41450971 | GIOVANNI'S RESTAURANT | Pizza               | ...
2           | 41450971 | GIOVANNI'S RESTAURANT | Italian             | ...
1           | 41628459 | KFC                   | Chicken             | ...
1           | 50043003 | BANGIA                | Korean              | ...
1           | 50066768 | FIRST LAMB SHABU      | Chinese             | ...
...         | ...      | ...                   | ...                 | ...
```

Let's practice!

CLEANING DATA IN POSTGRESQL DATABASES

Creating pivot tables

CLEANING DATA IN POSTGRESQL DATABASES

Darryl Reeves, Ph.D.
Assistant Professor, Long Island University - Brooklyn

datacamp

Multiple category records

name	inspection_type	grade	...
...	...	...	...
EMPANADAS MONUMENTAL	Cycle Inspection / Re-inspection	B	...
ALPHONSO'S PIZZERIA & TRATTORIA	Cycle Inspection / Initial Inspection	A	...
THE SPARROW TAVERN	Cycle Inspection / Initial Inspection	A	...
BURGER KING	Cycle Inspection / Re-inspection	A	...
ASTORIA PIZZA	Cycle Inspection / Re-inspection	B	...
...	...	...	...

Accessing inspection grades by type

```
SELECT
    inspection_type,
    grade,
    COUNT(grade)
FROM
    restaurant_inspection
WHERE
    grade IS NOT NULL
GROUP BY
    inspection_type,
    grade
ORDER BY
    inspection_type,
    grade;
```

Aggregated inspection results by type

inspection_type	grade	count
Cycle Inspection / Initial Inspection	A	1063
Cycle Inspection / Re-inspection	A	723
Cycle Inspection / Re-inspection	B	270
Cycle Inspection / Re-inspection	C	93
Cycle Inspection / Re-inspection	Z	29
Cycle Inspection / Reopening Inspection	C	8
Cycle Inspection / Reopening Inspection	P	26
Cycle Inspection / Reopening Inspection	Z	3
Pre-permit (Non-operational) / Initial Inspection	N	4
Pre-permit (Operational) / Initial Inspection	A	119
Pre-permit (Operational) / Initial Inspection	N	17
Pre-permit (Operational) / Re-inspection	A	79
Pre-permit (Operational) / Re-inspection	B	49
Pre-permit (Operational) / Re-inspection	C	13
Pre-permit (Operational) / Re-inspection	Z	9
Pre-permit (Operational) / Reopening Inspection	C	3
Pre-permit (Operational) / Reopening Inspection	P	3
Pre-permit (Operational) / Reopening Inspection	Z	1

Changing (pivoting) data orientation

inspection_type	A	B	C	N	P	Z
Cycle Inspection / Re-inspection	723	270	93	0	0	29
Cycle Inspection / Initial Inspection	1063	0	0	0	0	0
Pre-permit (Operational) / Reopening Inspection	0	0	3	0	3	1
Cycle Inspection / Reopening Inspection	0	0	8	0	26	3
Pre-permit (Non-operational) / Initial Inspection	0	0	0	4	0	0
Pre-permit (Operational) / Initial Inspection	119	0	0	17	0	0
Pre-permit (Operational) / Re-inspection	79	49	13	0	0	9

The FILTER clause

- Applies an aggregation over a subset of records
- Subset of records determined by accompanying `WHERE` clause
- Used in the `SELECT` list of a query

The FILTER clause

- Example: `AVG(qty_sold) FILTER (WHERE qty_sold > 1)`
- Format: `AGG_FUNC(expression) FILTER (WHERE condition)`
  - `AGG_FUNC()` - aggregate function

The pivot table query

```
SELECT
    summary_column,
    AGG(agg_column) FILTER (WHERE agg_column = PIVOT_VALUE_1) AS "pivot_column_1",
    AGG(agg_column) FILTER (WHERE agg_column = PIVOT_VALUE_2) AS "pivot_column_2",
    ...
    AGG(agg_column) FILTER (WHERE agg_column = PIVOT_VALUE_N) AS "pivot_column_N"
FROM
    source_table
GROUP BY
    summary_column;
```

The pivot table output

summary_column	pivot_column_1	pivot_column_2	...	pivot_column_N
summary_val_1	agg result for PV1	agg result for PV2		agg value for PVN
summary_val_2	agg result for PV1	agg result for PV2		agg value for PVN
...	...			...
summary_val_M	agg result for PV1	agg result for PV2	...	agg value for PVN

Pivoting restaurant inspection data

```
SELECT
    inspection_type,
    COUNT(grade) FILTER (WHERE grade = 'A') AS "A",
    COUNT(grade) FILTER (WHERE grade = 'B') AS "B",
    COUNT(grade) FILTER (WHERE grade = 'C') AS "C",
    COUNT(grade) FILTER (WHERE grade = 'N') AS "N",
    COUNT(grade) FILTER (WHERE grade = 'P') AS "P",
    COUNT(grade) FILTER (WHERE grade = 'Z') AS "Z"
FROM
    restaurant_inspections
WHERE
    grade IS NOT NULL
GROUP BY
    inspection_type;
```

Pivot table output for inspection data

inspection_type	A	B	C	N	P	Z
Cycle Inspection / Re-inspection	723	270	93	0	0	29
Cycle Inspection / Initial Inspection	1063	0	0	0	0	0
Pre-permit (Operational) / Reopening Inspection	0	0	3	0	3	1
Cycle Inspection / Reopening Inspection	0	0	8	0	26	3
Pre-permit (Non-operational) / Initial Inspection	0	0	0	4	0	0
Pre-permit (Operational) / Initial Inspection	119	0	0	17	0	0
Pre-permit (Operational) / Re-inspection	79	49	13	0	0	9

Let's practice!

CLEANING DATA IN POSTGRESQL DATABASES

Course wrap-up

CLEANING DATA IN POSTGRESQL DATABASES

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

datacamp

Course content

Chapter 1: Data cleaning basics

Chapter 2: Missing, duplicate, and invalid data

Chapter 3: Converting data

Chapter 4: Transforming data

Onward!

- Functions for Manipulating Data in PostgreSQL
- Reporting in SQL
- PostgreSQL Summary Stats and Window Functions
- Exploratory Data Analysis in SQL

Congratulations!

CLEANING DATA IN POSTGRESQL DATABASES