Handling missing data

CLEANING DATA IN POSTGRESQL DATABASES

SQL

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     | ...
... | ...               | ...   | ...                                       | ...
```
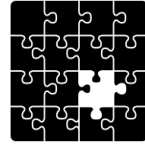
Representations for missing values:

- `NULL` (general)

- `''` - empty string (used for string columns)
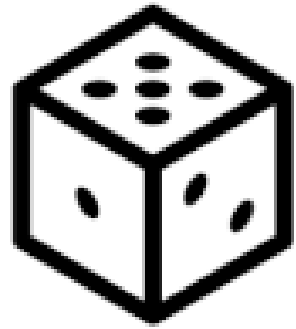
Causes of missing data

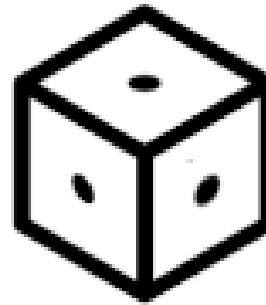## What causes missing data?

- human error

- systematic issues

Types of missing data

Missing Completely at Random

(MCAR)

Missing at Random
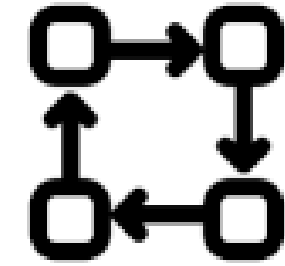
(MAR)

Missing Not at Random

(MNAR)

Types of missing data

Missing Completely at Random

(MCAR)

No systematic relationship between missing data and other values
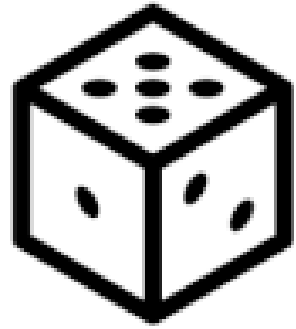
Missing at Random

(MAR)

Missing Not at Random

(MNAR)

Types of missing data
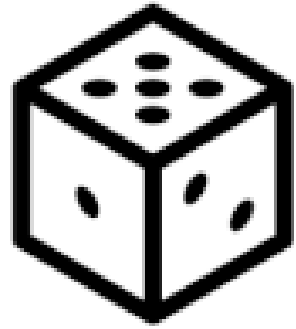
Missing Completely at Random
(MCAR)

No systematic relationship between missing data and other values
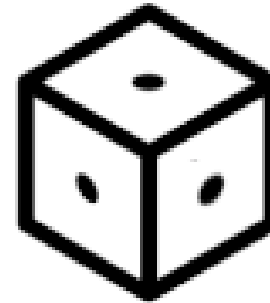
Missing at Random
(MAR)

*Systematic relationship between missing data and other **observed** values*
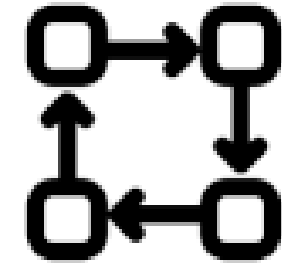
Missing Not at Random
(MNAR)

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
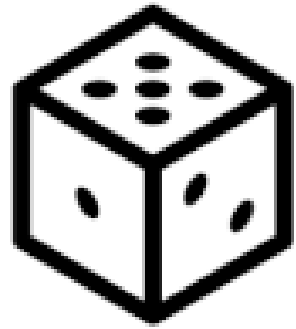
Missing Completely at Random

(MCAR)

No systematic relationship between missing data and other values

Missing at Random

(MAR)
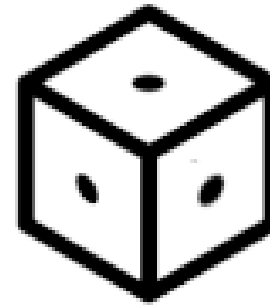
*Systematic relationship between missing data and other **observed** values*
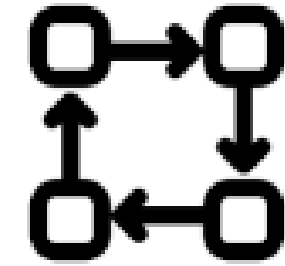
Missing Not at Random

(MNAR)

*Systematic relationship between missing data and **unobserved** values*

Identifying missing data

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

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

Identifying missing data

```sql
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, ...])
```

```sql
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

SQL

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

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

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

```
579
```

Detecting duplicated data

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

```
579
```

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

```
83
```

Detecting duplicated data

```sql
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

```sql
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

```sql
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.)

```sql
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

SQL

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

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

```sql
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

```sql
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

```sql
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

```sql
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

```sql
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

```sql
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

SQL

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

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

```sql
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

```sql
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

```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!

datacamp