Introduction to Cleaning Data

CLEANING DATA IN SQL SERVER DATABASES

Miriam Antona
Software Engineer

datacamp

Topics covered

- **Chapter 1**: Starting with Cleaning Data
- **Chapter 2**: Dealing with nulls, duplicate data, and dates
- **Chapter 3**: Dealing with out of range values, different data types, and pattern matching
- **Chapter 4**: Combining, splitting, and transforming data

Dataset: Monthly airline flights by USA airports 2014-2015

`airports`

airport_code	airport_name	airport_city	airport_state
MSP	Minneapolis-St Paul International	Minneapolis	Minnesota
JFK	John F. Kennedy International	New York City	New York
LAX	Los Angeles International	Los Angeles	California
DFW	Dallas/Fort Worth International	Dallas/Fort Worth	Texas
BOS	Logan International	Boston	Massachusetts
SFO	San Francisco International	San Francisco	Californiaa
ATL	Hartsfield-Jackson Atlanta International	Atlanta	Georgia
...	...	...	...

Dataset: Monthly airline flights by USA airports 2014-2015

`carriers`

code	name
YV	Mesa Airlines Inc.
AA	American Airlines Inc.
DL	Delta Air Lines Inc.
HA	Hawaiian Airlines Inc.
MQ	American Eagle Airlines Inc.
EV	ExpressJet Airlines Inc.
...	...

Dataset: Monthly airline flights by USA airports 2014-2015

`flight statistics`

registration_code	airport_code	carrier_code	canceled	on_time	delayed	...
...	...	...	...	...	...	...
000000119	JFK	AA	74	819	233	...
120	JFK	B6	438	1865	1010	...
000000121	JFK	HA	0	25	3	...
122	JFK	MQ	102	386	159	...
000000124	JFK	UA	22	296	88	...
000000125	JFK	US	15	191	63	...
000000126	JFK	VX	12	225	61	...
...	...	...	...	...	...	...

Dataset: Monthly airline flights by USA airports 2014-2015

`pilots`

pilot_code	pilot_name	pilot_surname	carrier_code	entry_date
1	Thomas	Peters	HA	2011-10-01
2	Hiroki	Konoe	MQ	2011-01-21
3	Arturo	Montero	UA	2012-12-28
4	David	Captain	US	2000-10-01
5	Ainhoa	Guerrera	VX	2000-10-05
6	Alvin	Andersen	OO	2012-01-15
7	William	Champy	F9	2011-03-15
...	...	...	...	...

Why is cleaning data important?

- Common to acquire **messy/dirty data** not ready for analysis
- **Lot of time** spent cleaning data vs. time spent analyzing data
- Cleaning process -> clear information

Filling numbers with leading zeros

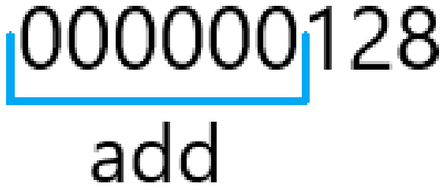
```
SELECT * FROM flight_statistics
```

registration_code	airport_code	carrier_code	canceled	on_time	delayed	...
...	...	...	...	...	...	...
000000119	JFK	AA	74	819	233	...
120	JFK	B6	438	1865	1010	...
000000121	JFK	HA	0	25	3	...
122	JFK	MQ	102	386	159	...
000000124	JFK	UA	22	296	88	...
000000125	JFK	US	15	191	63	...
000000126	JFK	VX	12	225	61	...
...	...	...	...	...	...	...

Filling numbers with leading zeros

VALID: 000000128 - until 9 digits

INVALID: 128



Filling numbers with leading zeros - Using REPLICATE and LEN

```
REPLICATE (string, integer)
```

Repeats a string a specified number of times.

Filling numbers with leading zeros - Using REPLICATE and LEN

```
REPLICATE (string, integer)
```

Repeats a string a specified number of times.

```
REPLICATE('0', 9 - LEN(registration_code))
```

Filling numbers with leading zeros - Using REPLICATE and LEN

```
REPLICATE (string, integer)
```

Repeats a string a specified number of times.

```
REPLICATE('0', 9 - LEN(registration_code))
```

```
-- registration_code: 120 => LEN(120) = 3
REPLICATE('0', 6)
```

Filling numbers with leading zeros - Using REPLICATE, LEN

`+` operator

```
SELECT
    REPLICATE('0', 9 - LEN(registration_code)) + registration_code AS registration_code
FROM flight_statistics
```

`CONCAT` - since SQL Server 2012

```
SELECT
    CONCAT(REPLICATE('0', 9 - LEN(registration_code)), registration_code) AS registration_code
FROM flight_statistics
```

Filling numbers with leading zeros - Using REPLICATE, LEN, and CONCAT

```
| registration_code |
|-------------------|
| ...               |
| 000000119         |
| 000000120         |
| 000000121         |
| 000000122         |
| 000000123         |
| 000000124         |
| 000000125         |
| 000000126         |
| ...               |
```

Filling numbers with leading zeros - Using FORMAT

```
FORMAT (value, format [, culture ] )
```

- Available since SQL Server 2012
- value: numeric, date and time

```
SELECT
    FORMAT(CAST(registration_code AS INT), '000000000') AS registration_code
FROM flight_statistics;
```

Filling numbers with leading zeros - Using FORMAT

```
| registration_code |
|-------------------|
| ...               |
| 000000119         |
| 000000120         |
| 000000121         |
| 000000122         |
| 000000123         |
| 000000124         |
| 000000125         |
| 000000126         |
| ...               |
```

Let's practice!

CLEANING DATA IN SQL SERVER DATABASES

Cleaning messy strings

CLEANING DATA IN SQL SERVER DATABASES

Miriam Antona
Software Engineer

datacamp

Removing additional spaces

```
SELECT * FROM carriers
```

```
| code | name                              |
|------|-----------------------------------|
| YV   | Mesa Airlines Inc.                |
| AA   |   American Airlines Inc.          |
| B6   |   JetBlue Airways                 |
| DL   |   Delta Air Lines Inc.            |
| HA   |   Hawaiian Airlines Inc.          |
| MQ   |   American Eagle Airlines Inc.    |
| EV   |  ExpressJet Airlines Inc.         |
| UA   |  United Air Lines Inc.            |
| US   |  US Airways Inc.                  |
| ...  | ...                               |
```

Removing additional spaces - TRIM

```
TRIM ( [characters ] string )
```

- Available since SQL Server 2017
- Removes any specified character from the start and end of a string
- Removes space character if we don't specify any character.

```
SELECT TRIM('   JetBlue Airways    ');
```

```
JetBlue Airways
```

Removing additional spaces - RTRIM and LTRIM

- For older versions than SQL Server 2017 -> `RTRIM` and `LTRIM`.

```
-- Removes all trailing spaces
RTRIM ( character_expression )
```

```
-- Removes all leading spaces
LTRIM ( character_expression )
```

```
SELECT LTRIM(RTRIM('   JetBlue Airways    '));
```

```
JetBlue Airways
```

Removing additional spaces

```
SELECT code, TRIM(name) AS name FROM carriers
```

```
SELECT code, LTRIM(RTRIM(name)) AS name FROM carriers
```

```
| code | name                          |
|------|-------------------------------|
| YV   | Mesa Airlines Inc.            |
| AA   | American Airlines Inc.        |
| B6   | JetBlue Airways               |
| DL   | Delta Air Lines Inc.          |
| HA   | Hawaiian Airlines Inc.        |
| US   | US Airways Inc.               |
| ...  | ...                           |
```

Unifying strings

```
SELECT * FROM airports
ORDER BY airport_state
```

airport_code	airport_name	airport_city	airport_state
...	...	...	...
MIA	Miami International	Miami	fl
TPA	Tampa International	Tampa	Fl
FLL	Fort Lauderdale-Hollywood International	Fort Lauderdale	FL
MCO	Orlando International	Orlando	Florida
...	...	...	...

Unifying strings - REPLACE

"Fl" / "fl" / "Florida" -> **"Florida"**

```
REPLACE ( string_to_replace , occurrences , string_replacement )
```

- Replaces all occurrences of a specified string with another string
- Performs comparisons based on the collation of the input
  - Use `COLLATE` to perform a comparison in a specific collation
  - Suppose the collation of our inputs is case insensitive

Unifying strings - REPLACE

```
SELECT
    airport_code, airport_name, airport_city,
    REPLACE(airport_state, 'FL', 'Florida') AS airport_state
FROM airports
ORDER BY airport_state
```

```
| airport_code | airport_name                          | airport_city    | airport_state |
|--------------|---------------------------------------|-----------------|---------------|
| ...          | ...                                   | ...             | ...           |
| MIA          | Miami International                   | Miami           | Florida       |
| TPA          | Tampa International                   | Tampa           | Florida       |
| FLL          | Fort Lauderdale-Hollywood International | Fort Lauderdale | Florida       |
| MCO          | Orlando International                 | Orlando         | Floridaorida  |
| ...          | ...                                   | ...             | ...           |
```

Unifying strings - REPLACE

```
SELECT
    airport_code, airport_name, airport_city,
    REPLACE
        (REPLACE(airport_state, 'FL', 'Florida'),
         'Floridaorida', 'Florida') AS airport_state
FROM airports
ORDER BY airport_state
```

```
| airport_code | airport_name                          | airport_city    | airport_state |
|--------------|---------------------------------------|-----------------|---------------|
| MCO          | Orlando International                 | Orlando         | Florida       |
| TPA          | Tampa International                   | Tampa           | Florida       |
| FLL          | Fort Lauderdale-Hollywood International | Fort Lauderdale | Florida       |
| MIA          | Miami International                   | Miami           | Florida       |
| ...          | ...                                   | ...             | ...           |
```

Unifying strings - REPLACE + CASE

```
SELECT airport_code, airport_name, airport_city,
    CASE
        WHEN airport_state <> 'Florida' THEN REPLACE(airport_state, 'FL', 'Florida')
        ELSE airport_state
    END AS airport_state
FROM airports
ORDER BY airport_state
```

```
| airport_code | airport_name                          | airport_city    | airport_state |
|--------------|---------------------------------------|-----------------|---------------|
| MCO          | Orlando International                 | Orlando         | Florida       |
| TPA          | Tampa International                   | Tampa           | Florida       |
| FLL          | Fort Lauderdale-Hollywood International | Fort Lauderdale | Florida       |
| MIA          | Miami International                   | Miami           | Florida       |
| ...          | ...                                   | ...             | ...           |
```

Unifying strings - REPLACE + UPPER

"Fl" / "fl" / "Florida" -> **"FL"**

```
SELECT
    airport_code, airport_name, airport_city,
    REPLACE(airport_state, 'Florida', 'FL') AS airport_state
FROM airports
ORDER BY airport_state
```

```
| airport_code | airport_name                          | airport_city    | airport_state |
|--------------|---------------------------------------|-----------------|---------------|
| MCO          | Orlando International                 | Orlando         | FL            |
| TPA          | Tampa International                   | Tampa           | Fl            |
| FLL          | Fort Lauderdale-Hollywood International | Fort Lauderdale | FL            |
| MIA          | Miami International                   | Miami           | fl            |
| ...          | ...                                   | ...             | ...           |
```

Unifying strings - REPLACE + UPPER

```
UPPER ( character_expression )
```

- Converts a given string to uppercase.

```
SELECT
    airport_code, airport_name, airport_city,
        UPPER(
              REPLACE(airport_state, 'Florida', 'FL')
        ) AS airport_state
FROM airports
ORDER BY airport_state
```

Unifying strings - REPLACE + UPPER

airport_code	airport_name	airport_city	airport_state
...	...	...	...
MCO	Orlando International	Orlando	FL
TPA	Tampa International	Tampa	FL
FLL	Fort Lauderdale-Hollywood International	Fort Lauderdale	FL
MIA	Miami International	Miami	FL
...	...	...	...

Let's practice!

CLEANING DATA IN SQL SERVER DATABASES

Comparing the similarity between strings

CLEANING DATA IN SQL SERVER DATABASES

Miriam Antona
Software Engineer

datacamp

Describing the problem

- Messy strings

```
| airport_state |
|---------------|
| Caalifornia   |
| California    |
| Californiaa   |
| Illinois      |
| Ilynois       |
| Tejas         |
| Texas         |
```

```
SOUNDEX
```

```
DIFFERENCE
```

SOUNDEX

```
SOUNDEX ( character_expression )
```

- Phonetic algorithm
- Returns four-character code
- Based on English language, but also works with many words in other languages

```
SELECT SOUNDEX('Illinois') AS soundex_code1;
SELECT SOUNDEX('Ilynois') AS soundex_code2;
SELECT SOUNDEX('California') AS soundex_code3;
```

```
| soundex_code1 |   | soundex_code2 |   | soundex_code3 |
|---------------|   |---------------|   |---------------|
| I452          |   | I452          |   | I416          |
```

SOUNDEX - how it works

Example: "Illinois"

- Writes the first letter of the word
- Replaces to zero(0) vowels and letters "h", "w", "y" to zero(0), after the first letter

"**I**llinois" -> I

"Ill**i**n**o**i**s**" -> Ill0n00s

SOUNDEX - how it works

Example: "Illinois"

- Replaces consonants after the first letter

SOUNDEX - how it works

Example: "Illinois"

- Replaces consonants after the first letter

Letters	Represented by
b, f, p, v	1
c, g, j, k, q, s, x, z	2
d, t	3
l	4
m, n	5
r	6

"Ill0n00s" -> I4405002

SOUNDEX - how it works

- Replaces same adjacent digits with one
- Removes all the zeros (0)
- If the letter's digit is the same as the first digit, it removes the first digit.
- Appends zeros if code contains less than 3 digits.
- Removes final digits if code has more than 3 digits.

"I**44**05**00**2" -> I40502

"I4**0**5**0**2" -> I452

"I452" (don't apply)

SOUNDEX - Exceptions

```
SELECT SOUNDEX('Arizona') AS soundex_code1;
SELECT SOUNDEX('Arkansas') AS soundex_code2;
```

```
| soundex_code1 |    | soundex_code2 |
|---------------|    |---------------|
| A625          |    | A625          |
```

SOUNDEX - checking similarities

```
SELECT DISTINCT A1.airport_state
FROM airports A1
INNER JOIN airports A2
    ON SOUNDEX(A1.airport_state) = SOUNDEX(A2.airport_state)
    AND A1.airport_state <> A2.airport_state
```

```
| airport_state |
|---------------|
| Caalifornia   |
| California    |
| Californiaa   |
| Illinois      |
| Ilynois       |
| New Jersey    |
| New York      |
| Tejas         |
| Texas         |
```

SOUNDEX - checking similarities

```
SELECT DISTINCT A1.airport_state
FROM airports A1
INNER JOIN airports A2
    ON SOUNDEX(REPLACE(A1.airport_state, ' ', '')) = SOUNDEX(REPLACE(A2.airport_state, ' ', ''))
    AND A1.airport_state <> A2.airport_state
```

"New York" -> "NewYork"

```
| airport_state |
|---------------|
| Caalifornia   |
| California    |
| Californiaa   |
| Illinois      |
| Ilynois       |
| Tejas         |
| Texas         |
```

DIFFERENCE

```
DIFFERENCE ( character_expression , character_expression )
```

- Compares two SOUNDEX values
- Returns a value from 0 to 4
  - 0 -> little or no similarity
  - 4 -> very similar or identically matching

DIFFERENCE

```
SELECT DIFFERENCE('Illinois', 'Ilynois') AS dif_1;
```

```
| dif1 |
|------|
|  4   |
```

```
SELECT DIFFERENCE('Illinois', 'California') AS dif_2;
```

```
| dif2 |
|------|
| 1    |
```

DIFFERENCE - checking similarities

```
SELECT DISTINCT A1.airport_state, A2.airport_state
FROM airports A1
INNER JOIN airports A2
    ON DIFFERENCE(REPLACE(A1.airport_state, ' ', ''), REPLACE(A2.airport_state, ' ', '')) = 4
    AND A1.airport_state <> A2.airport_state
```

```
| airport_state | airport_state |
|---------------|---------------|
| Caalifornia   | California    |
| Caalifornia   | Californiaa   |
| California    | Caalifornia   |
| California    | Californiaa   |
| Californiaa   | Caalifornia   |
| Californiaa   | California    |
| Illinois      | Ilynois       |
| Ilynois       | Illinois      |
| Massachusetts | Michigan      |
| Tejas         | Texas         |
| Texas         | Tejas         |
```

Let's practice!

CLEANING DATA IN SQL SERVER DATABASES