SQL Data Cleaning Project

Optimizing Club Membership Database

Shivam Adbhute

Indian Institute of Technology Guwahati

GITHUB

INTRODUCTION

The club membership database contained various inconsistencies and data quality issues. These included duplicate entries, inconsistent formatting in names and addresses, and data entry errors. The need for cleaning was evident to ensure accurate analysis and reporting of member information.

This project focused on cleaning and optimizing a club membership database. The primary goals were to standardize data formats, remove duplicates, correct inconsistencies, and improve overall data quality. Through a series of SQL operations, we successfully transformed a messy dataset into a clean, structured, and more usable form.

Project Objectives:

1. **Check for duplicate entries and remove them:** Identify and eliminate duplicate records to ensure each entry is unique, reducing redundancy and improving data quality.
2. **Remove extra spaces and/or other invalid characters:** Clean up data by trimming unnecessary spaces and removing special characters to standardize input formats.
3. **Separate or combine values as needed:** Adjust data fields by splitting or merging values to align with data structure requirements, enhancing data usability.
4. **Ensure that certain values (age, dates, etc.) are within a certain range:** Validate key data points to confirm they fall within specified acceptable ranges, ensuring accuracy and consistency.
5. **Check for outliers:** Detect and manage anomalous data points that deviate significantly from the norm to prevent skewed analysis.
6. **Correct incorrect spelling or inputted data:** Review and rectify misspelled or wrongly inputted data to maintain high data integrity.
7. **Check for null or empty values**: Identify and address missing data to prevent gaps in analysis and ensure completeness in reporting.

Methodology

Let's take a look at the first few rows to examine the data in its original form-

```
SELECT
     *
FROM club_member_info
LIMIT 10;
```

Table preview-

	full_name	age	martial_status	email	phone	full_address	job_title	membership_date
2	addie lush	40	married	alush0@shutterfly.com	254-389-8708	3226 Eastlawn	Assistant Professor	7/31/2013
3	ROCK CRADICK	46	married	rcradick1@newsvine.com	910-566-2007	4 Harbort Aven	Programmer III	5/27/2018
4	???Sydel Sharvell	46	divorced	ssharvell2@amazon.co.jp	702-187-8715	4 School Place,	Budget/Accounting Analyst	########
5	Constantin de la cruz	35		co3@bloglines.com	402-688-7162	6 Monument C	Desktop Support Technician	10/20/2015
6	Gaylor Redhole	38	married	gredhole4@japanpost.jp	917-394-6001	88 Cherokee Pa	Legal Assistant	5/29/2019
7	Wanda del mar	44	single	wkunzel5@slideshare.net	937-467-6942	10864 Buhler P	Human Resources Assistant	3/24/2015
8	Jo-ann Kenealy	41	married	jkenealy6@bloomberg.com	513-726-9885	733 Hagan Park	Accountant IV	4/17/2013
9	Joete Cudiff	51	separated	jcudiff7@ycombinator.com	616-617-0965	975 Dwight Pla	Research Nurse	11/16/2014
10	mendie alexandrescu	46	single	malexandrescu8@state.gov	504-918-4753	34 Delladonna	Systems Administrator III	########

Lets create a temp table where we can manipulate and restructure the data without altering the original.

```
DROP TABLE IF EXISTS cleaned_club_member_info;
CREATE TABLE cleaned_club_member_info AS (
    SELECT
         member_id,
```

Some of the names have extra spaces and special characters. Trim excess whitespace, remove special characters and convert to lowercase.

In this particular dataset, special characters only occur in the first name that can be removed using a simple regex.

```
regexp_replace(split_part(trim(lower(full_name)), ' ', 1), '\W+',
'', 'g') AS first_name,
```

Some last names have multiple words ('de palma' or 'de la cruz').

Convert the string to an array to calculate its length and use a case statement to find entries with those particular types of surnames.

```
CASE
WHEN array_length(string_to_array(trim(lower(full_name)), ' '), 1) = 3
THEN concat(split_part(trim(lower(full_name)), ' ', 2) || ' ' ||
split_part(trim(lower(full_name)), ' ', 3))

WHEN array_length(string_to_array(trim(lower(full_name)), ' '), 1) = 4

THEN concat(split_part(trim(lower(full_name)), ' ', 2) || ' ' ||

split_part(trim(lower(full_name)), ' ', 3) || ' ' ||
split_part(trim(lower(full_name)), ' ', 4))
ELSE split_part(trim(lower(full_name)), ' ', 2)
END AS last_name,
```

During data entry, some ages have an additional digit at the end. Remove the last digit when a 3 digit age value occurs.

Check if the value is empty. If empty '' then change value to NULL.

First cast the integer to a string and test the character length.

If the condition is true, cast the integer to text, extract the first 2 digits and cast back to numeric type.

```
CASE
     WHEN length(age::text) = 0 THEN NULL
     WHEN length(age::text) = 3 THEN substr(age::text, 1,
2)::numeric
ELSE age
     END age,
```

Trim whitespace from maritial_status column and if empty, ensure its of null type

```
CASE
     WHEN trim(maritial_status) = '' THEN NULL
ELSE trim(maritial_status)
END AS maritial_status,
```

Email addresses are necessary and this dataset contains valid email addresses. Since email addresses are case insensitive, convert to lowercase and trim off any whitespace.

```
trim(lower(email)) AS member_email,
```

Trim whitespace from phone column and if empty or incomplete, ensure its of null type:

```
CASE
     WHEN trim(phone) = '' THEN NULL
     WHEN length(trim(phone)) < 12 THEN NULL
     ELSE trim(phone)
END AS phone,
```

Members must have a full address for billing purposes. However many members can live in the same household so addresses cannot be unique.

Convert to lowercase, trim off any whitespace and split the full address to individual street address, city and state.

```
split_part(trim(lower(full_address)), ',', 1) AS street_address,
split_part(trim(lower(full_address)), ',', 2) AS city,
split_part(trim(lower(full_address)), ',', 3) AS state,
```

Some job titles define a level in roman numerals (I, II, III, IV). Convert levels to numbers and add descriptors (ex. Level 3).

Trim whitespace from job title, rename to occupation and if empty convert to null type.

```
CASE
     WHEN trim(lower(job_title)) = '' THEN NULL
ELSE
CASE
     WHEN array_length(string_to_array(trim(job_title), ' '), 1) > 1
AND lower(split_part(job_title, ' ',
array_length(string_to_array(trim(job_title), ' '), 1))) = 'i'
           THEN replace(lower(job_title), ' i', ', level 1')
     WHEN array_length(string_to_array(trim(job_title), ' '), 1) > 1
AND lower(split_part(job_title, ' ',
array_length(string_to_array(trim(job_title), ' '), 1))) = 'ii'
           THEN replace(lower(job_title), ' ii', ', level 2')
     WHEN array_length(string_to_array(trim(job_title), ' '), 1) > 1
AND lower(split_part(job_title, ' ',
array_length(string_to_array(trim(job_title), ' '), 1))) = 'iii'
           THEN replace(lower(job_title), ' iii', ', level 3')
     WHEN array_length(string_to_array(trim(job_title), ' '), 1) > 1
AND lower(split_part(job_title, ' ',
array_length(string_to_array(trim(job_title), ' '), 1))) = 'iv'
           THEN replace(lower(job_title), ' iv', ', level 4')
ELSE trim(lower(job_title))
END
END AS occupation,
```

A few members show membership_date year in the 1900's. Change the year into the 2000's.

```
CASE
WHEN EXTRACT('year' FROM membership_date) < 2000
THEN concat(replace(EXTRACT('year' FROM membership_date)::text, '19',
'20') || '-' || EXTRACT('month' FROM membership_date) || '-' ||
EXTRACT('day' FROM membership_date))::date
ELSE membership_date
END AS membership_date
FROM club_member_info
);
```

Let's take a look at our cleaned table data.

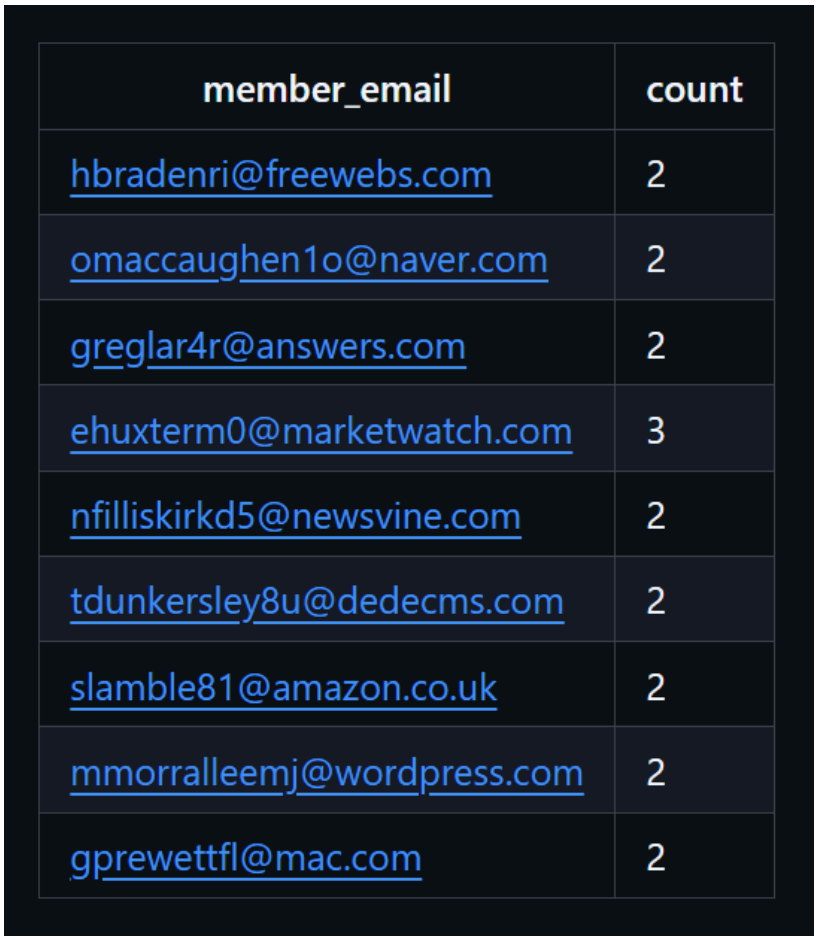
1	addie	lush	40	married	alush0@shutterfly.com	254-389-8708	3226 eastlawn pass	temple	texas	assistant professor	31-07-2013
2	rock	cradick	46	married	rcradick1@newsvine.com	910-566-2007	4 harbort avenue	fayetteville	north carolina	programmer, level 3	27-05-2018
3	sydel	sharvell	46	divorced	ssharvell2@amazon.co.jp	702-187-8715	4 school place	las vegas	nevada	budget/accounting analyst, level 1	06-10-2017
4	constantin	de la cruz	35		co3@bloglines.com	402-688-7162	6 monument crossing	omaha	nebraska	desktop support technician	20-10-2015
5	gaylor	redhole	38	married	gredhole4@japanpost.jp	917-394-6001	88 cherokee pass	new york city	new york	legal assistant	29-05-2019
6	wanda	del mar	44	single	wkunzel5@slideshare.net	937-467-6942	10864 buhler plaza	hamilton	ohio	human resources assistant, level 4	24-03-2015
7	joann	kenealy	41	married	jkenealy6@bloomberg.com	513-726-9885	733 hagan parkway	cincinnati	ohio	accountant, level 4	17-04-2013
8	joete	cudiff	51	separated	jcudiff7@ycombinator.com	616-617-0965	975 dwight plaza	grand rapids	michigan	research nurse	16-11-2014
9	mendie	alexandrescu	46	single	malexandrescu8@state.gov	504-918-4753	34 delladonna terrace	new orleans	louisiana	systems administrator, level 3	12-03-2021

Now that the data is cleaned, lets look for any duplicate entries. What is the record count?

All members must have a unique email address to join. Lets try to find duplicate entries.

```
SELECT
     member_email,
     count(member_email)
FROM
     cleaned_club_member_info
GROUP BY
     member_email
HAVING
     count(member_email) > 1
```

Results: 10 duplicate entries



member_email	count
hbradenri@freewebs.com	2
omaccaughen1o@naver.com	2
greglar4r@answers.com	2
ehuxterm0@marketwatch.com	3
nfilliskirkd5@newsvine.com	2
tdunkersley8u@dedecms.com	2
slamble81@amazon.co.uk	2
mmorralleemj@wordpress.com	2
gprewettfl@mac.com	2

Let's delete duplicate entries-

```
DELETE FROM
     cleaned_club_member_info AS c1
USING
     cleaned_club_member_info AS c2
WHERE
     c1.member_id < c2.member_id
AND
     c1.member_email = c2.member_email;
```

Results and Impact

1. Reduced record count from 2010 to 2000 by removing duplicates
2. Standardized name formats for easier searching and sorting
3. Improved data structure by splitting full addresses into components
4. Enhanced job title consistency by standardizing level indicators
5. Corrected historical date entries to ensure accuracy

Challenges Encountered and Solutions:

1. Multi-word last names: Solved using complex case statements
2. Inconsistent job title formats: Addressed by standardizing level indicators
3. Date inconsistencies: Resolved by identifying and updating outdated entries

Conclusion:

This data cleaning project successfully transformed a messy club membership database into a clean, consistent, and more valuable dataset. The cleaned data will support more accurate analysis and reporting, ultimately leading to better decision-making for the club.