



SQL-based data quality management in practice

Raiko Limmart, CEO & co-founder

User requirements

Security

On-premises deployment, role & privilege management, strict security standards

Ease of use

Self explanatory, usable by business and engineering, minimal manual effort

Transparency

Simplify audits, compliance reviews, and troubleshooting

Flexibility

Ensure data quality in complex environments

Alternative solutions

```
checks for main.students:
- invalid_count(degree) = 0:
  valid values: ['Bachelor', 'Master', 'PhD']
  name: "VALIDITY invalid degree"
- invalid_count(age) = 0:
  valid min: 1
  valid max: 100
  name: "VALIDITY invalid age"
```

```
- name: country_name
  tests:
    - accepted_values:
      values: ['United States', 'Canada', 'Mexico']
      config:
        severity: warn
```

```
checks_custom = yaml.safe_load("""
- check:
  arguments:
    col_name: customer_id
  function: is_not_null_and_not_empty
  criticality: error
  name: customer_id_is_null
- check:
  arguments:
    col_name: customer_email
    regex: '^[A-Za-z0-9._%+-]+@example\.com$'
  function: regex_match
  criticality: error
  name: customer_emaild_is_not_valid""")
```

SQL-based validations



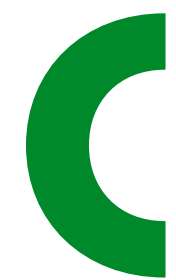
Flexibility and precision

Include any level of filtering, joins, aggregations, and conditional logic



Transparency

Clear, readable query that shows exactly what data is being assessed and how the results are derived



Maintainability

Validations can be easily updated for schema changes or new sources



Proficiency

Create, review, and enhance validations without learning new tools

Validation format

Generic:

```
SELECT COUNT(*) FROM {object} WHERE {conditions}
```

More complex:

```
SELECT COUNT(*) FROM (  
    SELECT {projection} FROM {object}  
    {subqueries, JOINS}  
    WHERE {conditions}  
) a
```

Expected result:

=0 or >0

Sample validations 1/3

1 Availability - Data is accessible when needed.

Description Check if data is available in the 'customer_dimension' table

```
1 SELECT COUNT(*)
2 FROM public.customer_dimension
```

2 Timeliness - Data is up-to-date.

Description Check if customers for the last 24h have been loaded.

```
1 SELECT COUNT(*)
2 FROM public.customers
3 WHERE registration_date > NOW() - INTERVAL '1 day'
```

3 Uniqueness - No duplicate records representing the same entity.

Description Verify that all email addresses are unique.

```
1 SELECT COUNT(*)
2 FROM (
3     SELECT email FROM public.customers GROUP BY email HAVING COUNT(*) > 1
4 ) AS duplicate_emails
```

Sample validations 2/3

4 **Validity** - Data values fall within the defined formats and business rules.

Description Verify that all customer_type values are either 'Individual' or 'Company'.

```
1 SELECT COUNT(*)
2 FROM public.customer_dimension
3 WHERE customer_type NOT IN ('Individual', 'Company')
```

5 **Completeness** - All required data is present and populated.

Description Verify that the country_code column is not empty for any customer.

```
1 SELECT COUNT(*)
2 FROM public.customers
3 WHERE country_code IS NULL OR country_code = ''
```

6 **Accuracy** - Data correctly represents the real-world object.

Description Verify that all sold products in the online_sales_fact table exist in the product_dimension table.

```
1 SELECT COUNT(*)
2 FROM online_sales.online_sales_fact osf
3 LEFT JOIN public.product_dimension pd ON osf.product_key = pd.product_key
4 WHERE pd.product_key IS NULL
```

Sample validations 3/3

7

Description

Check if all customers with a purchase from the 'medical' category are of type 'Company'.

```
1 SELECT COUNT(*)
2 FROM online_sales.online_sales_fact osf
3   JOIN public.customer_dimension cd ON osf.customer_key = cd.customer_key
4   JOIN public.product_dimension pd ON osf.product_key = pd.product_key
5 WHERE pd.category_description = 'medical'
6   AND cd.customer_type <> 'Company'
```

8

Description

Check if there are any online sales records with a sale date that does not match the date key in the date_dimension table.

```
1 WITH sales_dates AS (
2   SELECT sale_date_key,
3     online_sales_saledate
4   FROM online_sales.online_sales_fact
5 ),
6 valid_dates AS (
7   SELECT date_key,
8     date
9   FROM public.date_dimension
10 )
11 SELECT COUNT(*)
12 FROM sales_dates sd
13   LEFT JOIN valid_dates vd ON sd.sale_date_key = vd.date_key
14 WHERE sd.online_sales_saledate <> vd.date
```

Data reconciliation

Connection A: DO MySQL ▾

```
1 SELECT
2 id, first_name, last_name, email, gender, address_city, address_country,
  company_name
3 FROM mysql.CUSTOMERS_MYSQL WHERE registration_date > NOW() - INTERVAL '1 day'
```

Connection B: DO postgres ▾

```
1 SELECT
2 id, first_name, last_name, email, gender, address_city, address_country,
  company_name
3 FROM public.customers_postgres
```

id	first_name	last_name	email	gender	address_cit
11	Buddy	Southern	bsoutherna@bloglovin.com	Female	Wielgie
15	Halsey	Belfit	hbelfite@tripod.com	Female	Pingdong
163	Judas	Sparham	jsparham4i@usnews.com	Female	Godong
246	Teresita	Petran	tpetran6t@unc.edu	Male	Seredyna
273	Manolo	Stoade	mstoade7k@wsj.co.uk	Male	Lüderitz
275	Blisse	Pursey	bpursey7m@jugem.jp	Male	Hongy
309	Vernon	Grigoriscu	vgrigoriscu8k@altervista.org	Male	Gövie
354	Fergus	Kettle	fkettle9t@samsung.com	Female	Dainan
35	Sumner	Bennedsen	sbennedseny@google.nl	null	null
402	Barde	Balfye	bbalfyeb5@globo.com	Female	Taupse
5	Ginnie	Faulder	gfaulder4@blog.com	Female	null
61	Wilden	Klemke	wklemke1o@usa.com	Male	Diré

id	first_name	last_name	email	gender	address_cit
11	Buddy	Southern	bsoutherna@bloglovin.com	Female	Wielgie
15	Halsey	Belfit	hbelfite@tripod.com	Male	Pingdong
163	Judas	Sparham	jsparham4i@usnews.com	Male	Godong
246	Teresita	Petran	tpetran6t@unc.edu	Male	Seredyna-Bu
273	Manolo	Stoade	mstoade7k@wsj.com	Male	Lüderitz
275	Blisse	Pursey	bpursey7m@jugem.jp	Male	Hongyi
309	Vernon	Grigoriscu	vgrigoriscu8k@altervista.org	Male	Gävie
354	Fergus	Kettle	fkettle9t@samsung.com	Female	Dainan
35	Sumner	Bennedsen	sbennedseny@google.nl	Female	null
402	Barde	Balfye	bbalfyeb5@globo.com	Female	Tuapse
58	Temple	Faraday	tfaraday1l@myspace.com	Male	Kaiaf
59	Shani	Ferrieres	sferrieres1m@tinyurl.com	Male	Baima
5	Ginnie	Faulder	gfaulder4@blog.com	Female	null
61	Wilden	Klemke	wklemke1o@usa.gov	Male	Diré
76	Heriberto	Goane	hgoane23@wsj.com	Male	null
77	Boote	Wymer	null	Male	Telsen

Simplifying query creation

Dynamic rules

Create validation queries from pre-defined SQL-based templates.

Statistical suggestions

Generate queries based on statistical analysis to find patterns, prefixes, and valid values.

LLM-based suggestions

Use AI to help generate business rules based on metadata and statistics.

Natural language

Use AI to create a validation based on a business rule and metadata.

Dynamic rules

Test case description:

<schema>.<object> - Duplicates

Rule description:

Check if object has any duplicate rows

SQL:

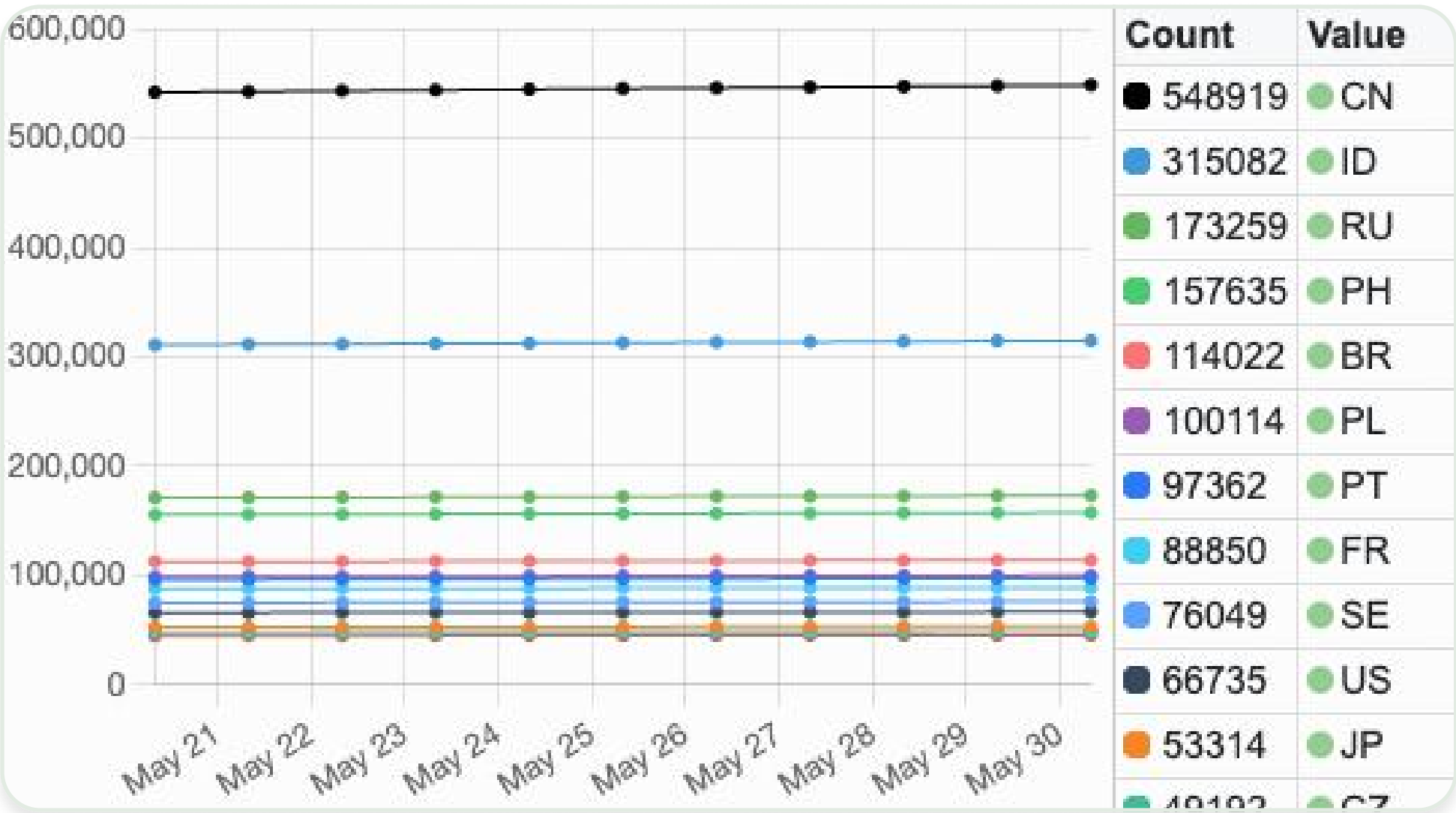
```
1 SELECT COUNT(*) FROM (SELECT <columns> FROM <schema>.<object>
  GROUP BY <columns> HAVING COUNT(*)>1) a
```

Expected result:

=0

Statistical suggestions

1



Description

Expect 'country_code' to be fixed length

```
1 SELECT COUNT(*)
2 FROM public.customers
3 WHERE LENGTH(CAST(country_code AS TEXT)) <> 2
```

2

Count	Value
17	436 Main St
16	2 Lake St
15	290 School St
14	54 Hereford Rd
14	467 Mission St
14	35 Church St
14	303 Mission St
14	174 Lake St
14	104 School St
13	175 Maple St
13	16 Hereford Rd

Description

Expect 'customer_address' to match a pattern [custom]

```
1 SELECT COUNT(*)
2 FROM public.customer_dimension
3 WHERE NOT REGEXP_LIKE(
4     CAST(customer_address AS VARCHAR),
5     '[0-9]+ [A-Za-z]+ [A-Za-z]{2}'
6 )
```

LLM-based suggestions

Quality ↕↗	Attribute ↕↗	Type ↕↗	Description ↕↗	Empty values ↕↗	Existing values ↕↗	Unique values ↕↗	Terms	Database comment ↕↗
90 %	customer_key 🔑	int		0 0.00%	50000 100.00%	50000 100.00%	Term Customer ID	
100 %	customer_type 🔑	varchar(16)	customer_dimension	0 0.00%	50000 100.00%	2 0.00%		Customer type can be Individual or..
85.71 %	customer_name 🔑	varchar(256)	First name + last name	0 0.00%	50000 100.00%	36322 72.64%	Term Name Term PII Term First name Term Last n	Customer name from CRM
100 %	customer_gender 🔑	varchar(8)		1710 3.42%	48290 96.58%	2 0.00%	Term Gender	Male or Female
100 %	title	varchar(8)		1710 3.42%	48290 96.58%	5 0.01%		
100 %	household_id	int		1710 3.42%	48290 96.58%	24120 48.24%		
100 %	customer_address 🔑	varchar(256)		2 0.00%	49998 100.00%	9940 19.88%	Term Address Term PII	Address in format "number street"

Description ↕↗	SQL ↕↗	Expected result ↕↗
Check if there are any customers with a negative customer_age.	SELECT COUNT(*) FROM public.customer_dimension WHERE customer_age < 0	=0
Verify that all customer_gender values are either 'Male' or 'Female'.	SELECT COUNT(*) FROM public.customer_dimension WHERE customer_gender NOT IN ('Male', 'Female')	=0
Validate that customer_state values are exactly 2 characters long.	SELECT COUNT(*) FROM public.customer_dimension WHERE LENGTH(customer_state) <> 2	=0
Verify that all customer_type values are either 'Individual' or 'Company'.	SELECT COUNT(*) FROM public.customer_dimension WHERE customer_type NOT IN ('Individual', 'Company')	=0
Verify that customer_since dates are not in the future.	SELECT COUNT(*) FROM public.customer_dimension WHERE customer_since > CURRENT_DATE	=0
Check if there are any customers with a null customer_name.	SELECT COUNT(*) FROM public.customer_dimension WHERE customer_name IS NULL	=0
Verify that all customer_address values follow the format 'number street'.	SELECT COUNT(*) FROM public.customer_dimension WHERE customer_address !~ '^[0-9]+ .+'	=0

Validations from natural language

BR input 1

Schema public	Object customers	Columns email	37
Additional information (optional) validate email format and .com domain			

BR output 1

SQL	<pre>1 SELECT COUNT(*) 2 FROM public.customers 3 WHERE email !~* '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\..com\$'</pre>
Expected result	=0
Description	Check if all email addresses in the 'customers' table have a valid format and end with '.com'.

BR input 2

Schema online_sales	Object online_sales_fact	Columns All selected	
Schema public	Object customer_dimension	Columns All selected	
Schema public	Object product_dimension	Columns All selected	
Additional information (optional) all customers with a purchase from "medical" category must be of type "Company"			

BR output 2

SQL	<pre>1 SELECT COUNT(*) 2 FROM online_sales.online_sales_fact osf 3 JOIN public.customer_dimension cd ON osf.customer_key = cd.customer_key 4 JOIN public.product_dimension pd ON osf.product_key = pd.product_key 5 WHERE pd.category_description = 'medical' 6 AND cd.customer_type <> 'Company'</pre>
Expected result	=0
Description	Check if all customers with a purchase from the 'medical' category are of type 'Company'.

Questions & Answers

SQL-based data quality management in practice

Raiko Limmart
raiko@selectzero.io
www.selectzero.io