

## Analyzing NPS Data in Snowflake

### **Analyzing NPS Data in Snowflake**

Shweta Ajay Shinde

Masters in Data Analytics, San Jose State University

Data 226: Data warehousing

Instructor: Keeyong Han

18<sup>th</sup> Sept 2024

## Analyzing NPS Data in Snowflake

1. (+1) Explain what NPS (Net Promoter Score) is and how it is calculated.

**Net Promoter Score (NPS)** is a way to measure how happy customers are and how likely they are to tell others about a company's product or service. The score ranges from -100 to 100 and is derived from survey responses to the question: "On a scale from 0 to 10, how likely are you to recommend our company/product/service to a friend or colleague?"

Calculation:

- Categorize Respondents:
  - **Promoters (Score 9-10):** Loyal customers who are likely to spread positive word-of-mouth.
  - **Passives (Score 7-8):** Satisfied but unenthusiastic customers who are vulnerable to competitors.
  - **Detractors (Score 0-6):** Unhappy customers who may spread negative feedback.
- Calculate the Percentage:
  - **% Promoters** = (Number of Promoters / Total Responses) \* 100
  - **% Detractors** = (Number of Detractors / Total Responses) \* 100
- Compute NPS:
  - $NPS = \% \text{ Promoters} - \% \text{ Detractors}$     OR
  - $NPS = (\text{Number of Promoters} - \text{Number of Detractors}) * 100 / \text{Total Responses}$

2. (+1) Create raw\_data and analytics schemas under a database in your Snowflake.

**CREATE SCHEMA RAW\_DATA;**

**CRATE SCHEMA ANALYTICS;**

The screenshot displays the Snowflake web interface. On the left, the 'Databases' tab is active, showing a tree view of objects. Under the 'NPS\_RAW\_DATA' database, the 'ANALYTICS' schema is highlighted. The main panel on the right shows the SQL editor with the following commands:

```
1 CREATE SCHEMA RAW_DATA;
2 CREATE SCHEMA ANALYTICS;
3
```

Below the editor, the 'Results' tab is selected, showing a table with the following data:

	status
1	Schema ANALYTICS successfully created.

## Analyzing NPS Data in Snowflake

3. (+1) Create a table named nps with primary key attribute under raw\_data schema.

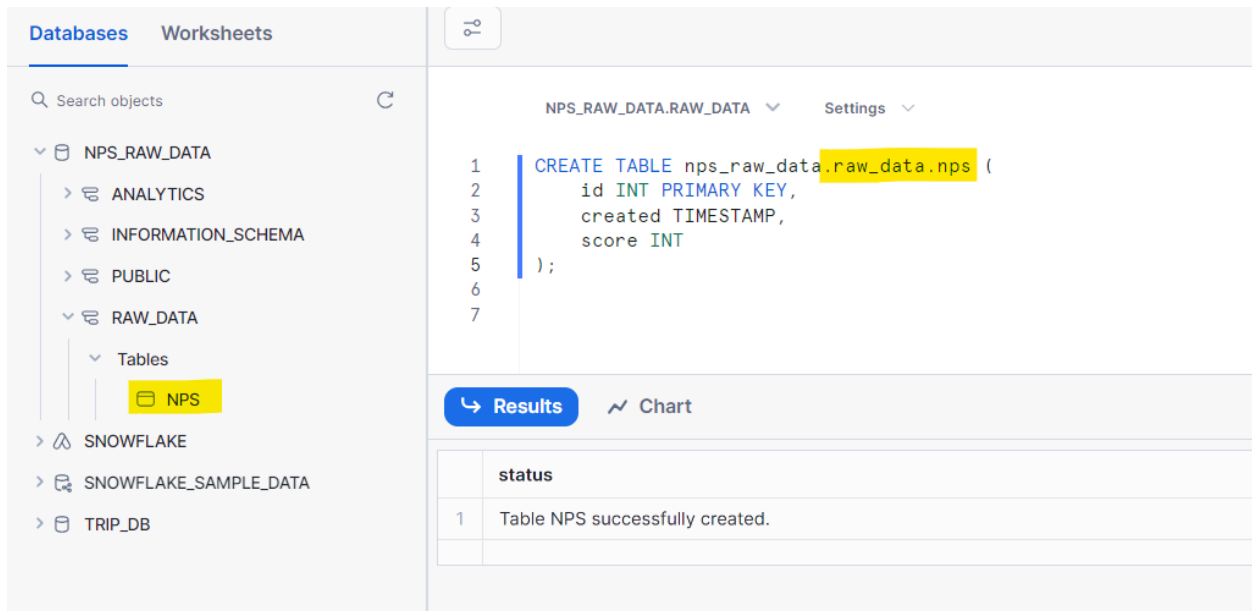
```
CREATE TABLE nps_raw_data.raw_data.nps (  

  id INT PRIMARY KEY,  

  created TIMESTAMP,  

  score INT  

);
```

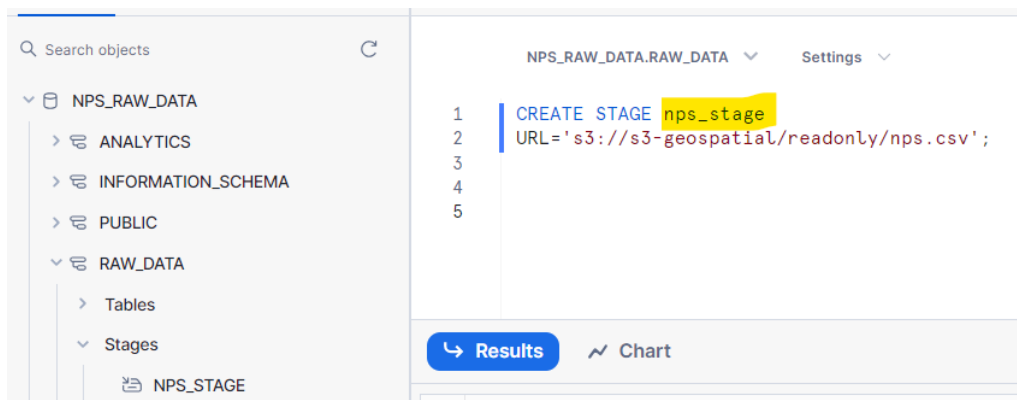


4. (+2) Copy a file (nps.csv) into the nps table using a stage :-s3://s3-geospatial/readonly/nps.csv

First, create a stage:

```
CREATE STAGE nps_stage  

URL='s3://s3-geospatial/readonly/nps.csv'
```



## Analyzing NPS Data in Snowflake

Then, copy the CSV data into the nps table:

```
COPY INTO raw_data.nps

FROM @nps_stage

FILE_FORMAT = (TYPE = 'CSV' FIELD_OPTIONALLY_ENCLOSED_BY = ''
SKIP_HEADER = 1);
```

The screenshot shows the Snowflake query editor with the following SQL code:

```
1 COPY INTO raw_data.nps
2 FROM @nps_stage
3 FILE_FORMAT = (TYPE = 'CSV' FIELD_OPTIONALLY_ENCLOSED_BY = ''
4 SKIP_HEADER = 1);
5
```

The results pane displays a table with the following columns: file, status, rows\_parsed, rows\_loaded, error\_limit, errors\_seen, first\_error, first\_error\_line, first\_error\_character. The data shows that 157,757 rows were successfully loaded from the CSV file s3://s3-geospatial/readonly/nps.csv.

file	status	rows_parsed	rows_loaded	error_limit	errors_seen	first_error	first_error_line	first_error_character
s3://s3-geospatial/readonly/nps.csv	LOADED	157757	157757	1	0	null	null	null

Query Details: Query duration 1.7s, Rows 1, Query ID 01b724fd-0002-e7a1-0...

**Select \* from raw\_data.nps**

The screenshot shows the Snowflake query editor with the following SQL code:

```
1 select * from raw_data.nps;
2
3
4
```

The results pane displays a table with the following columns: ID, CREATED, SCORE. The data shows 14 rows of NPS scores, ranging from 1 to 10.

ID	CREATED	SCORE
1	2019-01-01 22:04:21.000	1
2	2019-01-02 03:47:26.000	9
3	2019-01-03 19:31:41.000	10
4	2019-01-04 05:02:11.000	10
5	2019-01-04 18:02:53.000	1
6	2019-01-04 23:06:18.000	2
7	2019-01-05 13:45:47.000	0
8	2019-01-05 09:26:13.000	2
9	2019-01-05 22:36:38.000	6
10	2019-01-06 13:22:47.000	10
11	2019-01-06 14:40:11.000	1
12	2019-01-06 06:09:47.000	4
13	2019-01-06 23:37:19.000	3
14	2019-01-06 02:33:32.000	2

Query Details: Query duration 769ms, Rows 157.8K, Query ID 01b724fe-0002-e7a9-0...

Visualizations: A bar chart for ID (1 to 157,757) and a line chart for CREATED (2019-01-01 to 2019-12-30) are shown. An 'Ask Copilot' button is also present.

## Analyzing NPS Data in Snowflake

5.(+2) What types of data quality validations can we perform against the nps table? Please name at least 3 methods.

Here are three types of data quality validations you can perform against an NPS table using SQL:

### 1. Check for Null Values

Ensure that there are no missing values in key columns, such as the score and ID identifier. SQL to check if there are any NULL values.

```
SELECT COUNT(*) AS NullCount  
FROM nps  
WHERE score IS NULL OR id IS NULL;
```

NPS\_RAW\_DATA.RAW\_DATA Settings

```
1 SELECT COUNT(*) AS NullCount
2 FROM nps
3 WHERE score IS NULL OR id IS NULL;
4
5
```

Results Chart

	NULLCOUNT	
1	0	

### 2. Validate Score Range

Check that the NPS scores fall within the valid range of 0 to 10. SQL to check the valid range.

```
SELECT COUNT(*) AS InvalidScoreCount  
FROM nps  
WHERE score < 0 OR score > 10;
```

NPS\_RAW\_DATA.RAW\_DATA Settings

```
1 SELECT COUNT(*) AS InvalidScoreCount
2 FROM nps
3 WHERE score < 0 OR score > 10;
4
5
6
```

Results Chart

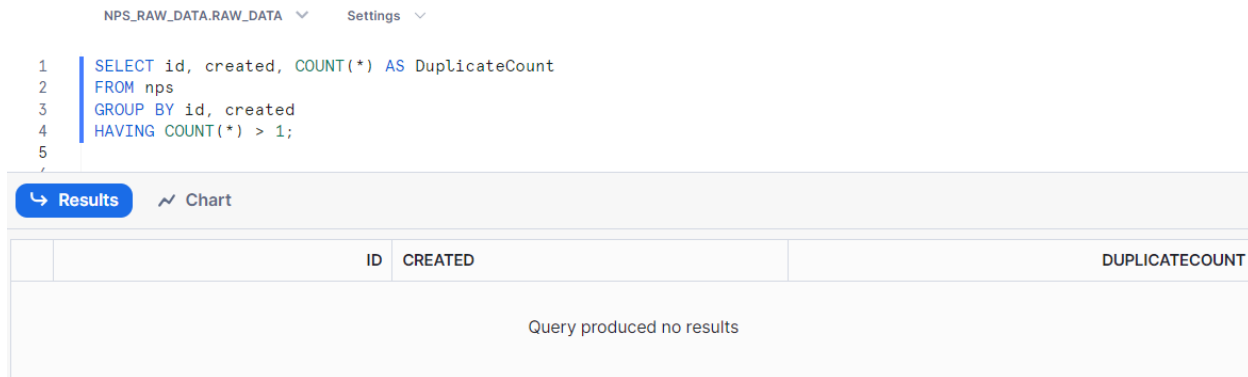
	INVALIDSCORECOUNT	
1	0	

## Analyzing NPS Data in Snowflake

### 3. Check for Duplicate Entries

Ensure there are no duplicate records for the same ID on the same survey date.  
SQL to check if there are duplicates.

```
SELECT id, created, COUNT(*) AS DuplicateCount
FROM nps
GROUP BY id, created
HAVING COUNT(*) > 1;
```



ID	CREATED	DUPLICATECOUNT
Query produced no results		

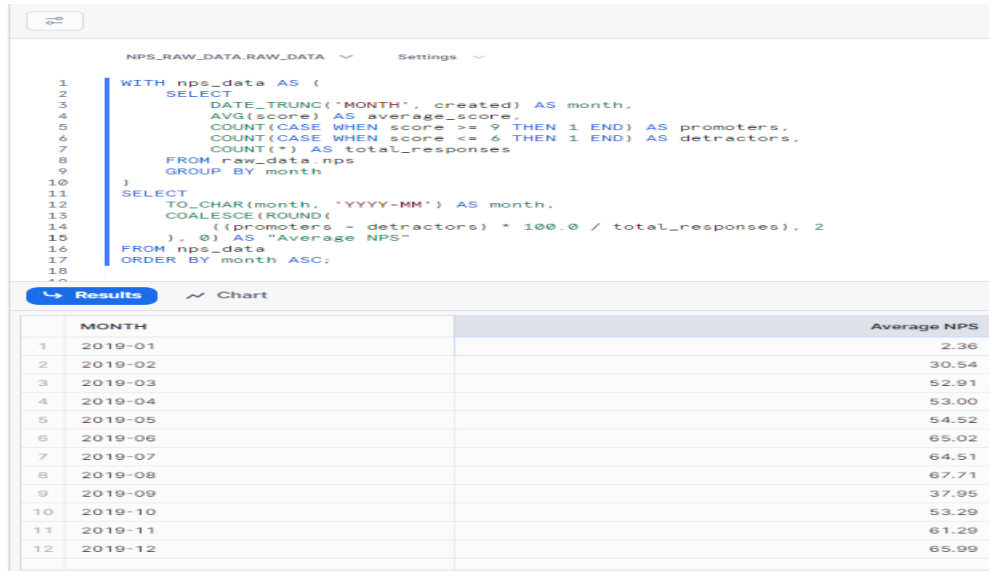
Overall, these tests help maintain accurate and reliable data for better decision-making

6.(+4) Develop a SELECT SQL query to calculate monthly NPS, with results sorted by month in ascending order.

→Here I am using WITH to Improved Readability, Reuse of Logic and Organization of query

```
WITH nps_data AS (
  SELECT
    DATE_TRUNC('MONTH', created) AS month,
    AVG(score) AS average_score,
    COUNT(CASE WHEN score >= 9 THEN 1 END) AS promoters,
    COUNT(CASE WHEN score <= 6 THEN 1 END) AS detractors,
    COUNT(*) AS total_responses
  FROM raw_data.nps
  GROUP BY month
)
SELECT
  TO_CHAR(month, 'YYYY-MM') AS month,
  COALESCE(ROUND(((promoters - detractors) * 100.0 / total_responses), 2), 0) AS
  "AverageNPS"
FROM nps_data
ORDER BY month ASC;
```

## Analyzing NPS Data in Snowflake



The screenshot shows the Snowflake SQL Editor interface. The top bar indicates the database is 'NPS\_RAW\_DATA' and the schema is 'RAW\_DATA'. The query is as follows:

```

1  WITH nps_data AS (
2      SELECT
3          DATE_TRUNC('MONTH', created) AS month,
4          AVG(score) AS average_score,
5          COUNT(CASE WHEN score >= 9 THEN 1 END) AS promoters,
6          COUNT(CASE WHEN score <= 6 THEN 1 END) AS detractors,
7          COUNT(*) AS total_responses
8      FROM raw_data.nps
9      GROUP BY month
10 )
11 SELECT
12     TO_CHAR(month, 'YYYY-MM') AS month,
13     COALESCE(ROUND(
14         ((promoters - detractors) * 100.0 / total_responses), 2
15     ), 0) AS "Average NPS"
16 FROM nps_data
17 ORDER BY month ASC;

```

Below the query, the 'Results' tab is active, displaying a table with 12 rows of data for the year 2019:

	MONTH	Average NPS
1	2019-01	2.36
2	2019-02	30.54
3	2019-03	52.91
4	2019-04	53.00
5	2019-05	54.52
6	2019-06	65.02
7	2019-07	64.51
8	2019-08	67.71
9	2019-09	37.95
10	2019-10	53.29
11	2019-11	61.29
12	2019-12	65.99

Or we can use the below SQL without WITH:-

```

SELECT
    TO_CHAR(DATE_TRUNC('MONTH', created), 'YYYY-MM') AS month,
    COALESCE(ROUND(
        ((COUNT(CASE WHEN score >= 9 THEN 1 END) -
        COUNT(CASE WHEN score <= 6 THEN 1 END)) * 100.0 /
        NULLIF(COUNT(*), 0)), 2), 0) AS "Average NPS"
FROM raw_data.nps
GROUP BY month
ORDER BY month ASC;

```

MONTH	Average NPS
2019-01	2.36
2019-02	30.54
2019-03	52.91
2019-04	53.00
2019-05	54.52
2019-06	65.02
2019-07	64.51
2019-08	67.71
2019-09	37.95
2019-10	53.29
2019-11	61.29
2019-12	65.99

## Analyzing NPS Data in Snowflake

7. (+2) Use CTAS (CREATE TABLE AS SELECT) to generate the nps\_summary table in the analytics schema, populating it with results from step 6.

```
CREATE TABLE analytics.nps_summary AS
SELECT
TO_CHAR(DATE_TRUNC('MONTH', created), 'YYYY-MM') AS month,
COALESCE(ROUND( ((COUNT(CASE WHEN score >= 9 THEN 1 END) - COUNT(CASE
WHEN score <= 6 THEN 1 END)) * 100.0 / NULLIF(COUNT(*), 0)), 2), 0) AS "Average NPS"
FROM raw_data.nps
GROUP BY month
ORDER BY month ASC;
```

The screenshot shows the Snowflake SQL Editor interface. On the left, the 'Databases' tab is active, showing a tree view of the database structure. The 'RAW\_DATA' database is selected, and the 'ANALYTICS' schema is expanded, showing the 'NPS\_SUMMARY' table. The main editor area displays the SQL query from the previous block. Below the query, the 'Results' tab is active, showing a single row with the status: 'Table NPS\_SUMMARY successfully created.'

**Select \* from analytics.nps\_summary;**

The screenshot shows the Snowflake SQL Editor interface. The main editor area displays the query: `select * from analytics.nps_summary;`. Below the query, the 'Results' tab is active, showing a table with 12 rows of data. The table has two columns: 'MONTH' and 'Average NPS'.

	MONTH	Average NPS
1	2019-01	2.36
2	2019-02	30.54
3	2019-03	52.91
4	2019-04	53.00
5	2019-05	54.52
6	2019-06	65.02
7	2019-07	64.51
8	2019-08	67.71
9	2019-09	37.95
10	2019-10	53.29
11	2019-11	61.29
12	2019-12	65.99