Dataset: Cohort_Raw

Performing the Exploratory Data Analysis

Step 1: Understand the Dataset

- View the first few rows to get an overview of the data.
- Check column names, data types, and the total number of records.

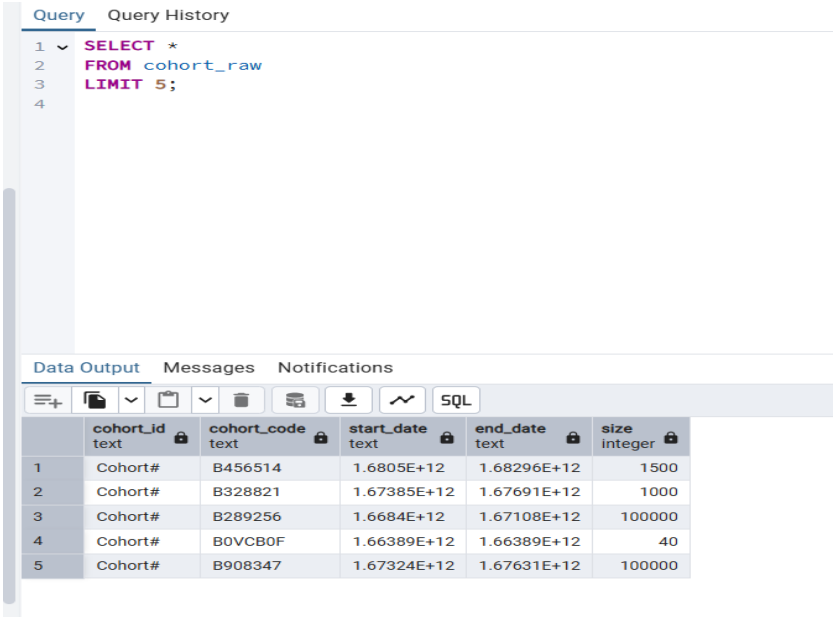
Research how to inspect a table's structure and data in PostgreSQL.

PostgreSQL Queries to Use:

1. View First Few Rows:

```
SELECT *
FROM cohort_raw
LIMIT 5;
```

🔍 *This gives you a quick peek at the dataset — ideal for spotting structure or obvious issues.*



	cohort_id text	cohort_code text	start_date text	end_date text	size integer
1	Cohort#	B456514	1.6805E+12	1.68296E+12	1500
2	Cohort#	B328821	1.67385E+12	1.67691E+12	1000
3	Cohort#	B289256	1.6684E+12	1.67108E+12	100000
4	Cohort#	B0VCB0F	1.66389E+12	1.66389E+12	40
5	Cohort#	B908347	1.67324E+12	1.67631E+12	100000

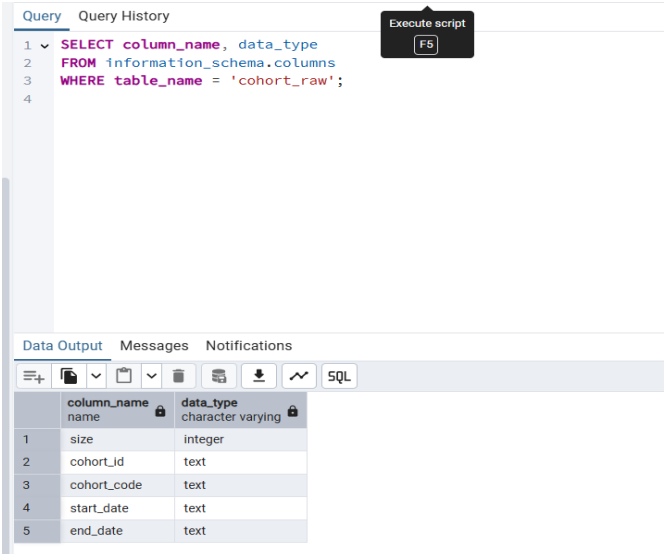
---

2. Check Column Names & Data Types:

```
SELECT column_name, data_type
```

```
FROM information_schema.columns
WHERE table_name = 'cohort_raw';
```

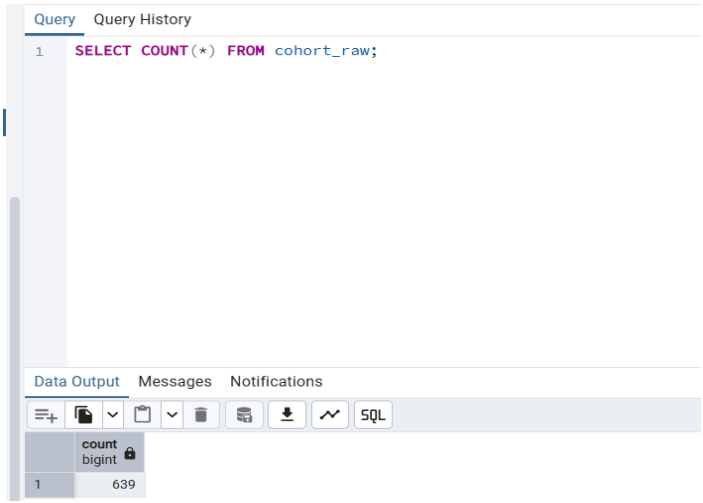
 This tells you what each column is, and whether it's stored as text, numeric, timestamp, etc.



---

3. Count Total Rows:

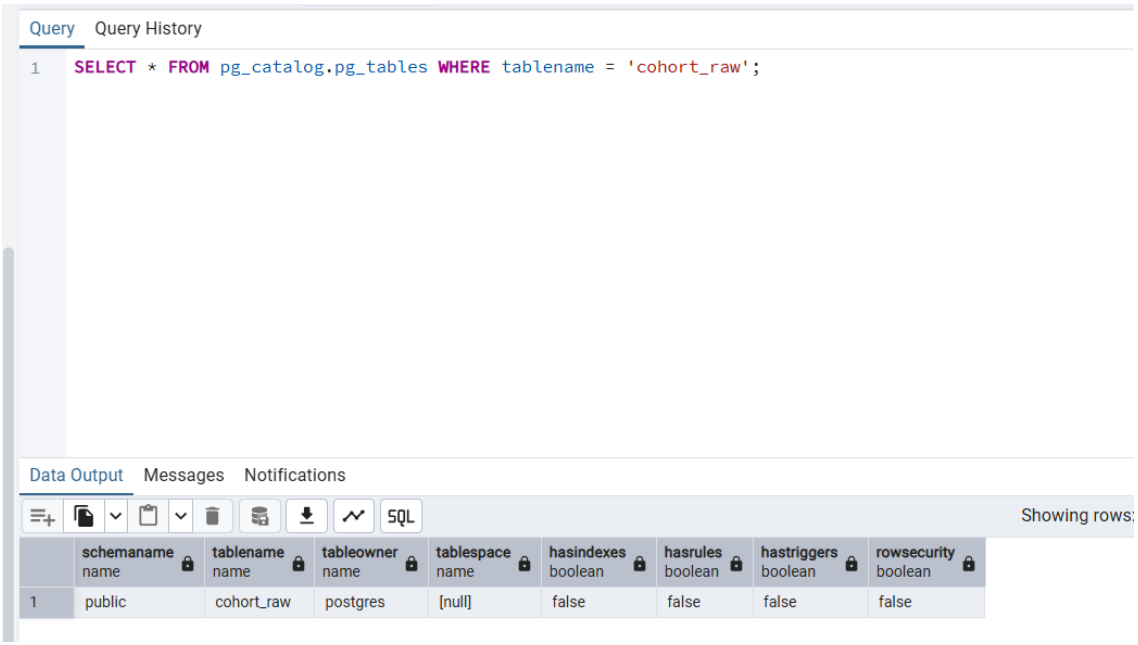
```
SELECT COUNT(*) FROM cohort_raw;
```



Optional: Full Table Structure (DDL)

If you want to inspect the table's SQL definition:

```
SELECT * FROM pg_catalog.pg_tables WHERE tablename = 'cohort_raw';
```



Step 2: Summarize the Data

- Generate basic statistics like mean, median, min, max, and standard deviation to understand the distribution.

Explore how to compute summary statistics in SQL.

Since the `CohortRaw(in).csv` dataset contains one numeric field of interest — `size` (number of learners in each cohort) — we'll compute:

- Minimum
- Maximum
- Mean (average)
- Median
- Standard deviation
- Count (just for completeness)

SQL Queries for Summary Statistics (PostgreSQL):

1. Basic Descriptive Stats:

```
SELECT
  MIN(size) AS min_size,
  MAX(size) AS max_size,
  AVG(size) AS mean_size,
  STDDEV(size) AS std_dev,
  COUNT(size) AS total_records
FROM cohort_raw;
```

---

2. Median (PostgreSQL does not have a built-in `MEDIAN()` function, but you can get it like this):

```
SELECT PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY size) AS median_size
FROM cohort_raw;
```

What These Queries Will Show:

Statistic	Description
`MIN(size)`	Smallest cohort size
`MAX(size)`	Largest cohort size
`AVG(size)`	Average cohort size
`STDDEV(size)`	Standard deviation — shows how spread out the sizes are
`COUNT(size)`	Total number of cohort records
`MEDIAN(size)`	Middle value of all sizes (better indicator than average if there are outliers)

Step 3: Identify Missing and Duplicate Values

- Detect missing data to determine if any cleaning is needed.
- Identify duplicate records that may need removal.

Look up methods to check for missing values and duplicates in PostgreSQL.

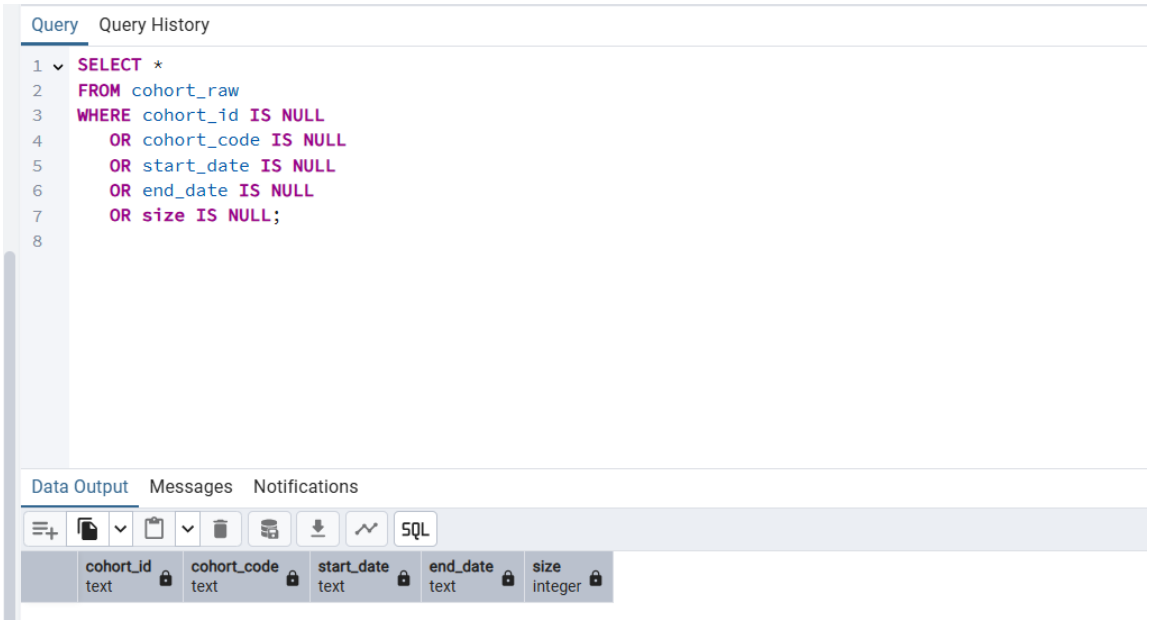
Part 1: Detect Missing (NULL) Values

1. Missing Count for Each Column

```
SELECT
  COUNT(*) AS total_rows,
  COUNT(cohort_id) AS cohort_id_not_null,
  COUNT(cohort_code) AS cohort_code_not_null,
  COUNT(start_date) AS start_date_not_null,
  COUNT(end_date) AS end_date_not_null,
  COUNT(size) AS size_not_null
FROM cohort_raw;
```

2. Show Rows With Any NULL

```
SELECT *
FROM cohort_raw
WHERE cohort_id IS NULL
   OR cohort_code IS NULL
   OR start_date IS NULL
   OR end_date IS NULL
   OR size IS NULL;
```

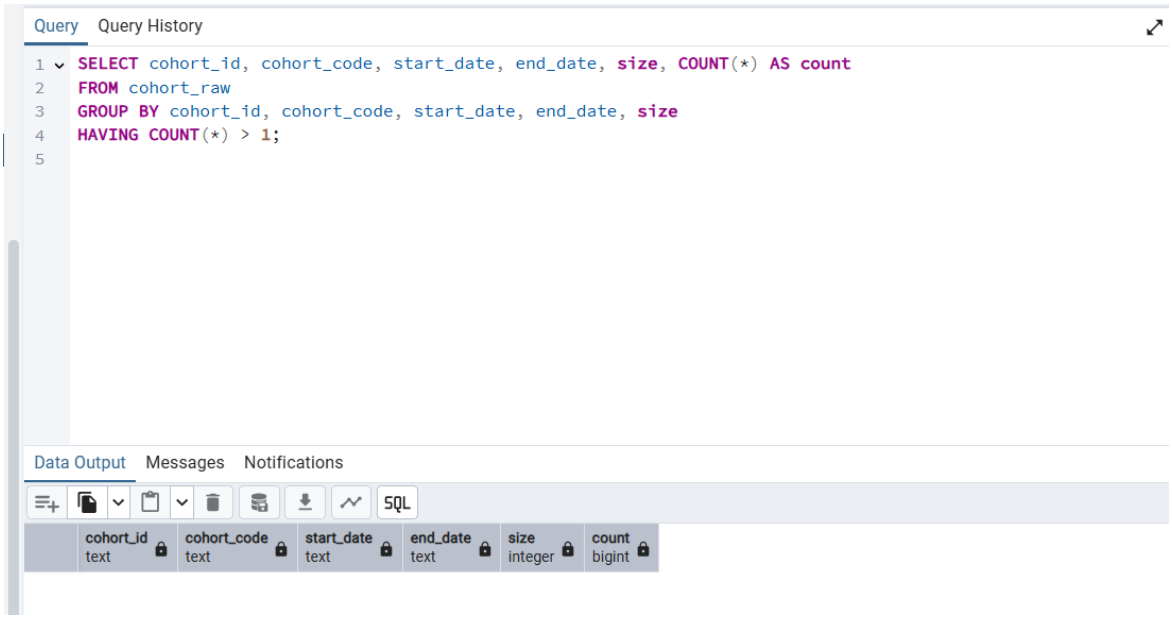


Part 2: Detect Duplicate Records

3. Fully Duplicate Rows (every column repeated)

```
SELECT cohort_id, cohort_code, start_date, end_date, size, COUNT(*) AS count
FROM cohort_raw
GROUP BY cohort_id, cohort_code, start_date, end_date, size
HAVING COUNT(*) > 1;
```

- **No duplications in this.**



4. Partially Duplicate Rows (e.g., duplicate `cohort_code`)

To check if a column supposed to be unique has duplicates:

```
SELECT cohort_code, COUNT(*) AS count
FROM cohort_raw
GROUP BY cohort_code
HAVING COUNT(*) > 1;
```

- **No partial duplications also.**

What to Look For:

Feature	Insight
NULL Values	Indicate incomplete or missing data that may need cleaning
Fully Duplicate Rows	Often accidental — likely safe to remove
Partially Duplicate Rows	May indicate inconsistent or repeated identifiers (e.g., cohort_code)

Step 4: Spot Outliers and Anomalies

- Use techniques like the Interquartile Range (IQR) to find unusual values that might distort analysis.

Research how outliers are detected using SQL and statistical methods.

Interquartile Range (IQR) Method Overview

Steps:

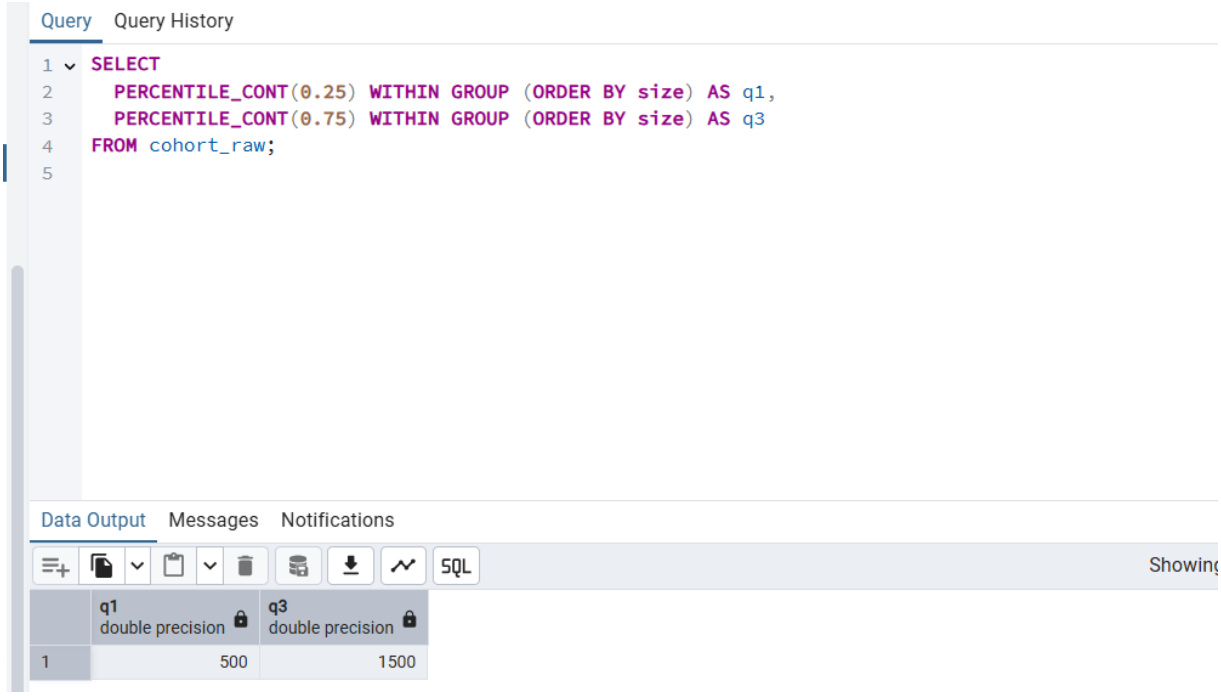
1. Calculate Q1 (25th percentile) and Q3 (75th percentile)
2. Compute IQR = Q3 − Q1
3. Define outlier bounds:
    - Lower Bound = Q1 − 1.5 × IQR
    - Upper Bound = Q3 + 1.5 × IQR

4. Flag values outside these bounds as outliers

PostgreSQL Queries to Detect Outliers

1. Get Q1, Q3, and IQR

```
SELECT
  PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY size) AS q1,
  PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY size) AS q3
FROM cohort_raw;
```



Once you have Q1 and Q3:

Q1= 500 , Q3= 1500

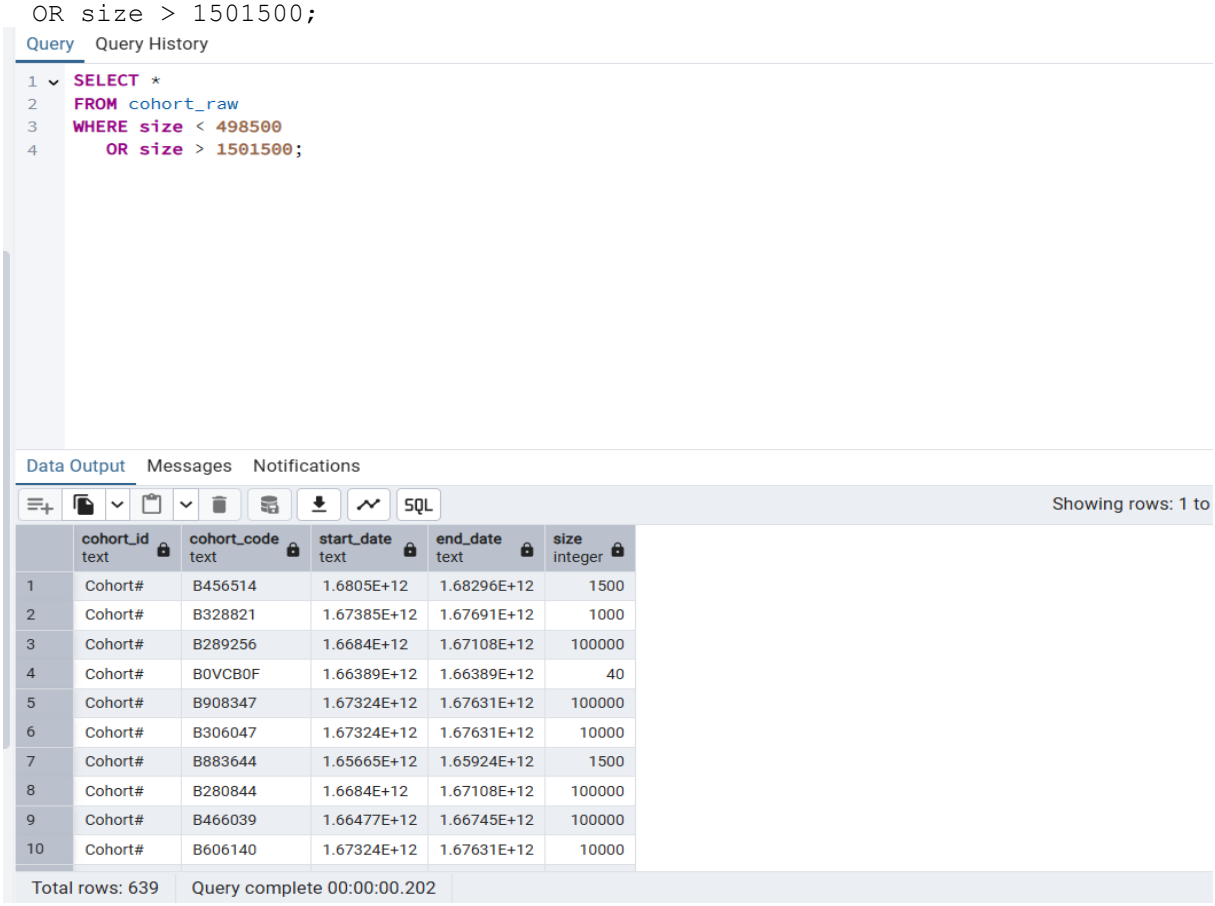
```
IQR = Q3 - Q1
IQR = 1500 - 500
IQR = 1000
Lower Bound = Q1 - 1.5 * IQR
            = 500 - 1.5 * 1000
            = 498500
Upper Bound = Q3 + 1.5 * IQR
            = 1500 + 1.5 * 1000
            = 1501500
```

---

2. Find Outliers Using Calculated Bounds

Assume you manually calculate and plug in the bounds, e.g.:

```
SELECT *
FROM cohort_raw
WHERE size < 498500
   OR size > 1501500;
```



	cohort_id text	cohort_code text	start_date text	end_date text	size integer
1	Cohort#	B456514	1.6805E+12	1.68296E+12	1500
2	Cohort#	B328821	1.67385E+12	1.67691E+12	1000
3	Cohort#	B289256	1.6684E+12	1.67108E+12	100000
4	Cohort#	B0VCB0F	1.66389E+12	1.66389E+12	40
5	Cohort#	B908347	1.67324E+12	1.67631E+12	100000
6	Cohort#	B306047	1.67324E+12	1.67631E+12	10000
7	Cohort#	B883644	1.65665E+12	1.65924E+12	1500
8	Cohort#	B280844	1.6684E+12	1.67108E+12	100000
9	Cohort#	B466039	1.66477E+12	1.66745E+12	100000
10	Cohort#	B606140	1.67324E+12	1.67631E+12	10000

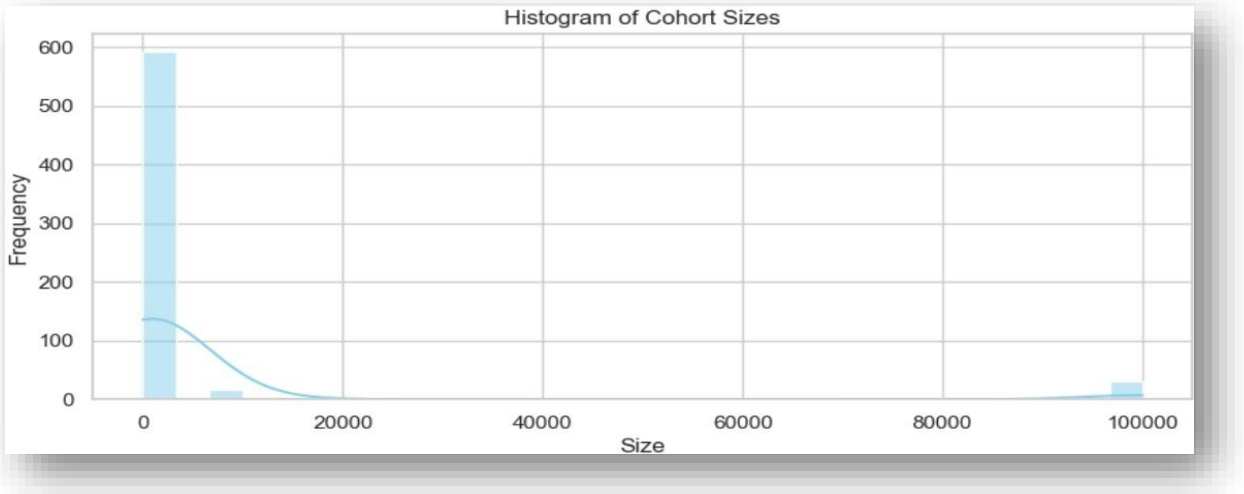
What to Look For:

Metric	Purpose
Q1 & Q3	Measure the range where most data lies
Outliers	Any value falling far outside this range
Next Step	Review outliers — correct, cap, or remove as needed

Step 5: Visualize Data Trends

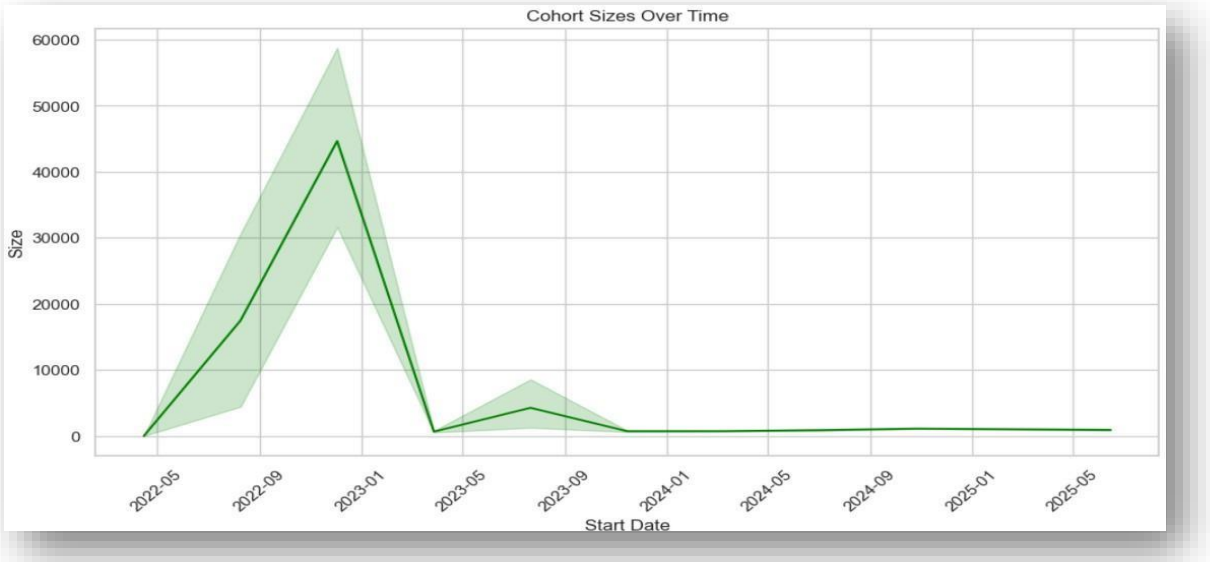
- Use graphs like histograms, line charts, and box plots to identify patterns.
- Analyze relationships between different variables.

Performed in Python

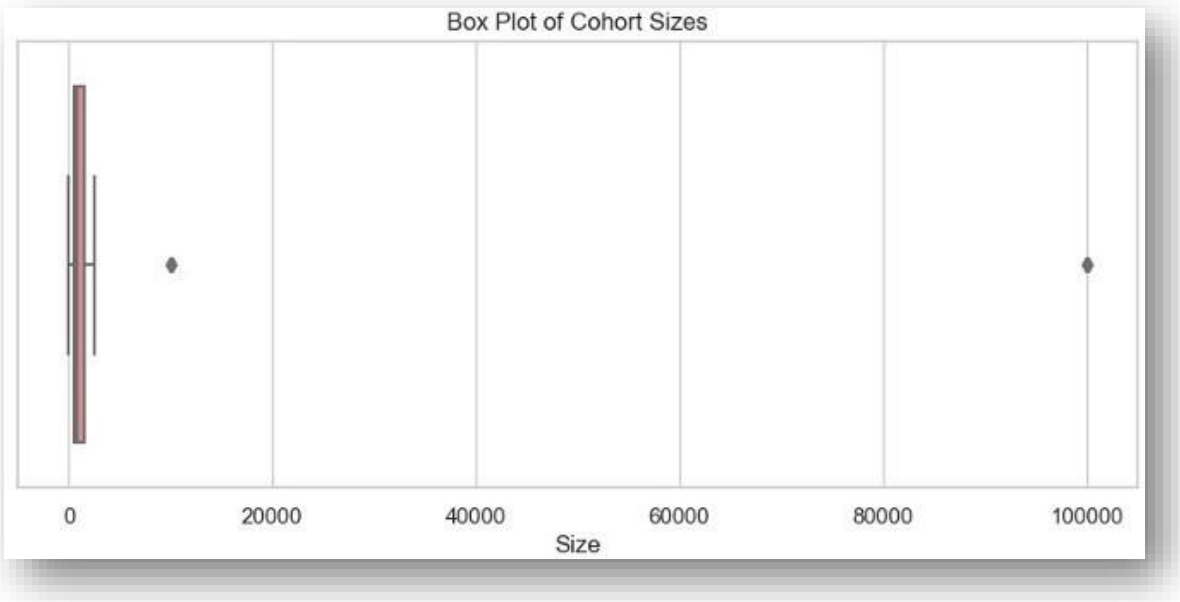


Histogram

Line Chart



Box Plot


Box Plot of Cohort Sizes
0
20000
40000
60000
80000
100000
Size

Dataset: Learneropportunity_Raw

Performing the Exploratory Data Analysis

Step 1: Understand the Dataset

- View the first few rows to get an overview of the data.
- Check column names, data types, and the total number of records.

Research how to inspect a table's structure and data in PostgreSQL.

Objective:

Understand the structure of the dataset by inspecting:

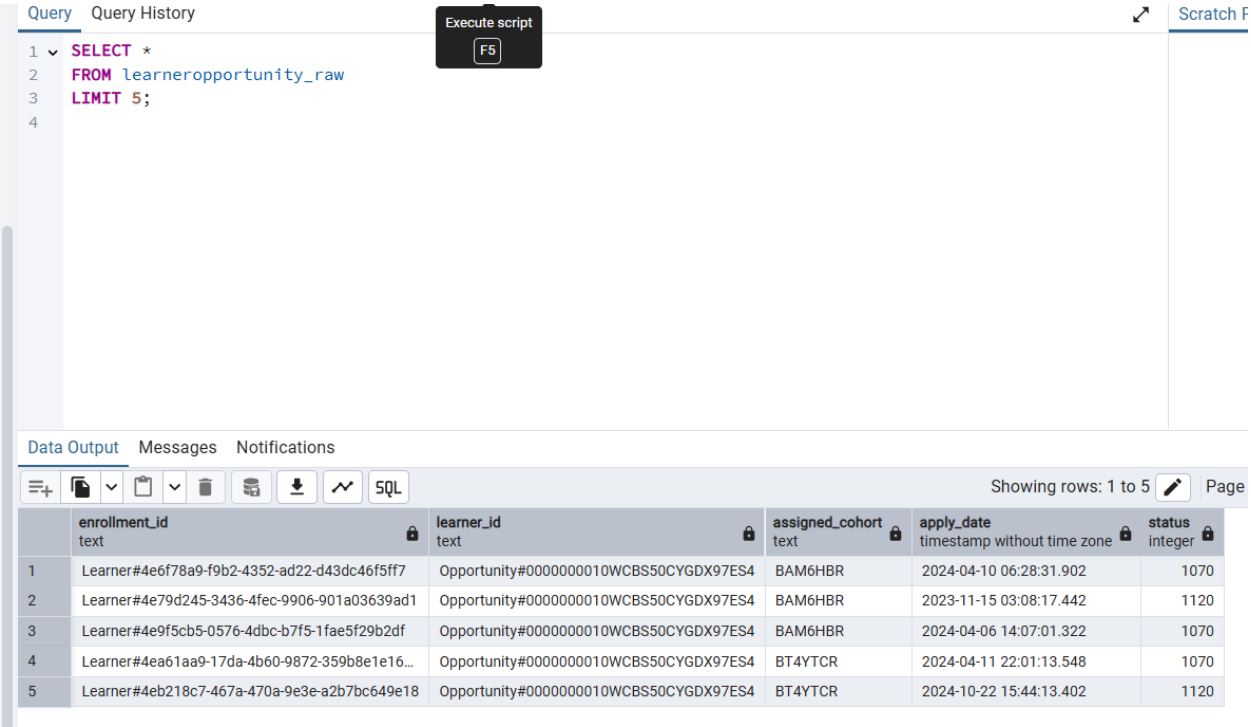
- First few rows
- Column names
- Data types
- Total number of records

Queries in PostgreSQL:

1. View First Few Rows:

```
SELECT *
FROM learneropportunity_raw
LIMIT 5;
```

This gives a quick snapshot of the data format and sample values.

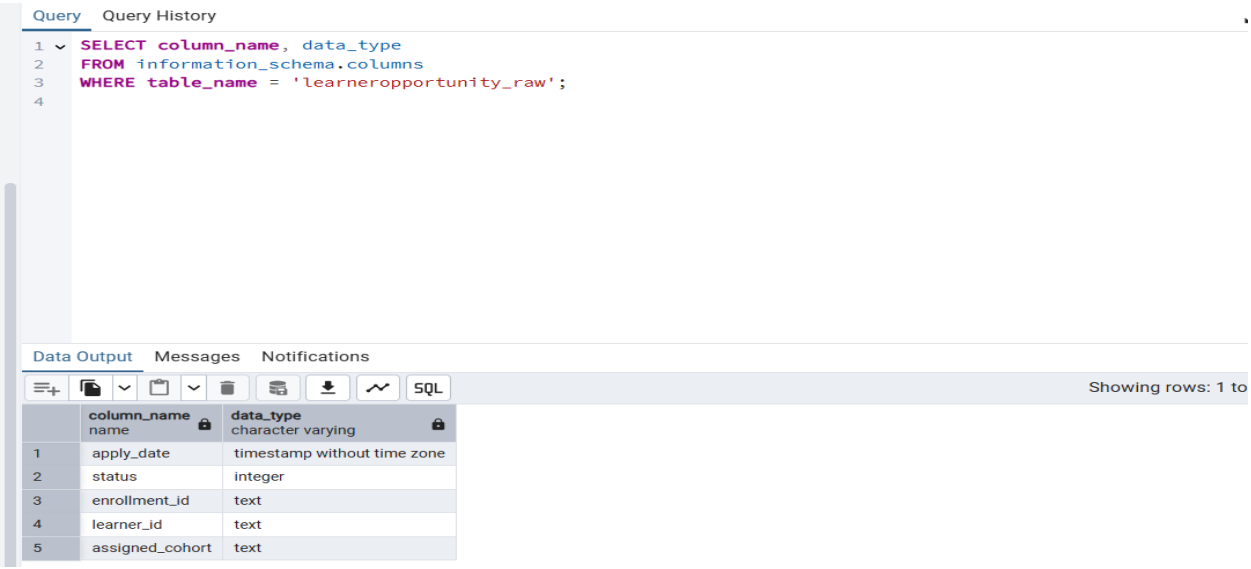


	enrollment_id text	learner_id text	assigned_cohort text	apply_date timestamp without time zone	status integer
1	Learner#4e6f78a9-f9b2-4352-ad22-d43dc46f5ff7	Opportunity#0000000010WCBS50CYGDX97ES4	BAM6HBR	2024-04-10 06:28:31.902	1070
2	Learner#4e79d245-3436-4fec-9906-901a03639ad1	Opportunity#0000000010WCBS50CYGDX97ES4	BAM6HBR	2023-11-15 03:08:17.442	1120
3	Learner#4e9f5cb5-0576-4dbc-b7f5-1fae5f29b2df	Opportunity#0000000010WCBS50CYGDX97ES4	BAM6HBR	2024-04-06 14:07:01.322	1070
4	Learner#4ea61aa9-17da-4b60-9872-359b8e1e16…	Opportunity#0000000010WCBS50CYGDX97ES4	BT4YTCR	2024-04-11 22:01:13.548	1070
5	Learner#4eb218c7-467a-470a-9e3e-a2b7bc649e18	Opportunity#0000000010WCBS50CYGDX97ES4	BT4YTCR	2024-10-22 15:44:13.402	1120

2. Check Column Names and Data Types:

```
SELECT column_name, data_type
FROM information_schema.columns
WHERE table_name = 'learneropportunity_raw';
```

Lists each column with its data type (e.g., text, integer, timestamp).



	column_name name	data_type character varying
1	apply_date	timestamp without time zone
2	status	integer
3	enrollment_id	text
4	learner_id	text
5	assigned_cohort	text

3. Get Total Number of Records:

```
SELECT COUNT(*)
FROM learneropportunity_raw;
```

 Confirms how many records are present in the dataset.

Step 2: Summarize the Data

- Generate basic statistics like mean, median, min, max, and standard deviation to understand the distribution.

Explore how to compute summary statistics in SQL.

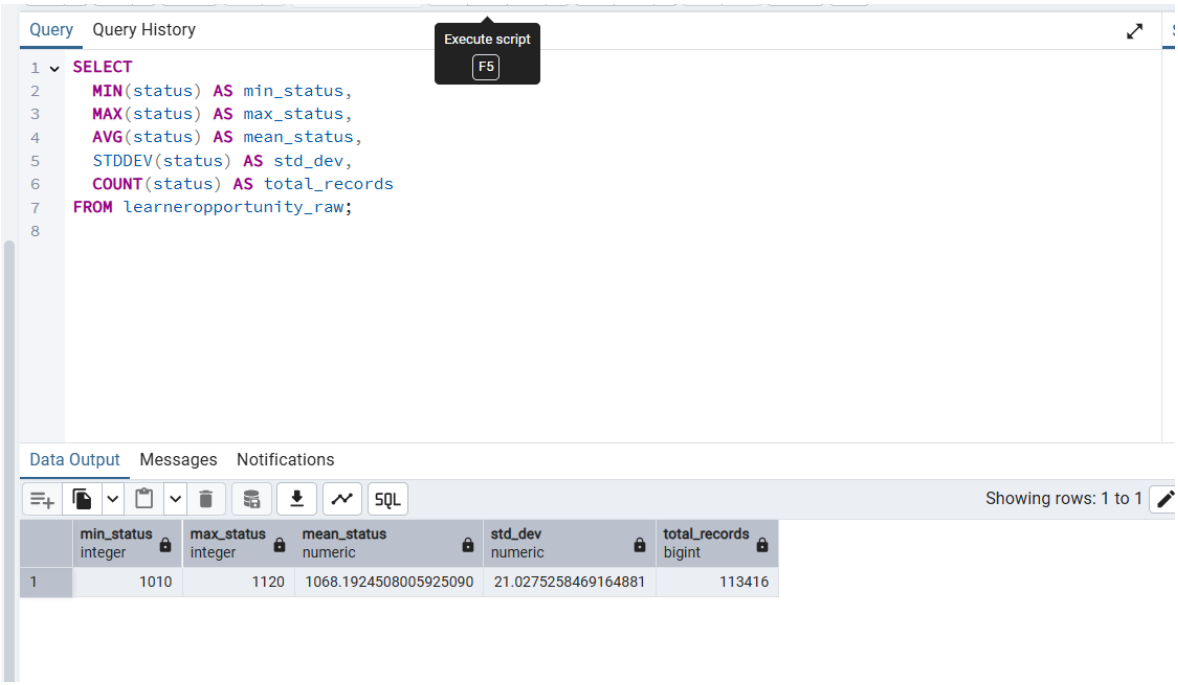
QL Queries for Summary Statistics (PostgreSQL):

---

1. Basic Descriptive Stats:

```
SELECT
  MIN(status) AS min_status,
  MAX(status) AS max_status,
```

```
  AVG(status) AS mean_status,
  STDDEV(status) AS std_dev,
  COUNT(status) AS total_records
FROM learneropportunity_raw;
```

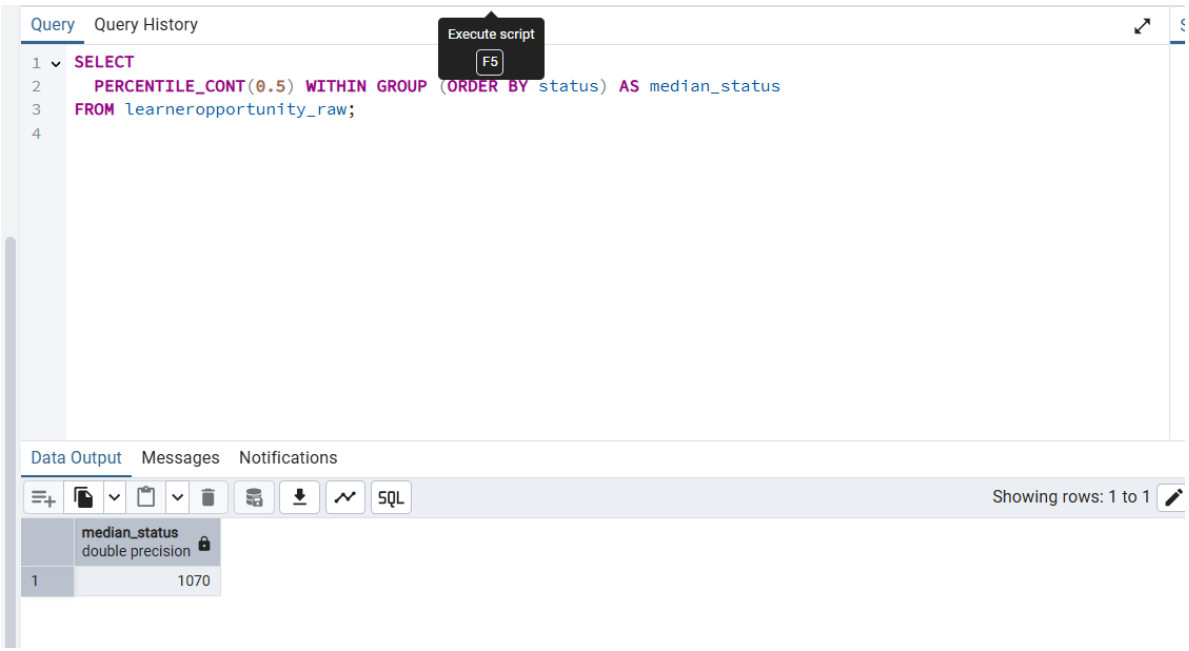


---

2. Median:

(PostgreSQL does not have a built-in `MEDIAN()` function, but you can calculate it using `PERCENTILE_CONT`.)

```
SELECT
  PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY status) AS median_status
FROM learneropportunity_raw;
```



What These Queries Will Show:

Statistic	Description
`MIN(status)`	Smallest status value (possibly earliest stage)
`MAX(status)`	Highest status code (possibly latest/confirmed/complete)
`AVG(status)`	Mean status value — useful if numerical progression has meaning
`STDDEV(status)`	Standard deviation — shows variation or tightness of clustering
`COUNT(status)`	Total number of status entries (non-null)
`MEDIAN(status)`	Middle status code — useful for detecting skew from outliers

Step 3: Identify Missing and Duplicate Values

- Detect missing data to determine if any cleaning is needed.
- Identify duplicate records that may need removal.

Look up methods to check for missing values and duplicates in PostgreSQL

Objective:

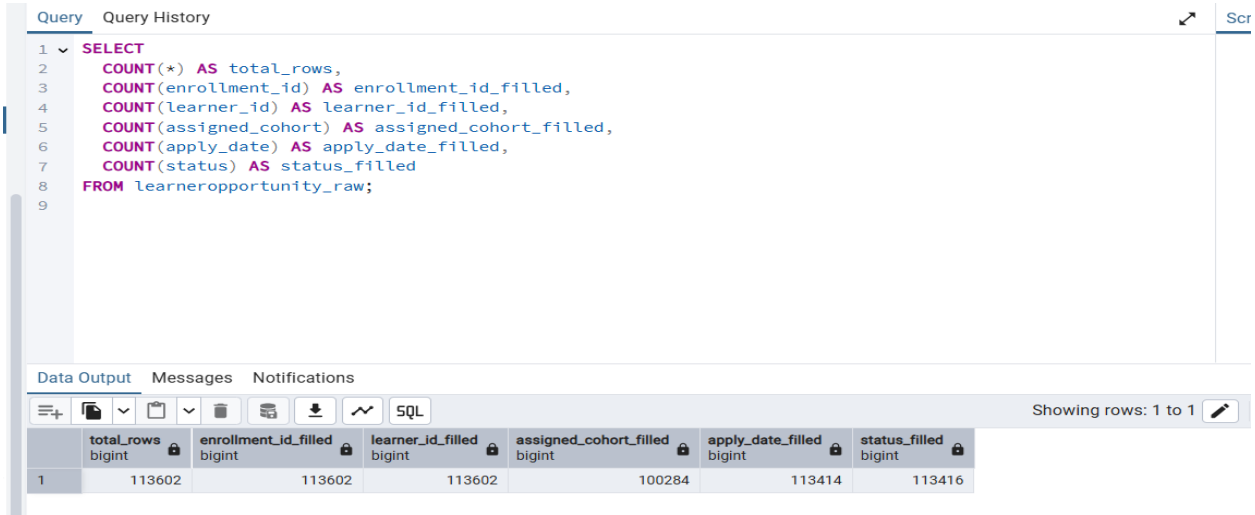
- Detect missing (NULL) values
- Identify fully or partially duplicate records

Part A: Check for Missing Values

1. Count of NULLs for All Columns

```
SELECT
  COUNT(*) AS total_rows,
  COUNT(enrollment_id) AS enrollment_id_filled,
  COUNT(learner_id) AS learner_id_filled,
  COUNT(assigned_cohort) AS assigned_cohort_filled,
  COUNT(apply_date) AS apply_date_filled,
  COUNT(status) AS status_filled
FROM learneropportunity_raw;
```

 Compare filled counts with total rows to detect missing values.

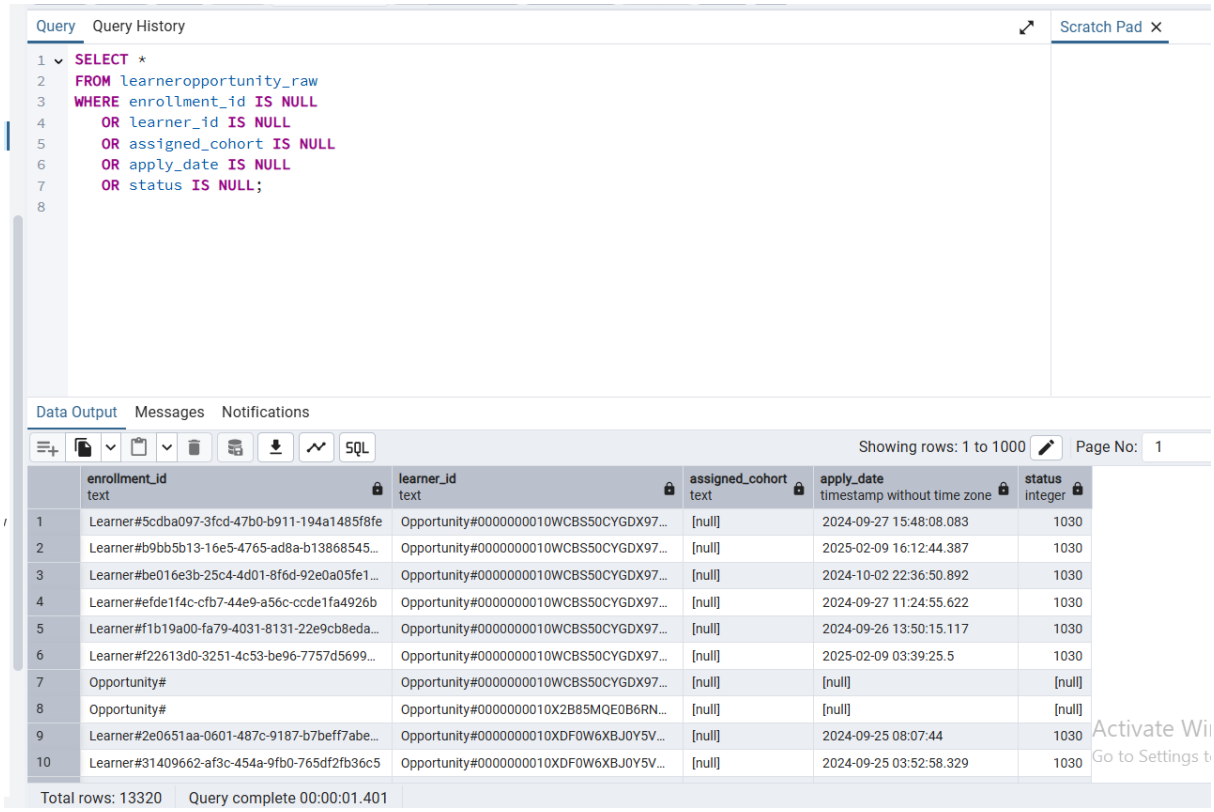


---

2. View Records With Any NULL

```
SELECT *
FROM learneropportunity_raw
WHERE enrollment_id IS NULL
   OR learner_id IS NULL
   OR assigned_cohort IS NULL
   OR apply_date IS NULL
   OR status IS NULL;
```

🔍 *Allows visual inspection of missing fields.*



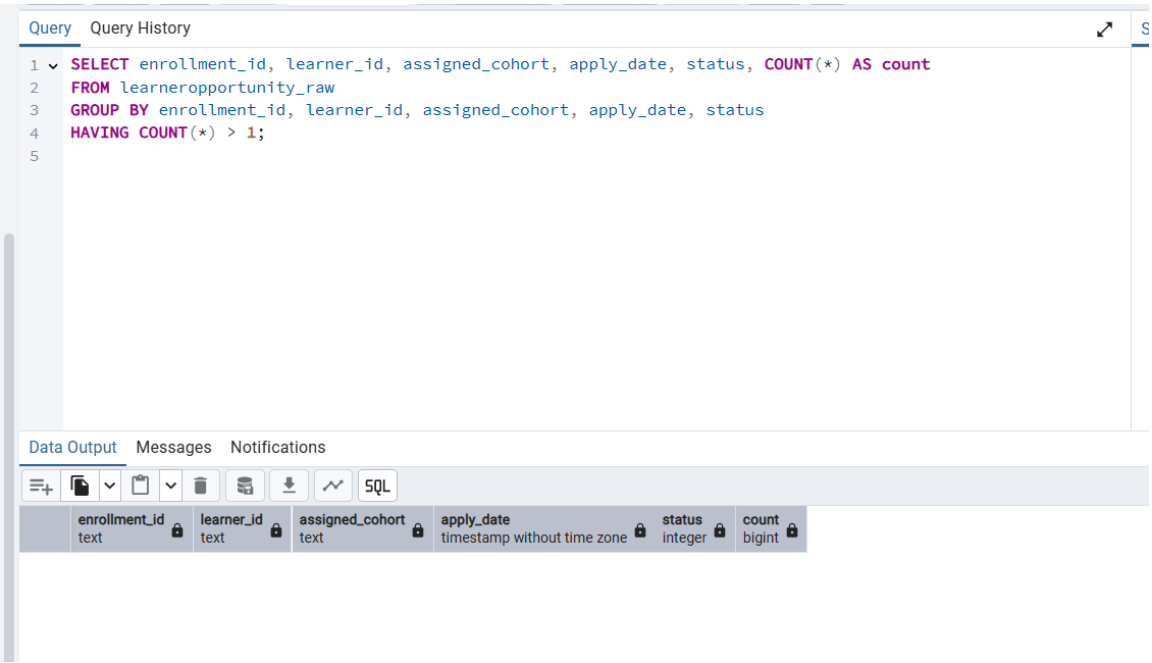
Part B: Check for Duplicates

3. Fully Duplicate Records

```
SELECT enrollment_id, learner_id, assigned_cohort, apply_date, status,
COUNT(*) AS count
FROM learneropportunity_raw
GROUP BY enrollment_id, learner_id, assigned_cohort, apply_date, status
HAVING COUNT(*) > 1;
```

📌 *These rows are identical in all fields — likely safe to remove.*

No full duplicate rows



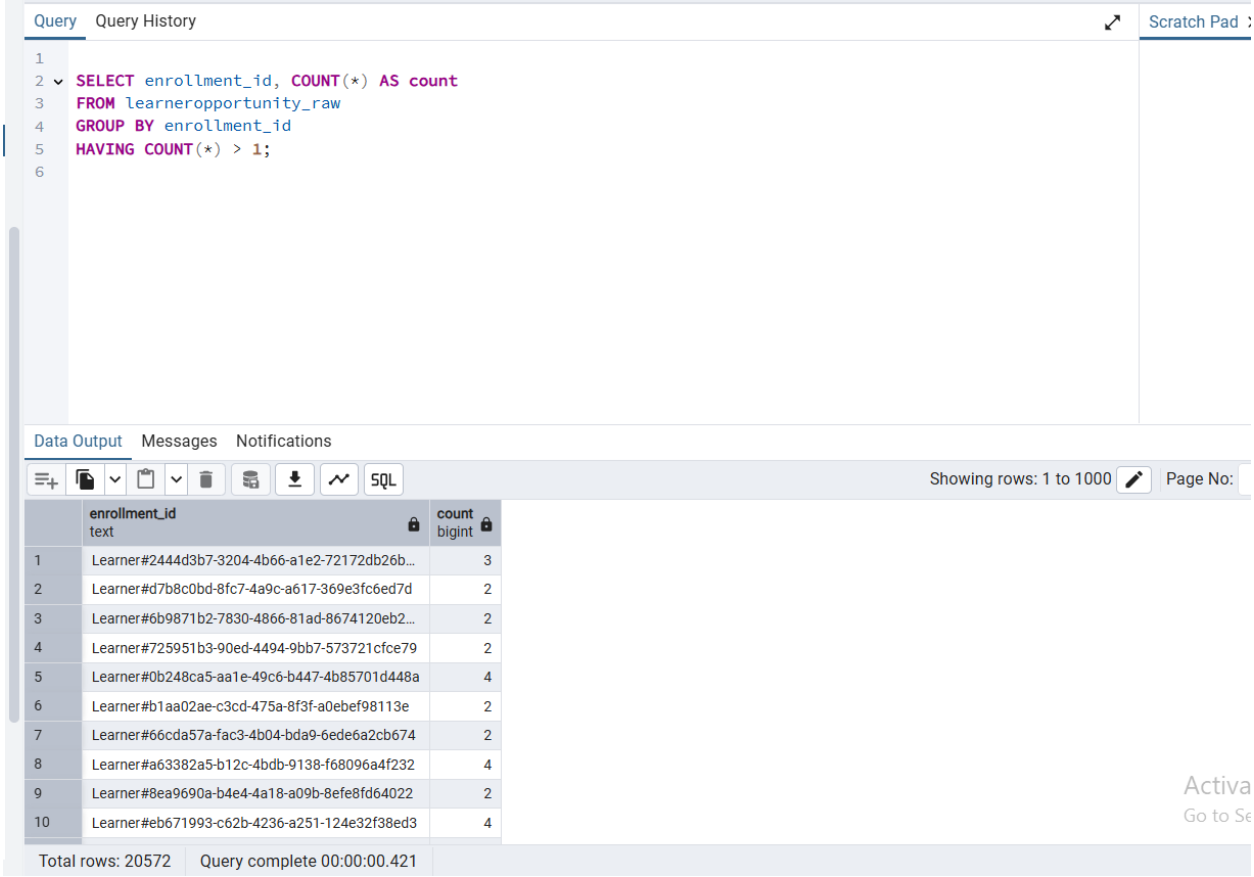
---

4. Partially Duplicate Enrollment IDs

```
SELECT enrollment_id, COUNT(*) AS count
FROM learneropportunity_raw
GROUP BY enrollment_id
HAVING COUNT(*) > 1;
```

⚠ *If `enrollment_id` is expected to be unique, this flags potential issues.*

Yes the enrollment_id is a duplicate value column that contains values 20572



What to Do Next:

Issue Type	Meaning
NULLs	May indicate missing inputs, timing issues, or data pipeline errors
Full Duplicates	Usually safe to delete unless time-sensitive duplicates are allowed
Partial Duplicates	May indicate errors in key uniqueness or merging process

Step 4: Spot Outliers and Anomalies

- Use techniques like the Interquartile Range (IQR) to find unusual values that might distort analysis.

Research how outliers are detected using SQL and statistical methods.

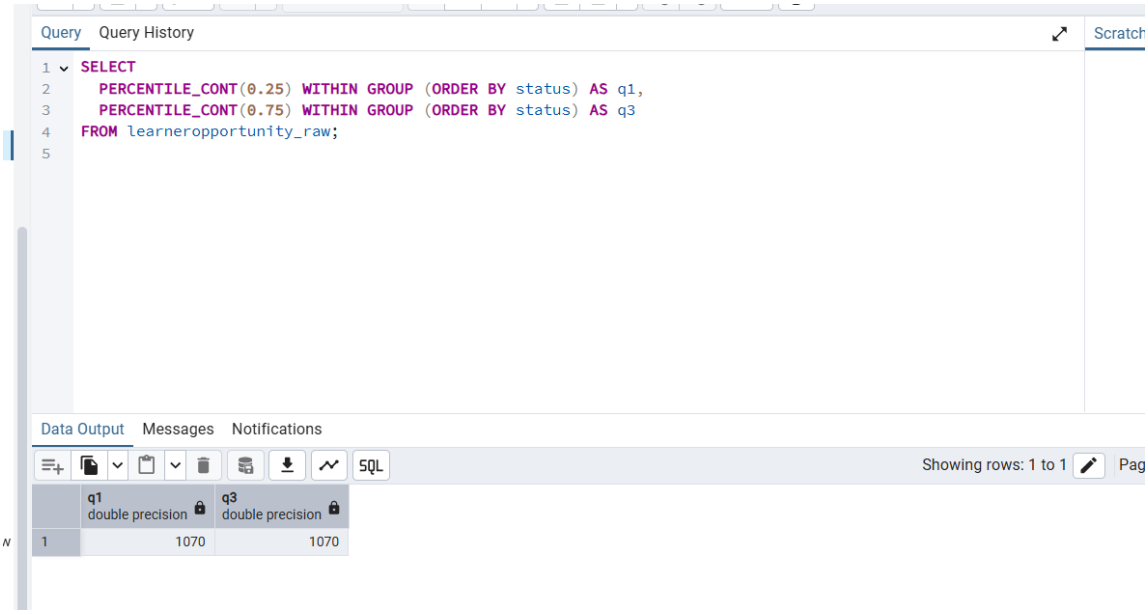
Objective:

Use the **Interquartile Range (IQR)** method to identify **outliers** — values that are unusually high or low and may distort analysis or reflect errors.

Step-by-Step: Outlier Detection Using IQR in SQL

1. Get Q1 (25th percentile) and Q3 (75th percentile)

```
SELECT
  PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY status) AS q1,
  PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY status) AS q3
FROM learneropportunity_raw;
```



---

2. Calculate IQR and Bounds (do this manually after running above)

```
IQR = Q3 - Q1
```

```
Q1 = 1070 , Q3 = 1070
IQR = 1070 - 1070
IQR = 0
Lower Bound = Q1 - 1.5 × IQR
            = 1070 - 1.5 * 0
            = 0
Upper Bound = Q3 + 1.5 × IQR
            = 1070 + 1.5 * 0
            = 0
```

You'll substitute the calculated bounds into the next query.

🔍 3. Find Outliers (values outside bounds)

```
SELECT *
FROM learneropportunity_raw
WHERE status < <lower_bound> OR status > <upper_bound>;
```

Query | Query History | Scratch Pa

```
1 SELECT *
2 FROM learneropportunity_raw
3 WHERE status < 1070 OR status > 1070;
4
```

Data Output | Messages | Graph Visualiser | Notifications

Showing rows: 1 to 1000 | Page No:

	enrollment_id text	learner_id text	assigned_cohort text	apply_date timestamp without time zone	status integer
1	Learner#4e79d245-3436-4fec-9906-901a03639a...	Opportunity#0000000010WCBS50CYGDX97ES4	BAM6HBR	2023-11-15 03:08:17.442	1120
2	Learner#4eb218c7-467a-470a-9e3e-a2b7bc649e...	Opportunity#0000000010WCBS50CYGDX97ES4	BT4YTCR	2024-10-22 15:44:13.402	1120
3	Learner#4ef49684-e9b0-40a9-b07e-07bfa78bdbc5	Opportunity#0000000010WCBS50CYGDX97ES4	BGRQZ2N	2024-10-08 09:50:06.608	1120
4	Learner#63cea3a8-615f-4cc9-b230-096d500923...	Opportunity#000000000G127E8VYE08TXBT6X	B880483	2022-08-19 01:53:08.792	1120
5	Learner#4f32e611-9b57-4bea-9b12-f6f5829f544b	Opportunity#0000000010WCBS50CYGDX97ES4	BAM6HBR	2024-03-30 05:41:03.93	1120
6	Learner#4fab276f-96c0-4094-8c93-8e1260022b33	Opportunity#0000000010WCBS50CYGDX97ES4	BAM6HBR	2023-05-18 08:48:49.645	1120
7	Learner#4ff4bd04-ebad-4631-9b7f-ad3b2cba393c	Opportunity#0000000010WCBS50CYGDX97ES4	BAM6HBR	2023-03-17 07:32:38.085	1120
8	Learner#501a32a4-434b-46e6-b150-2e6c55fffbea	Opportunity#0000000010WCBS50CYGDX97ES4	BAM6HBR	2023-07-03 00:21:15.044	1120
9	Learner#50235d57-9c98-4382-9bd6-4f74ab6d8e...	Opportunity#0000000010WCBS50CYGDX97ES4	BAM6HBR	2024-04-02 10:53:06.562	1120
10	Learner#50498fd5-1968-41bb-bf58-a205ee0bef10	Opportunity#0000000010WCBS50CYGDX97ES4	BAM6HBR	2023-08-24 18:19:31.951	1120

- **All values are 1070**, or nearly all — which is why:
  - Q1 = Q3 = 1070
  - IQR = 0

    - So **there are no outliers**
- This is **not an error**, just an indicator that:
    - The `status` field has **very low variance**
    - It's probably **not useful for outlier analysis**

Conclusion:

No outliers exist in the `status` column using IQR, because all values are clustered at 1070.

Dataset: Marketing Campaign Data All Accounts (2023-2024)(Detail1)

Performing the Exploratory Data Analysis

Step 1: Understand the Dataset

- View the first few rows to get an overview of the data.
- Check column names, data types, and the total number of records.

Research how to inspect a table's structure and data in PostgreSQL.

Step Objective:
Understand the structure of the dataset by inspecting:
• First few rows
• Column names
• Data types
• Total number of records

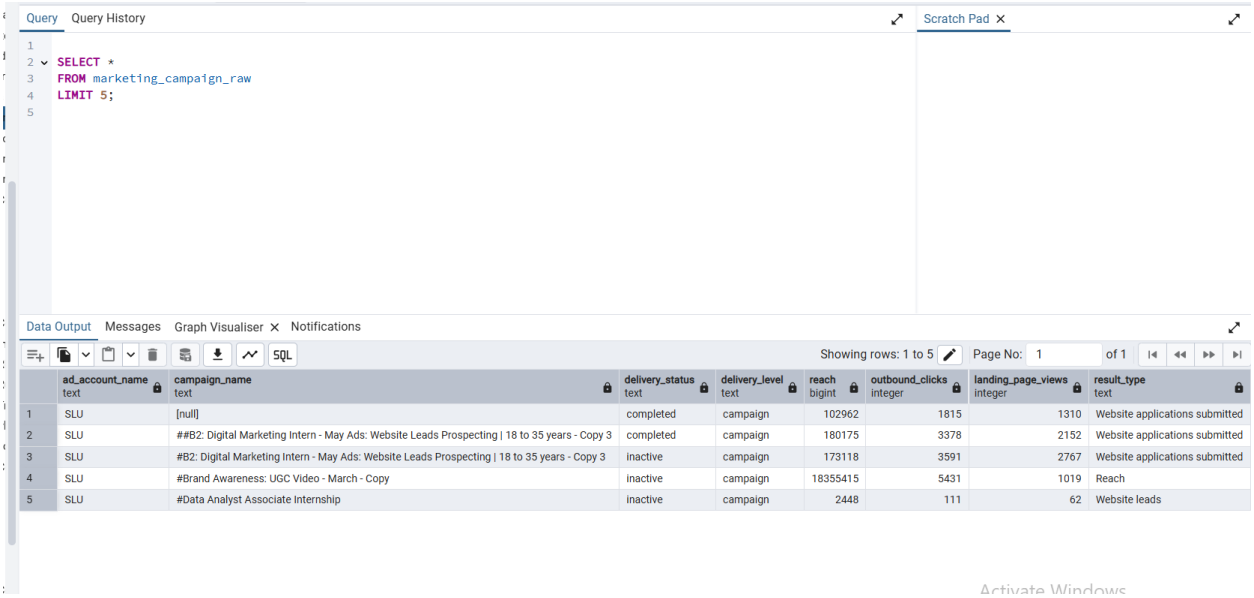
Queries in PostgreSQL:

---

1. View First Few Rows:

```
SELECT *
FROM marketing_campaign_raw
LIMIT 5;
```

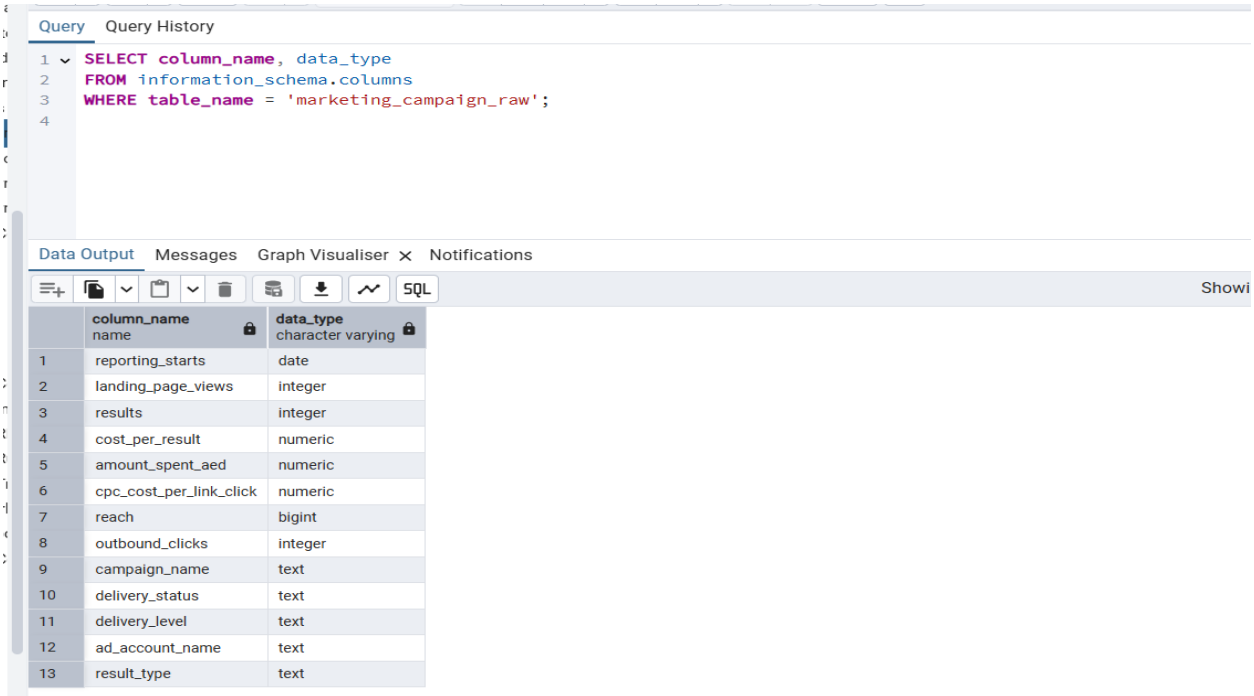
➞ Gives a quick snapshot of sample records, formatting, and value types.



2. Check Column Names and Data Types:

```
SELECT column_name, data_type
FROM information_schema.columns
WHERE table_name = 'marketing_campaign_raw';
```

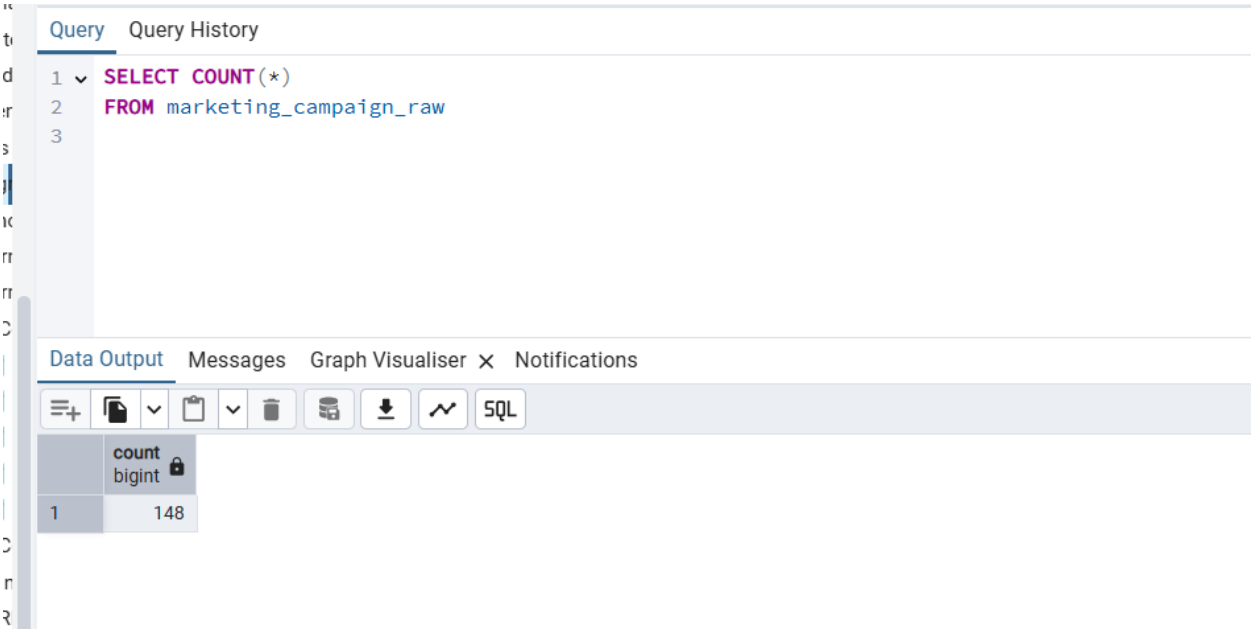
➞ Lists each column along with its data type — such as `text`, `integer`, `timestamp`, etc.



3. Get Total Number of Records:

```
SELECT COUNT(*)
FROM marketing_campaign_raw;
```

➞ Confirms how many rows (records) exist in the dataset



Step 2: Summarize the Data

- Generate basic statistics like mean, median, min, max, and standard deviation to understand the distribution.

Explore how to compute summary statistics in SQL.

Focused Numeric Columns from Your Sample:

- Reach
- Outbound clicks
- Landing page views
- Results
- Cost per result
- Amount spent (AED)
- CPC (cost per link click)

PostgreSQL Queries for Descriptive Statistics

---

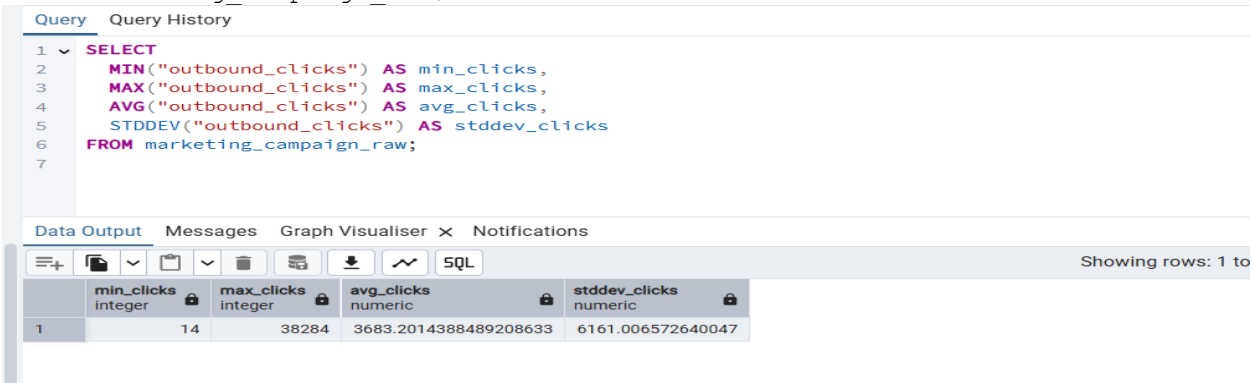
1. 📈 Basic Descriptive Statistics:

- **Reach**

```
SELECT
   MIN(reach) AS min_reach,
   MAX(reach) AS max_reach,
   AVG(reach) AS avg_reach,
   STDDEV(reach) AS stddev_reach
FROM marketing_campaign_raw;
```

- **Outbound clicks**

```
SELECT
  MIN("outbound_clicks") AS min_clicks,
  MAX("outbound_clicks") AS max_clicks,
  AVG("outbound_clicks") AS avg_clicks,
  STDDEV("outbound_clicks") AS stddev_clicks
FROM marketing_campaign_raw;
```

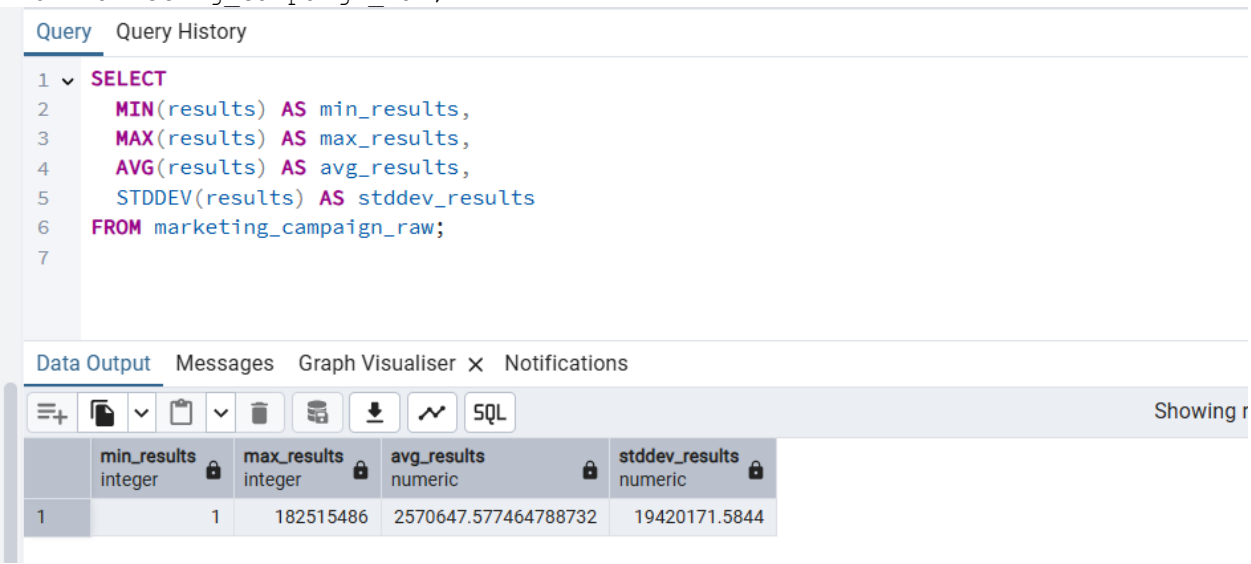


- **Landing page views**

```
SELECT
  MIN("landing_page_views") AS min_lpv,
  MAX("landing_page_views") AS max_lpv,
  AVG("landing_page_views") AS avg_lpv,
  STDDEV("landing_page_views") AS stddev_lpv
FROM marketing_campaign_raw;
```

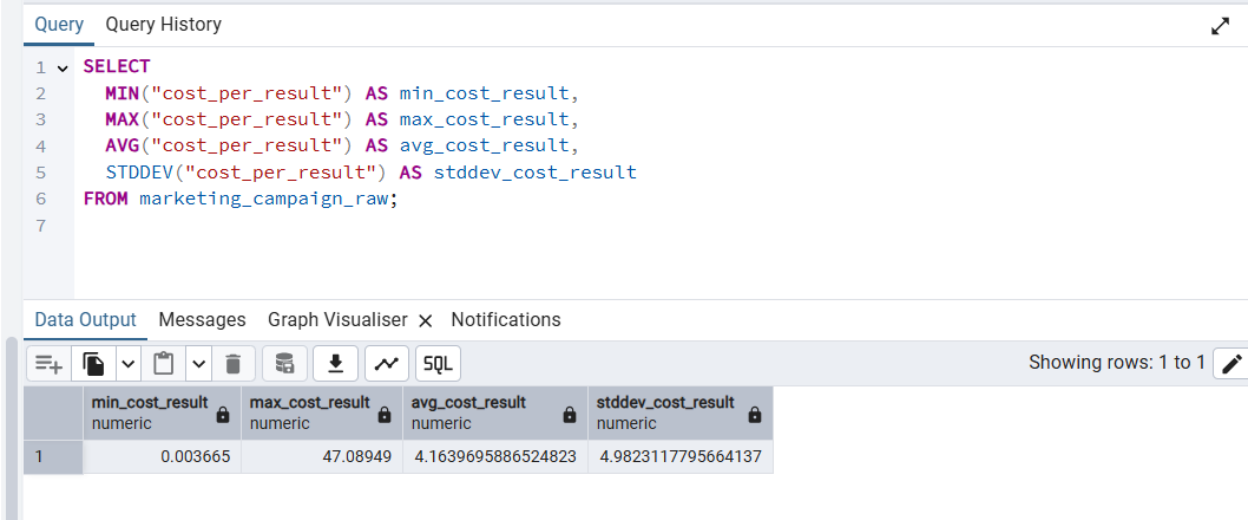
- **Results**

```
SELECT
  MIN(results) AS min_results,
  MAX(results) AS max_results,
  AVG(results) AS avg_results,
  STDDEV(results) AS stddev_results
FROM marketing_campaign_raw;
```



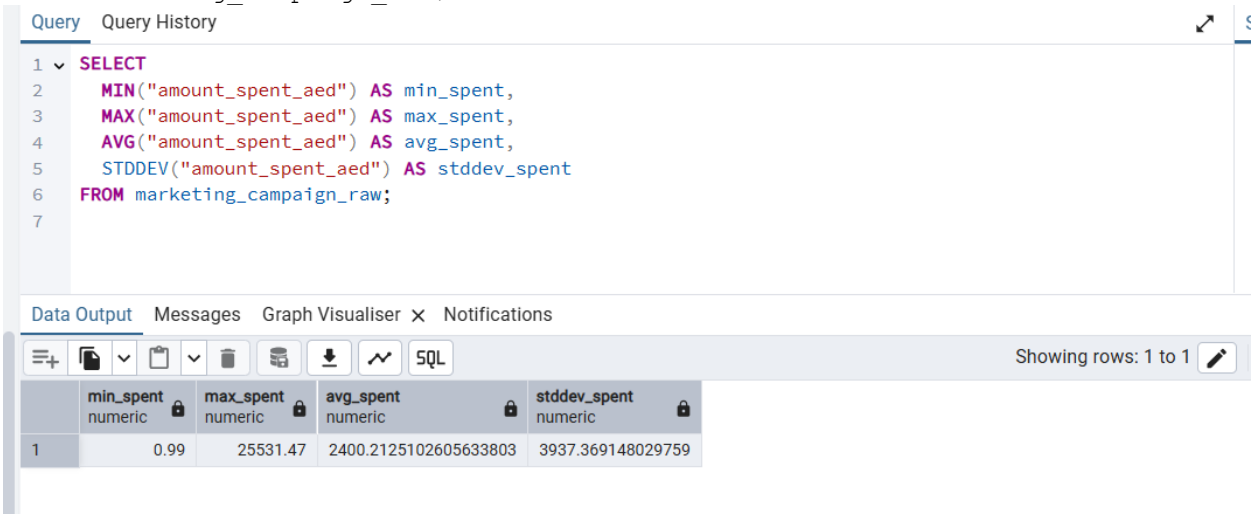
- **Cost per result**

```
SELECT
  MIN("cost_per_result") AS min_cost_result,
  MAX("cost_per_result") AS max_cost_result,
  AVG("cost_per_result") AS avg_cost_result,
  STDDEV("Cost per result") AS stddev_cost_result
FROM marketing_campaign_raw;
```



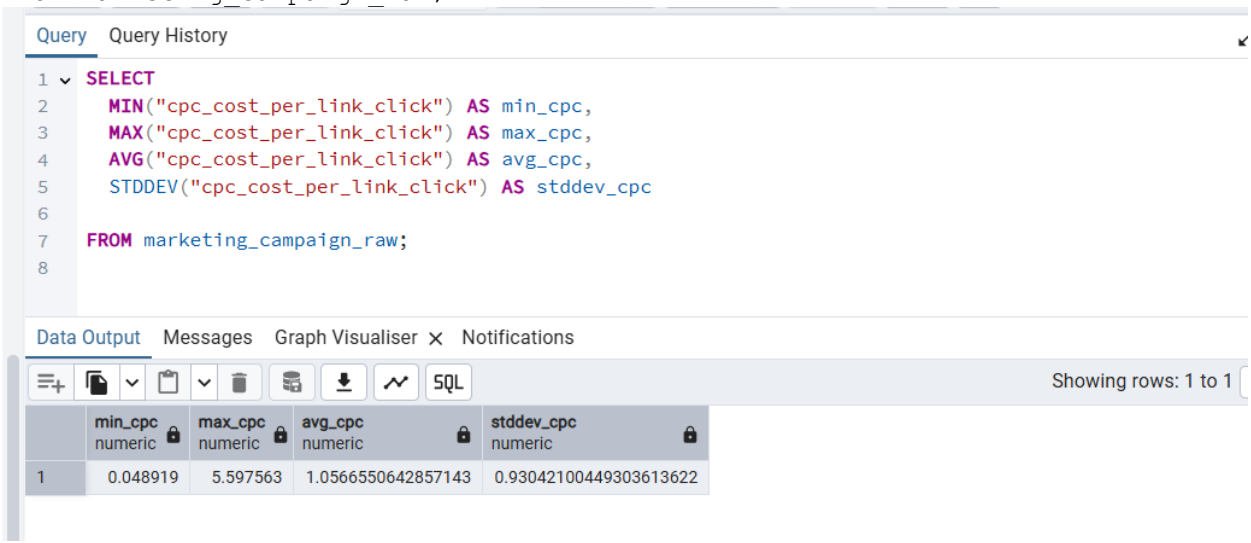
- **Amount spent (AED)**

```
SELECT
  MIN("amount_spent_aed") AS min_spent,
  MAX("amount_spent_aed") AS max_spent,
  AVG("amount_spent_aed") AS avg_spent,
  STDDEV("amount_spent_aed)") AS stddev_spent
FROM marketing_campaign_raw;
```



- **CPC (cost per link click)**

```
SELECT
  MIN("cpc_cost_per_link_click") AS min_cpc,
  MAX("cpc_cost_per_link_click") AS max_cpc,
  AVG("cpc_cost_per_link_click") AS avg_cpc,
  STDDEV("cpc_cost_per_link_click") AS stddev_cpc

FROM marketing_campaign_raw;
```



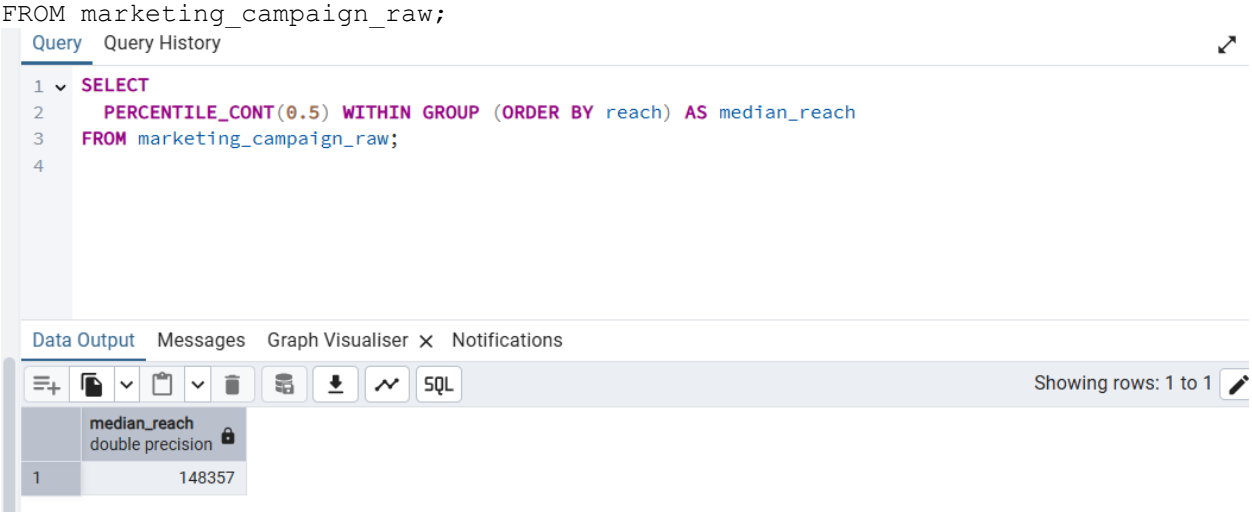
---

2. □ Median for Each Numeric Column (requires `PERCENTILE_CONT`)

Example for one column — apply similarly for others:

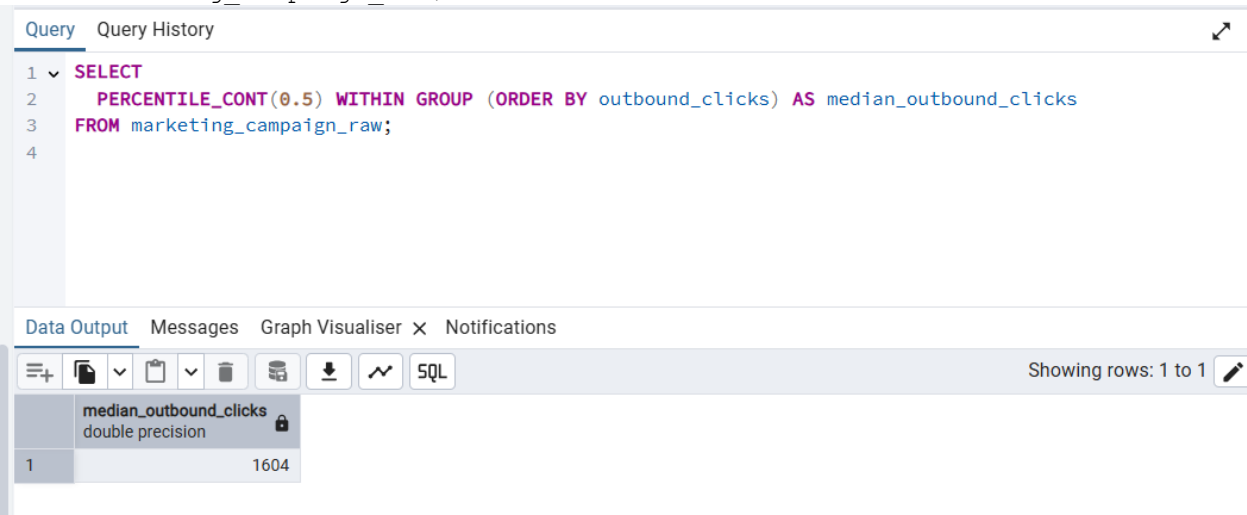
1-Median of `reach`:

```
SELECT
  PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY reach) AS median_reach
FROM marketing_campaign_raw;
```



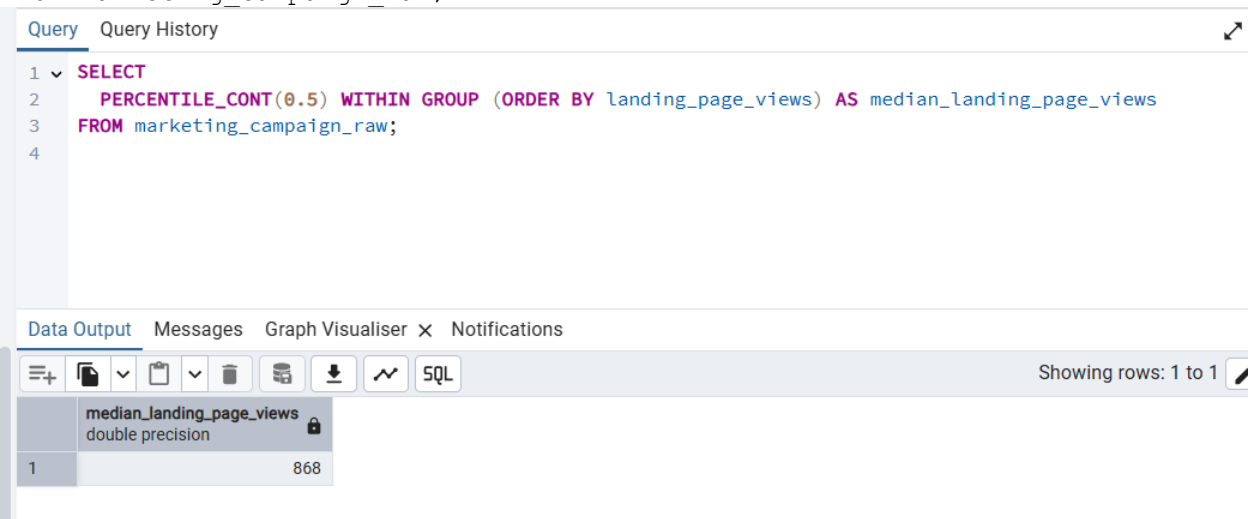
2-Median of `outbound_clicks`:

```
SELECT
  PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY outbound_clicks) AS
median_outbound_clicks
FROM marketing_campaign_raw;
```



3-Median of `landing_page_views`:

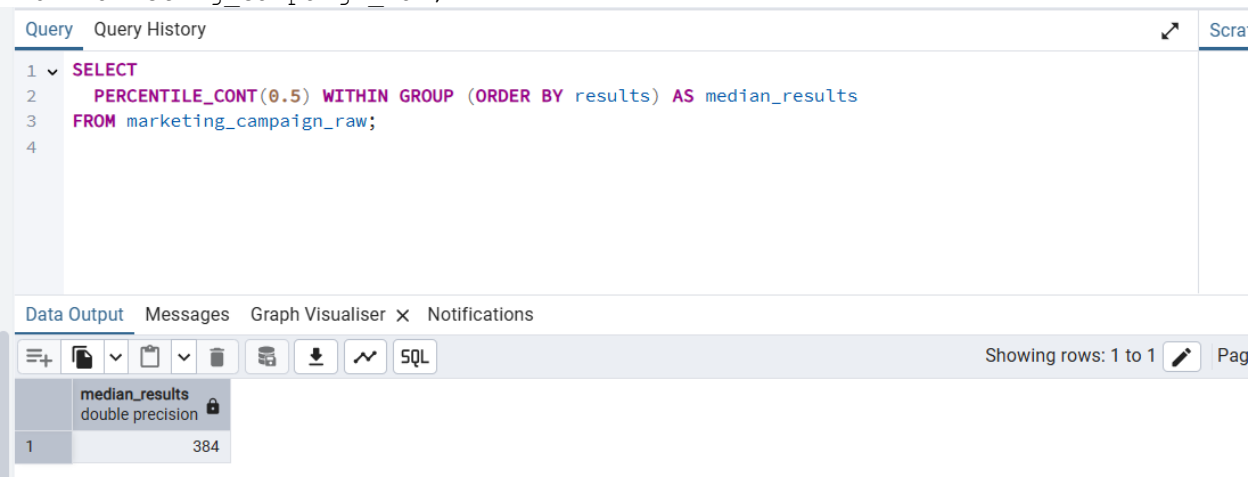
```
SELECT
  PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY landing_page_views) AS
median_landing_page_views
FROM marketing_campaign_raw;
```



4-Median of `results`:

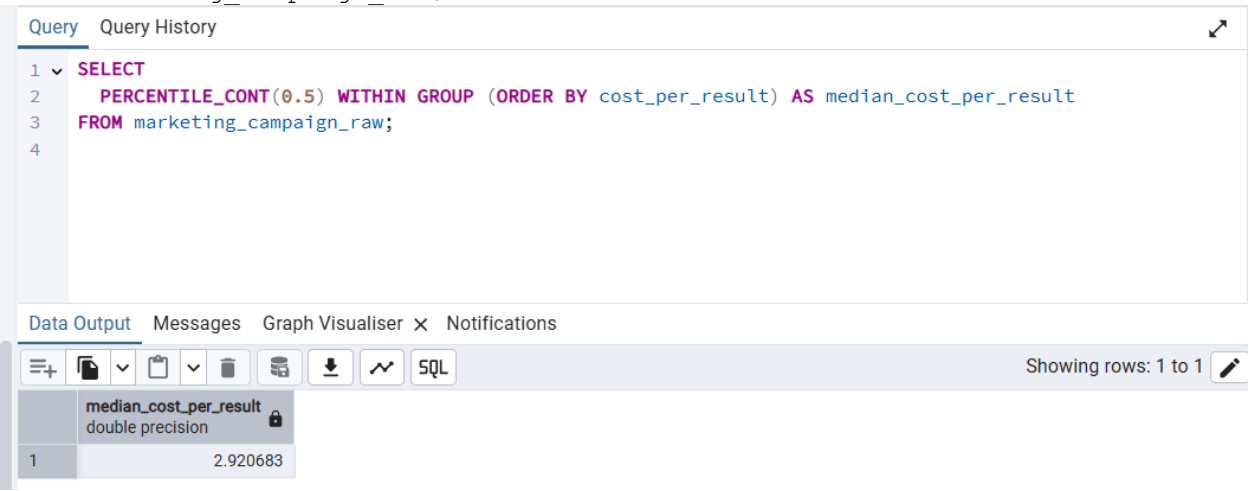
```
SELECT
  PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY results) AS median_results
```

```
FROM marketing_campaign_raw;
```



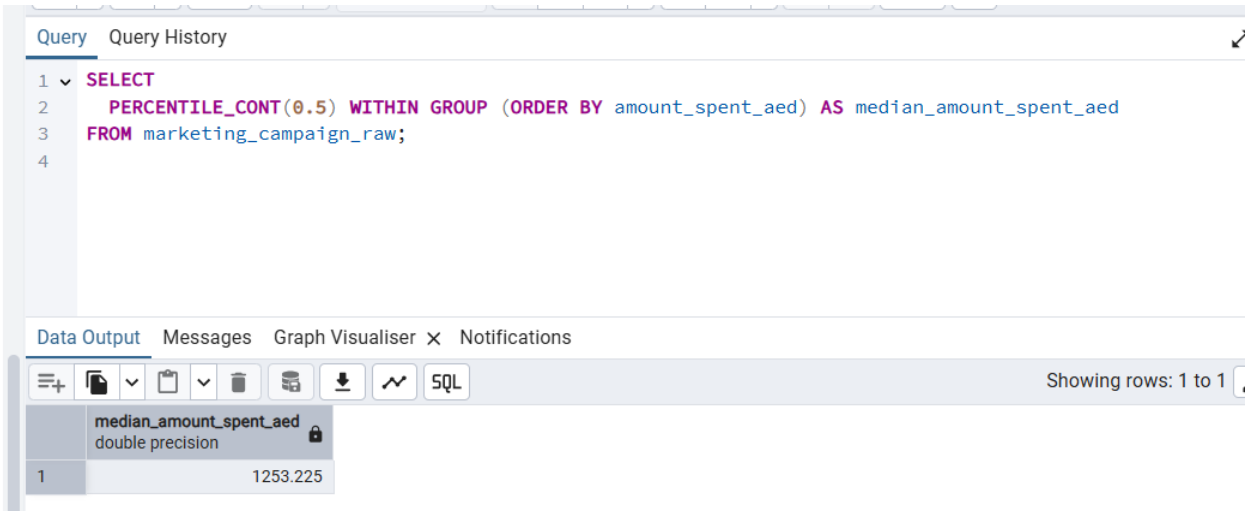
5-Median of `cost_per_result`:

```
SELECT
  PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY cost_per_result) AS
median_cost_per_result
FROM marketing_campaign_raw;
```



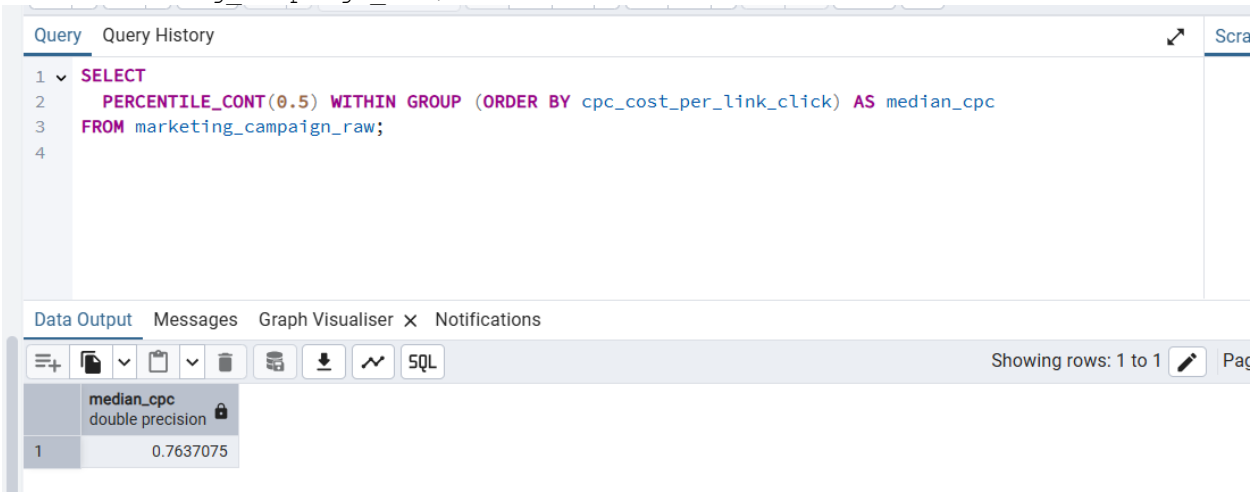
6-Median of `amount_spent_aed`:

```
SELECT
  PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY amount_spent_aed) AS
median_amount_spent_aed
FROM marketing_campaign_raw;
```



7-Median of `cpc_cost_per_link_click`:

```
SELECT
  PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY cpc_cost_per_link_click) AS
median_cpc
FROM marketing_campaign_raw;
```



What These Stats Reveal:

Metric	Insight Example
`MAX(Reach)`	Identifies your most-viewed campaign
`AVG(Outbound clicks)`	Shows how effective your campaigns are at driving traffic
`STDDEV(Amount Spent)`	Large spread may indicate outlier spending in some campaigns

Metric	Insight Example
`MEDIAN(Results)`	More stable than average, especially if there are outliers
`MIN(CPC)`	Useful for identifying most cost-efficient ad

Step 3: Identify Missing and Duplicate Values

- Detect missing data to determine if any cleaning is needed.
- Identify duplicate records that may need removal.

Look up methods to check for missing values and duplicates in PostgreSQL.

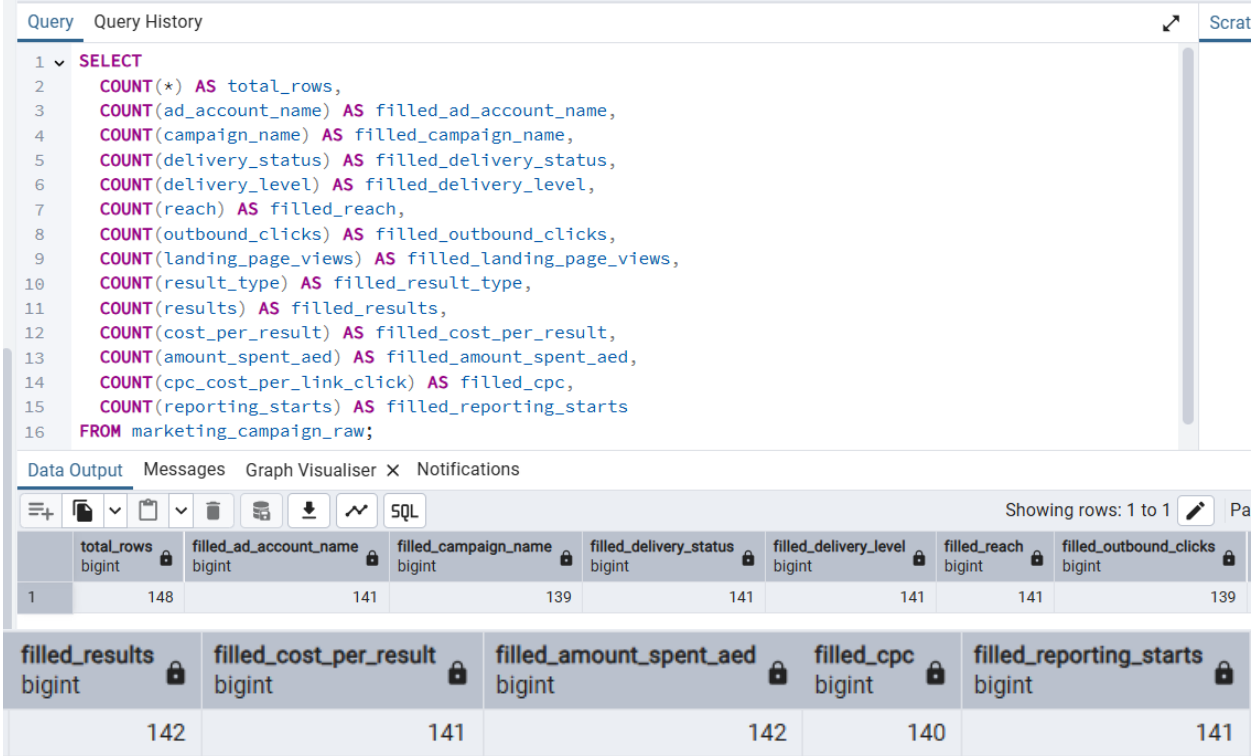
Dataset: `marketing_campaign_raw`
Objective:

- Detect missing (`NULL`) values in each column
- Identify duplicate records that might need removal or review

---

1. Check for Missing (NULL) Values in Each Column

```
SELECT
  COUNT(*) AS total_rows,
  COUNT(ad_account_name) AS filled_ad_account_name,
  COUNT(campaign_name) AS filled_campaign_name,
  COUNT(delivery_status) AS filled_delivery_status,
  COUNT(delivery_level) AS filled_delivery_level,
  COUNT(reach) AS filled_reach,
  COUNT(outbound_clicks) AS filled_outbound_clicks,
  COUNT(landing_page_views) AS filled_landing_page_views,
  COUNT(result_type) AS filled_result_type,
  COUNT(results) AS filled_results,
  COUNT(cost_per_result) AS filled_cost_per_result,
  COUNT(amount_spent_aed) AS filled_amount_spent_aed,
  COUNT(cpc_cost_per_link_click) AS filled_cpc,
  COUNT(reporting_starts) AS filled_reporting_starts
FROM marketing_campaign_raw;
```



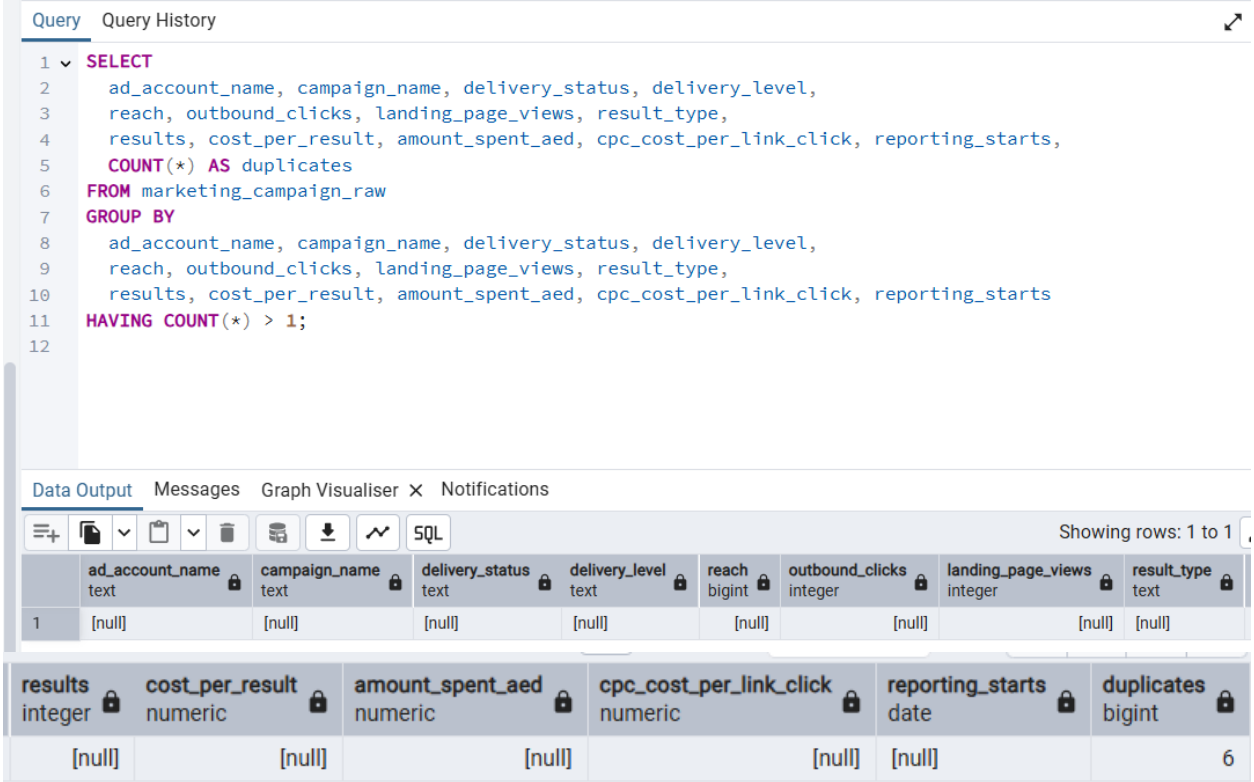
🔎 Interpretation:

- Compare each filled count with `total_rows`
- If any column has significantly fewer values, it's sparsely populated → candidate for cleaning/imputation.

---

2. Check for Duplicate Entire Rows

```
SELECT
  ad_account_name, campaign_name, delivery_status, delivery_level,
  reach, outbound_clicks, landing_page_views, result_type,
  results, cost_per_result, amount_spent_aed, cpc_cost_per_link_click,
reporting_starts,
  COUNT(*) AS duplicates
FROM marketing_campaign_raw
GROUP BY
  ad_account_name, campaign_name, delivery_status, delivery_level,
  reach, outbound_clicks, landing_page_views, result_type,
  results, cost_per_result, amount_spent_aed, cpc_cost_per_link_click,
reporting_starts
HAVING COUNT(*) > 1;
```



Interpretation:

- Shows exact duplicate rows
- If these exist, confirm whether they're valid or redundant before removing.

Count of All Duplicates:

```
SELECT COUNT(*) - COUNT(DISTINCT ROW(
  ad_account_name, campaign_name, delivery_status, delivery_level,
  reach, outbound_clicks, landing_page_views, result_type,
  results, cost_per_result, amount_spent_aed, cpc_cost_per_link_click,
reporting_starts
)) AS total_duplicate_rows
FROM marketing_campaign_raw;
```

Total Duplicates are 6.

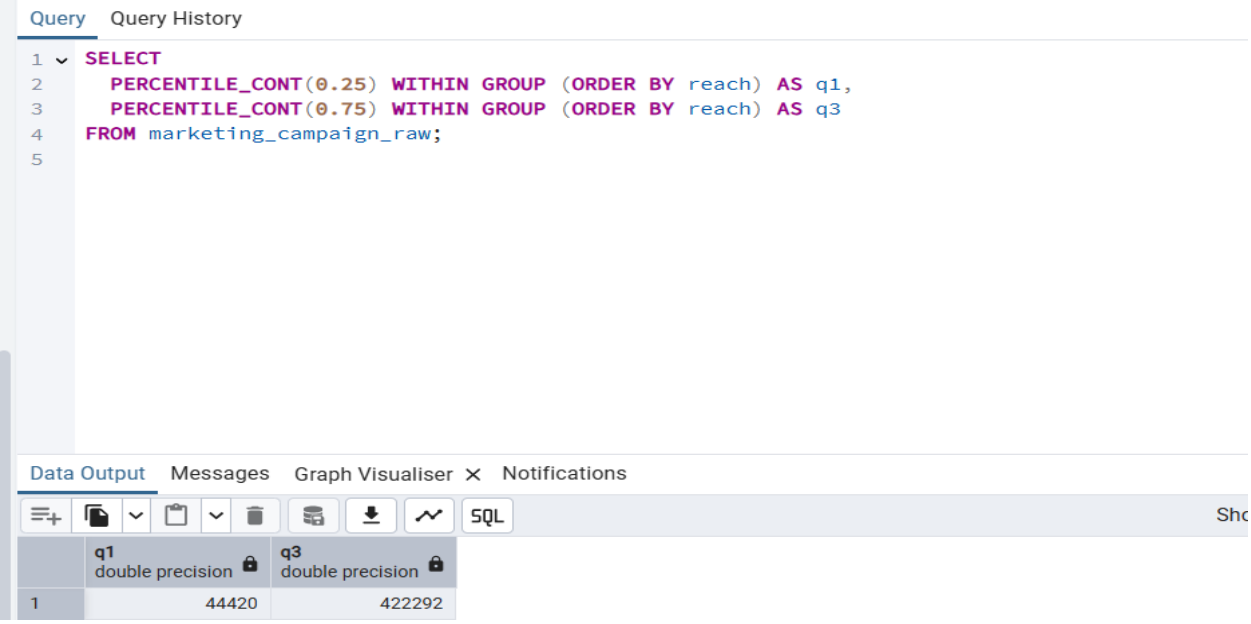
Step 4: Spot Outliers and Anomalies

- Use techniques like the Interquartile Range (IQR) to find unusual values that might distort analysis.

Research how outliers are detected using SQL and statistical methods.

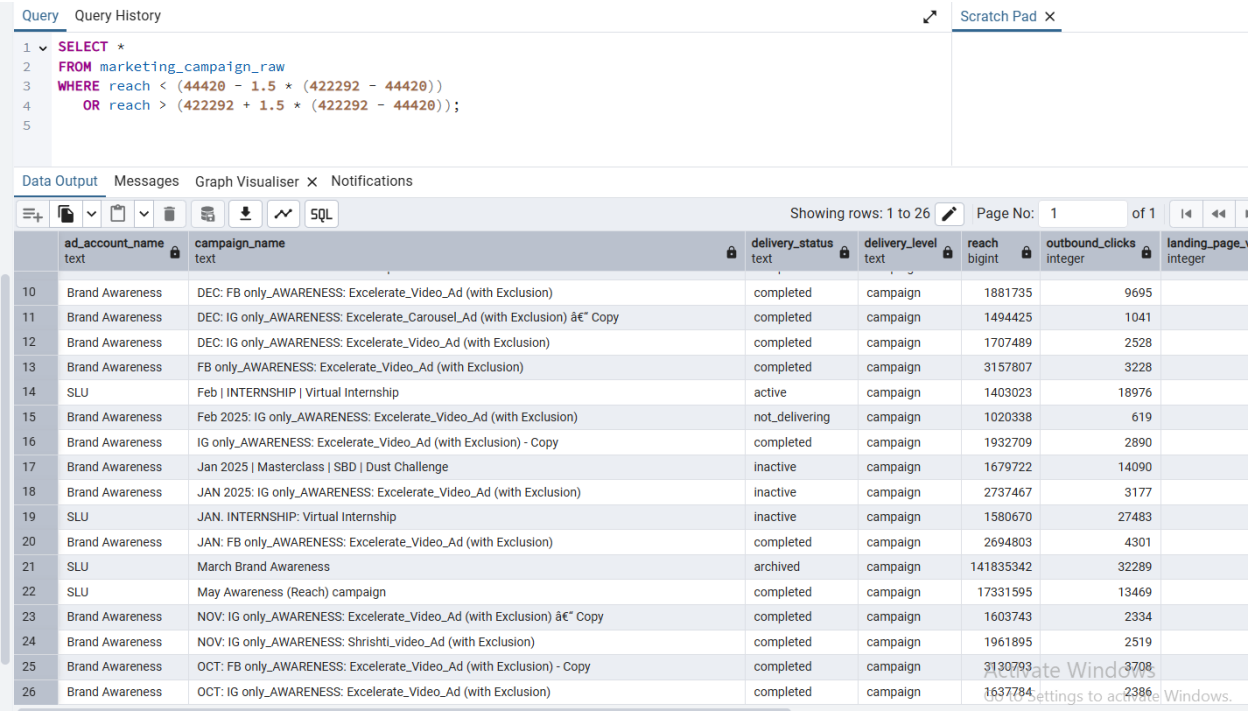
`reach` — Detect Outliers

```
SELECT
  PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY reach) AS q1,
  PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY reach) AS q3
FROM marketing_campaign_raw;
```



Q1 = 44420 , Q3 = 422292

```
SELECT *
FROM marketing_campaign_raw
WHERE reach < (44420 - 1.5 * (422292 - 44420))
   OR reach > (422292 + 1.5 * (422292 - 44420));
```



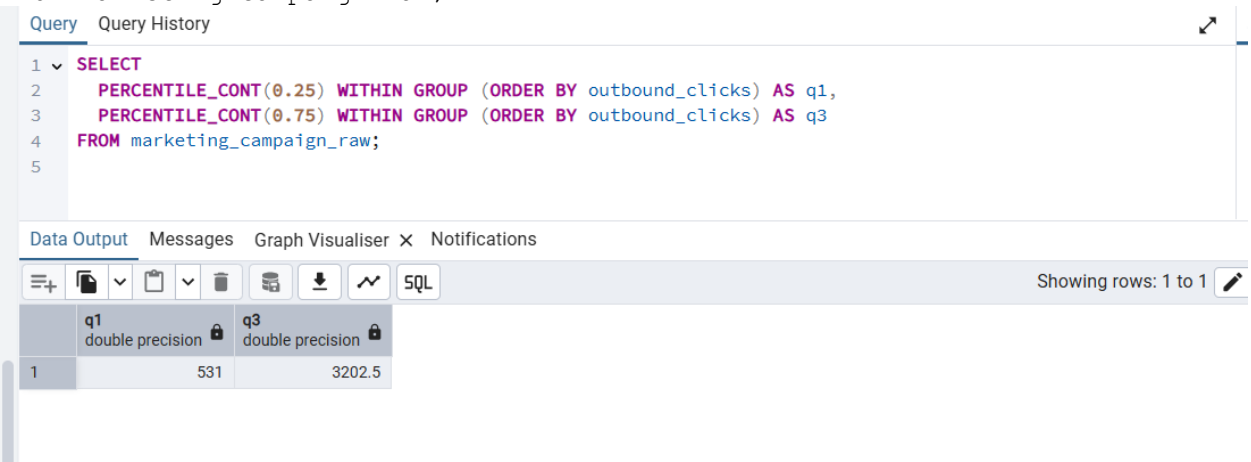
```
SELECT *
FROM marketing_campaign_raw
WHERE reach < (44420 - 1.5 * (422292 - 44420))
   OR reach > (422292 + 1.5 * (422292 - 44420));
```

Data Output | Messages | Graph Visualiser | Notifications — Showing rows: 1 to 26 — Page No: 1 of 1

	ad_account_name text	campaign_name text	delivery_status text	delivery_level text	reach bigint	outbound_clicks integer	landing_page_ integer
10	Brand Awareness	DEC: FB only_AWARENESS: Excelerate_Video_Ad (with Exclusion)	completed	campaign	1881735	9695	
11	Brand Awareness	DEC: IG only_AWARENESS: Excelerate_Carousel_Ad (with Exclusion) â€" Copy	completed	campaign	1494425	1041	
12	Brand Awareness	DEC: IG only_AWARENESS: Excelerate_Video_Ad (with Exclusion)	completed	campaign	1707489	2528	
13	Brand Awareness	FB only_AWARENESS: Excelerate_Video_Ad (with Exclusion)	completed	campaign	3157807	3228	
14	SLU	Feb | INTERNSHIP | Virtual Internship	active	campaign	1403023	18976	
15	Brand Awareness	Feb 2025: IG only_AWARENESS: Excelerate_Video_Ad (with Exclusion)	not_delivering	campaign	1020338	619	
16	Brand Awareness	IG only_AWARENESS: Excelerate_Video_Ad (with Exclusion) - Copy	completed	campaign	1932709	2890	
17	Brand Awareness	Jan 2025 | Masterclass | SBD | Dust Challenge	inactive	campaign	1679722	14090	
18	Brand Awareness	JAN 2025: IG only_AWARENESS: Excelerate_Video_Ad (with Exclusion)	inactive	campaign	2737467	3177	
19	SLU	JAN. INTERNSHIP: Virtual Internship	inactive	campaign	1580670	27483	
20	Brand Awareness	JAN: FB only_AWARENESS: Excelerate_Video_Ad (with Exclusion)	completed	campaign	2694803	4301	
21	SLU	March Brand Awareness	archived	campaign	141835342	32289	
22	SLU	May Awareness (Reach) campaign	completed	campaign	17331595	13469	
23	Brand Awareness	NOV: IG only_AWARENESS: Excelerate_Video_Ad (with Exclusion) â€" Copy	completed	campaign	1603743	2334	
24	Brand Awareness	NOV: IG only_AWARENESS: Shrishti_video_Ad (with Exclusion)	completed	campaign	1961895	2519	
25	Brand Awareness	OCT: FB only_AWARENESS: Excelerate_Video_Ad (with Exclusion) - Copy	completed	campaign	3130793	3708	
26	Brand Awareness	OCT: IG only_AWARENESS: Excelerate_Video_Ad (with Exclusion)	completed	campaign	1637784	2386	

2. `outbound_clicks` — Detect Outliers

```
SELECT
  PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY outbound_clicks) AS q1,
  PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY outbound_clicks) AS q3
FROM marketing_campaign_raw;
```



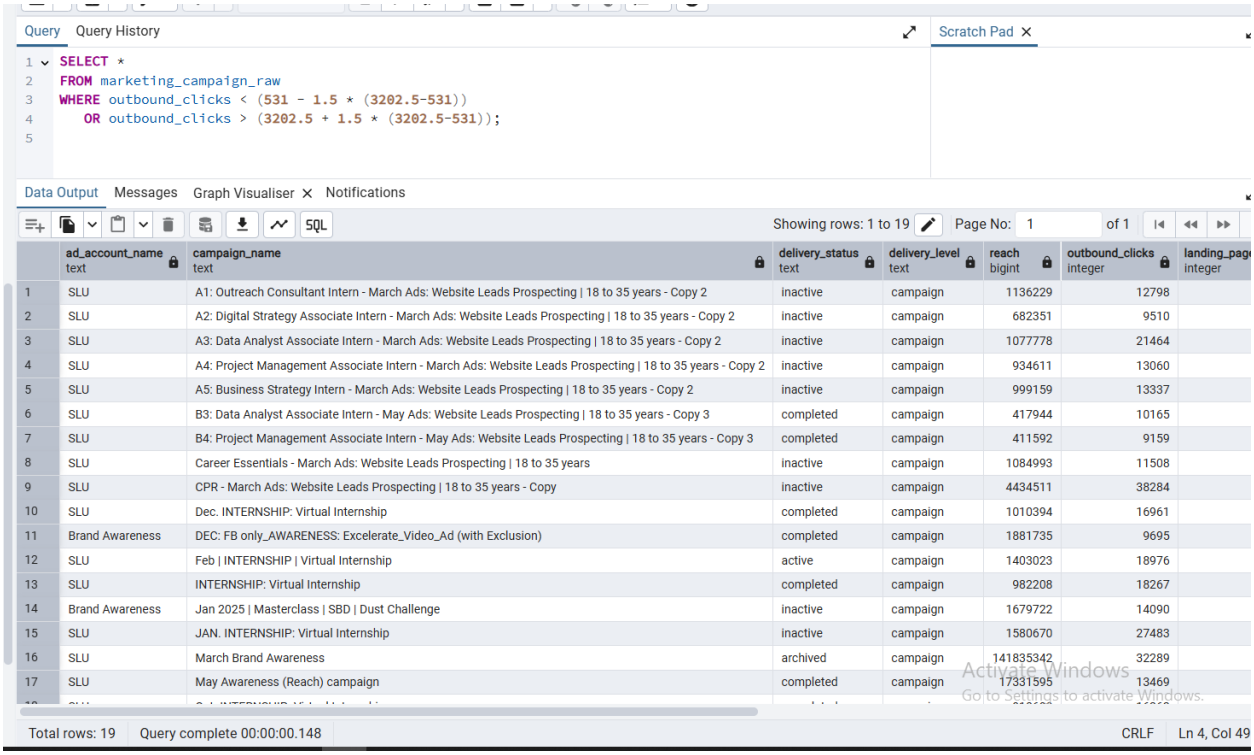
```
SELECT
  PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY outbound_clicks) AS q1,
  PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY outbound_clicks) AS q3
FROM marketing_campaign_raw;
```

Data Output | Messages | Graph Visualiser | Notifications — Showing rows: 1 to 1

	q1 double precision	q3 double precision
1	531	3202.5

Q1 = 531 , Q3 = 3202.5

```
SELECT *
FROM marketing_campaign_raw
WHERE outbound_clicks < (531 - 1.5 * (3202.5 - 531))
   OR outbound_clicks > (3202.5 + 1.5 * (3202.5 - 531));
```



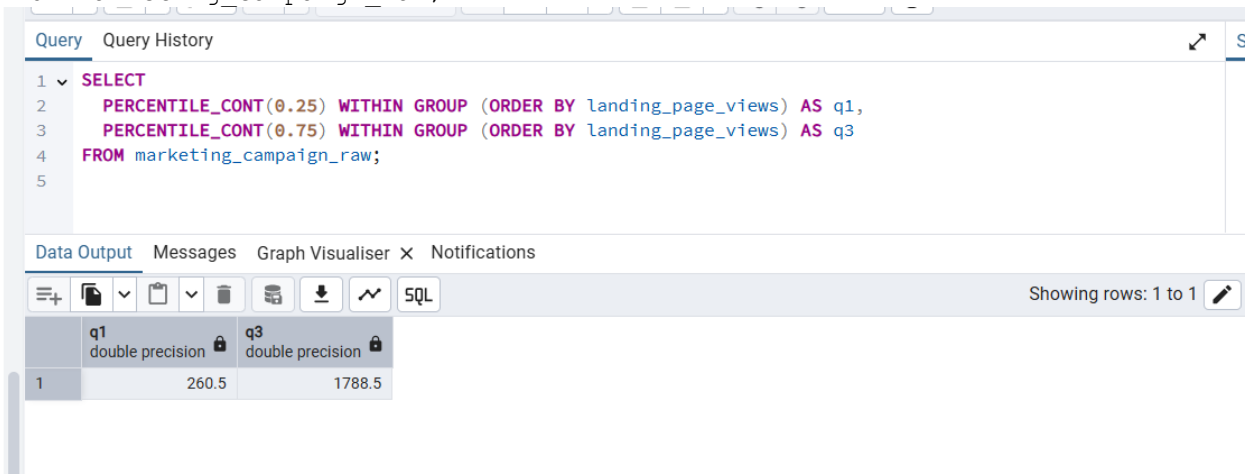
```
SELECT *
FROM marketing_campaign_raw
WHERE outbound_clicks < (531 - 1.5 * (3202.5-531))
   OR outbound_clicks > (3202.5 + 1.5 * (3202.5-531));
```

	ad_account_name text	campaign_name text	delivery_status text	delivery_level text	reach bigint	outbound_clicks integer	landing_pag integer
1	SLU	A1: Outreach Consultant Intern - March Ads: Website Leads Prospecting | 18 to 35 years - Copy 2	inactive	campaign	1136229	12798	
2	SLU	A2: Digital Strategy Associate Intern - March Ads: Website Leads Prospecting | 18 to 35 years - Copy 2	inactive	campaign	682351	9510	
3	SLU	A3: Data Analyst Associate Intern - March Ads: Website Leads Prospecting | 18 to 35 years - Copy 2	inactive	campaign	1077778	21464	
4	SLU	A4: Project Management Associate Intern - March Ads: Website Leads Prospecting | 18 to 35 years - Copy 2	inactive	campaign	934611	13060	
5	SLU	A5: Business Strategy Intern - March Ads: Website Leads Prospecting | 18 to 35 years - Copy 2	inactive	campaign	999159	13337	
6	SLU	B3: Data Analyst Associate Intern - May Ads: Website Leads Prospecting | 18 to 35 years - Copy 3	completed	campaign	417944	10165	
7	SLU	B4: Project Management Associate Intern - May Ads: Website Leads Prospecting | 18 to 35 years - Copy 3	completed	campaign	411592	9159	
8	SLU	Career Essentials - March Ads: Website Leads Prospecting | 18 to 35 years	inactive	campaign	1084993	11508	
9	SLU	CPR - March Ads: Website Leads Prospecting | 18 to 35 years - Copy	inactive	campaign	4434511	38284	
10	SLU	Dec. INTERNSHIP: Virtual Internship	completed	campaign	1010394	16961	
11	Brand Awareness	DEC: FB only_AWARENESS: Excelerate_Video_Ad (with Exclusion)	completed	campaign	1881735	9695	
12	SLU	Feb | INTERNSHIP | Virtual Internship	active	campaign	1403023	18976	
13	SLU	INTERNSHIP: Virtual Internship	completed	campaign	982208	18267	
14	Brand Awareness	Jan 2025 | Masterclass | SBD | Dust Challenge	inactive	campaign	1679722	14090	
15	SLU	JAN. INTERNSHIP: Virtual Internship	inactive	campaign	1580670	27483	
16	SLU	March Brand Awareness	archived	campaign	141835342	32289	
17	SLU	May Awareness (Reach) campaign	completed	campaign	17331595	13469	

Total rows: 19 | Query complete 00:00:00.148 | CRLF | Ln 4, Col 49

◆ 3. `landing_page_views` — Detect Outliers

```
SELECT
  PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY landing_page_views) AS q1,
  PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY landing_page_views) AS q3
FROM marketing_campaign_raw;
```

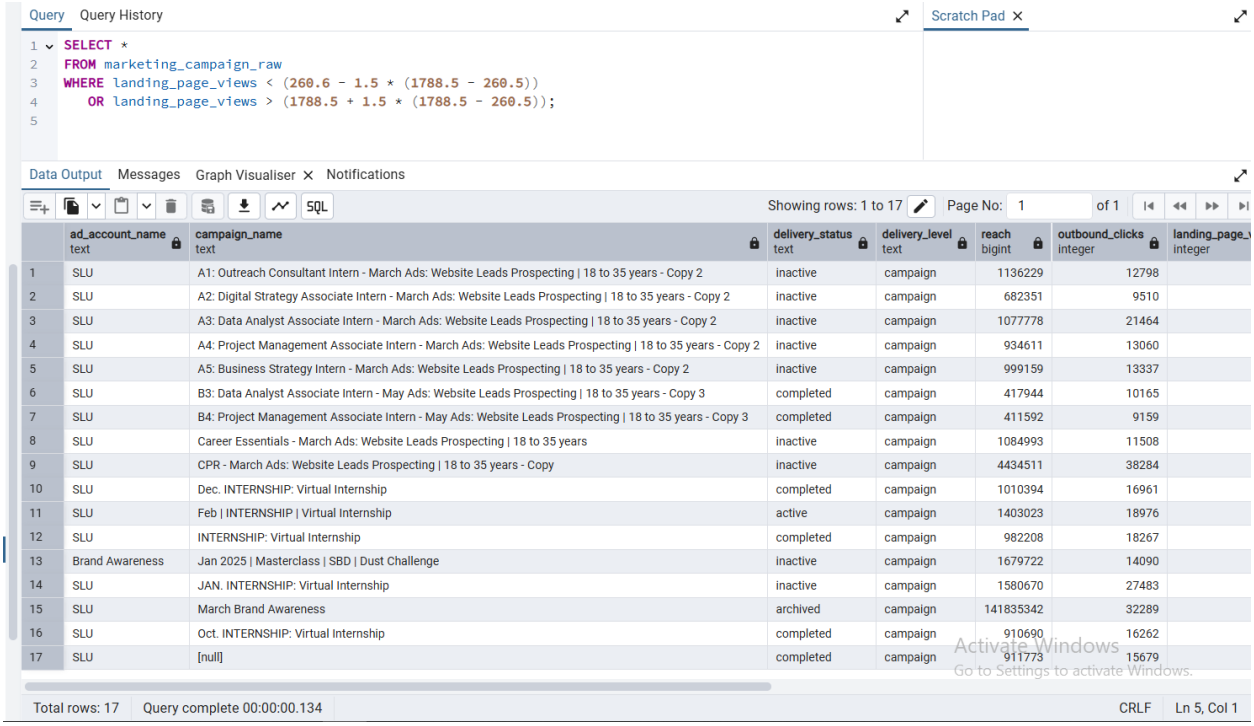


```
SELECT
  PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY landing_page_views) AS q1,
  PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY landing_page_views) AS q3
FROM marketing_campaign_raw;
```

	q1 double precision	q3 double precision
1	260.5	1788.5

Q1 = 260.5 , Q3 = 1788.5

```
SELECT *
FROM marketing_campaign_raw
WHERE landing_page_views < (260.6 - 1.5 * (1788.5 - 260.5))
   OR landing_page_views > (1788.5 + 1.5 * (1788.5 - 260.5));
```



Query | Query History | Scratch Pad

```
SELECT *
FROM marketing_campaign_raw
WHERE landing_page_views < (260.6 - 1.5 * (1788.5 - 260.5))
   OR landing_page_views > (1788.5 + 1.5 * (1788.5 - 260.5));
```

Data Output | Messages | Graph Visualiser | Notifications

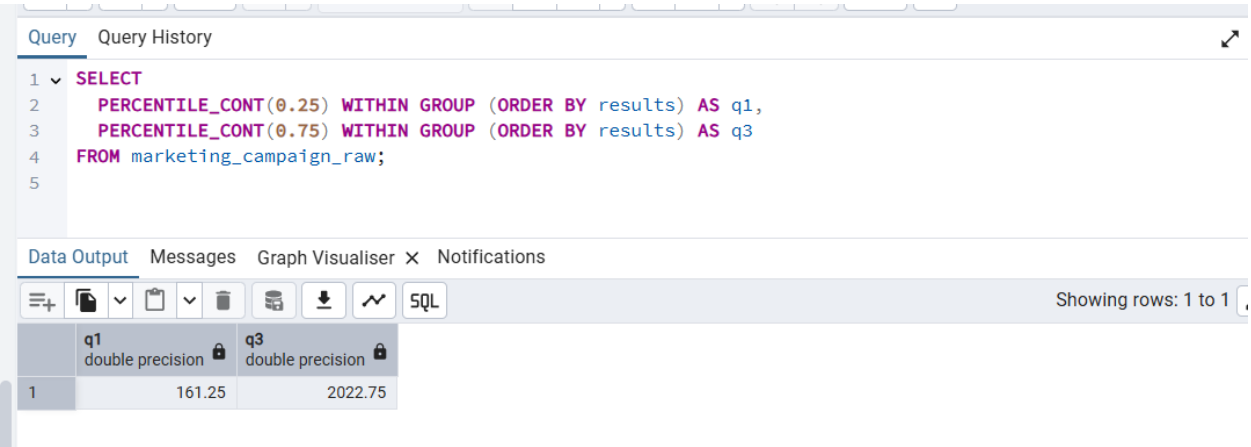
Showing rows: 1 to 17 | Page No: 1 of 1

	ad_account_name text	campaign_name text	delivery_status text	delivery_level text	reach bigint	outbound_clicks integer	landing_page_v integer
1	SLU	A1: Outreach Consultant Intern - March Ads: Website Leads Prospecting | 18 to 35 years - Copy 2	inactive	campaign	1136229	12798	
2	SLU	A2: Digital Strategy Associate Intern - March Ads: Website Leads Prospecting | 18 to 35 years - Copy 2	inactive	campaign	682351	9510	
3	SLU	A3: Data Analyst Associate Intern - March Ads: Website Leads Prospecting | 18 to 35 years - Copy 2	inactive	campaign	1077778	21464	
4	SLU	A4: Project Management Associate Intern - March Ads: Website Leads Prospecting | 18 to 35 years - Copy 2	inactive	campaign	934611	13060	
5	SLU	A5: Business Strategy Intern - March Ads: Website Leads Prospecting | 18 to 35 years - Copy 2	inactive	campaign	999159	13337	
6	SLU	B3: Data Analyst Associate Intern - May Ads: Website Leads Prospecting | 18 to 35 years - Copy 3	completed	campaign	417944	10165	
7	SLU	B4: Project Management Associate Intern - May Ads: Website Leads Prospecting | 18 to 35 years - Copy 3	completed	campaign	411592	9159	
8	SLU	Career Essentials - March Ads: Website Leads Prospecting | 18 to 35 years	inactive	campaign	1084993	11508	
9	SLU	CPR - March Ads: Website Leads Prospecting | 18 to 35 years - Copy	inactive	campaign	4434511	38284	
10	SLU	Dec. INTERNSHIP: Virtual Internship	completed	campaign	1010394	16961	
11	SLU	Feb | INTERNSHIP | Virtual Internship	active	campaign	1403023	18976	
12	SLU	INTERNSHIP: Virtual Internship	completed	campaign	982208	18267	
13	Brand Awareness	Jan 2025 | Masterclass | SBD | Dust Challenge	inactive	campaign	1679722	14090	
14	SLU	JAN. INTERNSHIP: Virtual Internship	inactive	campaign	1580670	27483	
15	SLU	March Brand Awareness	archived	campaign	141835342	32289	
16	SLU	Oct. INTERNSHIP: Virtual Internship	completed	campaign	910690	16262	
17	SLU	[null]	completed	campaign	911773	15679	

Total rows: 17 | Query complete 00:00:00.134 | CRLF | Ln 5, Col 1

◆ 4. `results` — Detect Outliers

```
SELECT
  PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY results) AS q1,
  PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY results) AS q3
FROM marketing_campaign_raw;
```



Query | Query History

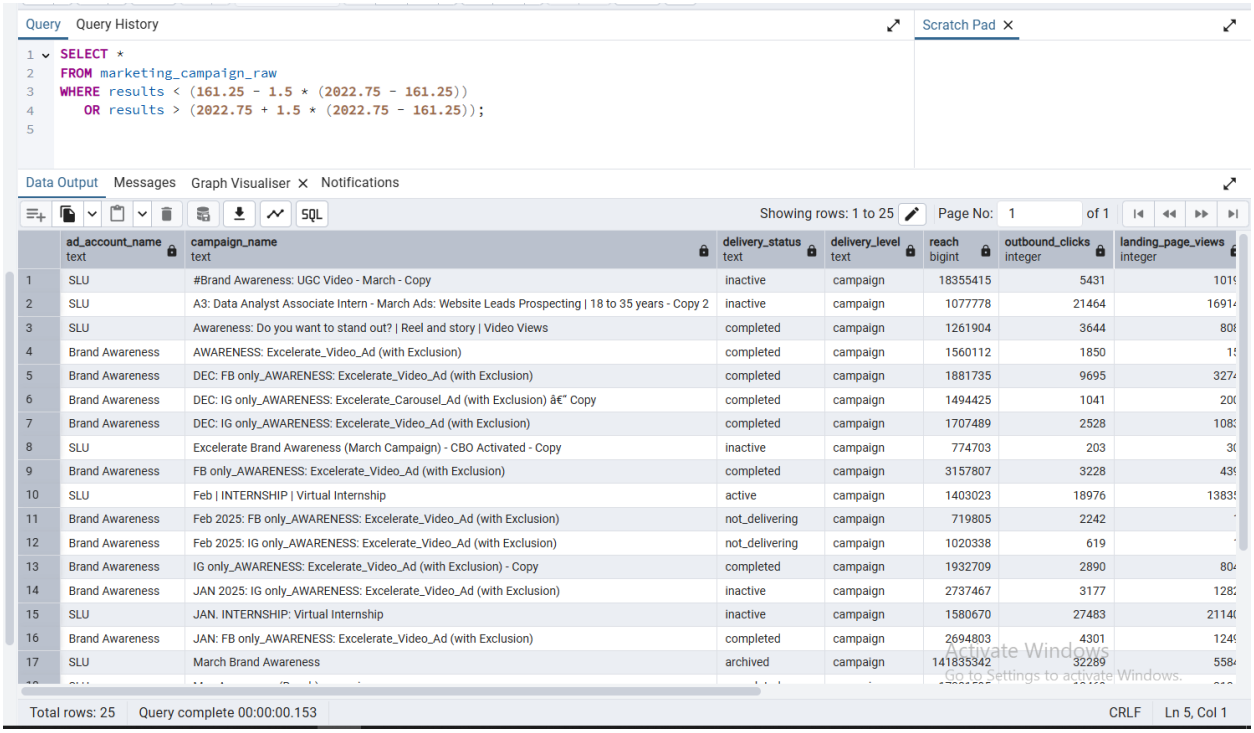
```
SELECT
  PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY results) AS q1,
  PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY results) AS q3
FROM marketing_campaign_raw;
```

Data Output | Messages | Graph Visualiser | Notifications

Showing rows: 1 to 1

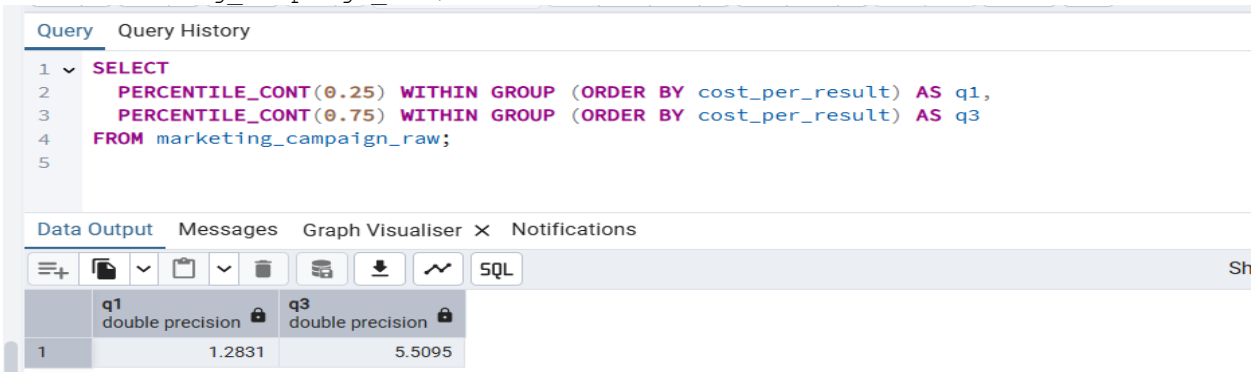
	q1 double precision	q3 double precision
1	161.25	2022.75

```
Q1 = 161.25 , Q2 = 2022.75
SELECT *

FROM marketing_campaign_raw

WHERE results < (161.25 - 1.5 * (2022.75 - 161.25))

   OR results > (2022.75 + 1.5 * (2022.75 - 161.25));
```



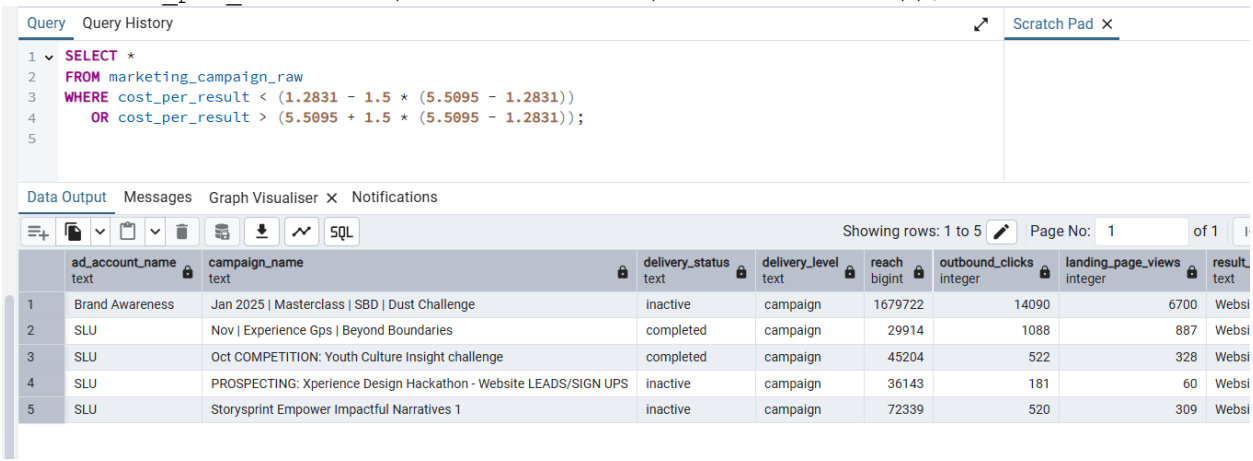
◆ 5. `cost_per_result` — Detect Outliers

```
SELECT
  PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY cost_per_result) AS q1,
  PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY cost_per_result) AS q3
FROM marketing_campaign_raw;
```



Q1 = 1.2831 , Q3 = 5.5095

```
SELECT *
FROM marketing_campaign_raw
WHERE cost_per_result < (1.2831 - 1.5 * (5.5095 - 1.2831))
   OR cost_per_result > (5.5095 + 1.5 * (5.5095 - 1.2831));
```

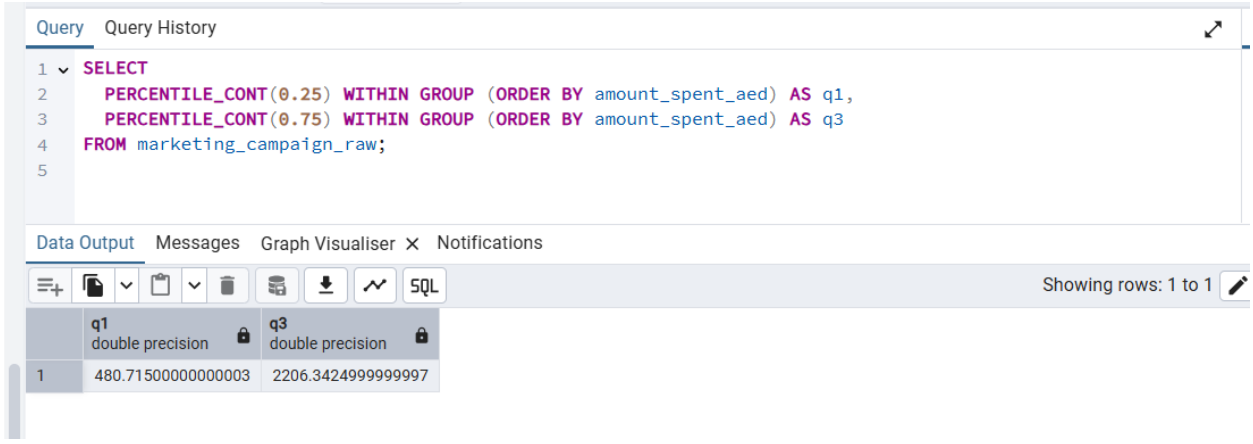


	ad_account_name text	campaign_name text	delivery_status text	delivery_level text	reach bigint	outbound_clicks integer	landing_page_views integer	result_ text
1	Brand Awareness	Jan 2025 | Masterclass | SBD | Dust Challenge	inactive	campaign	1679722	14090	6700	Websi
2	SLU	Nov | Experience Gps | Beyond Boundaries	completed	campaign	29914	1088	887	Websi
3	SLU	Oct COMPETITION: Youth Culture Insight challenge	completed	campaign	45204	522	328	Websi
4	SLU	PROSPECTING: Xperience Design Hackathon - Website LEADS/SIGN UPS	inactive	campaign	36143	181	60	Websi
5	SLU	Storysprint Empower Impactful Narratives 1	inactive	campaign	72339	520	309	Websi

---

◆ 6. `amount_spent_aed` — Detect Outliers

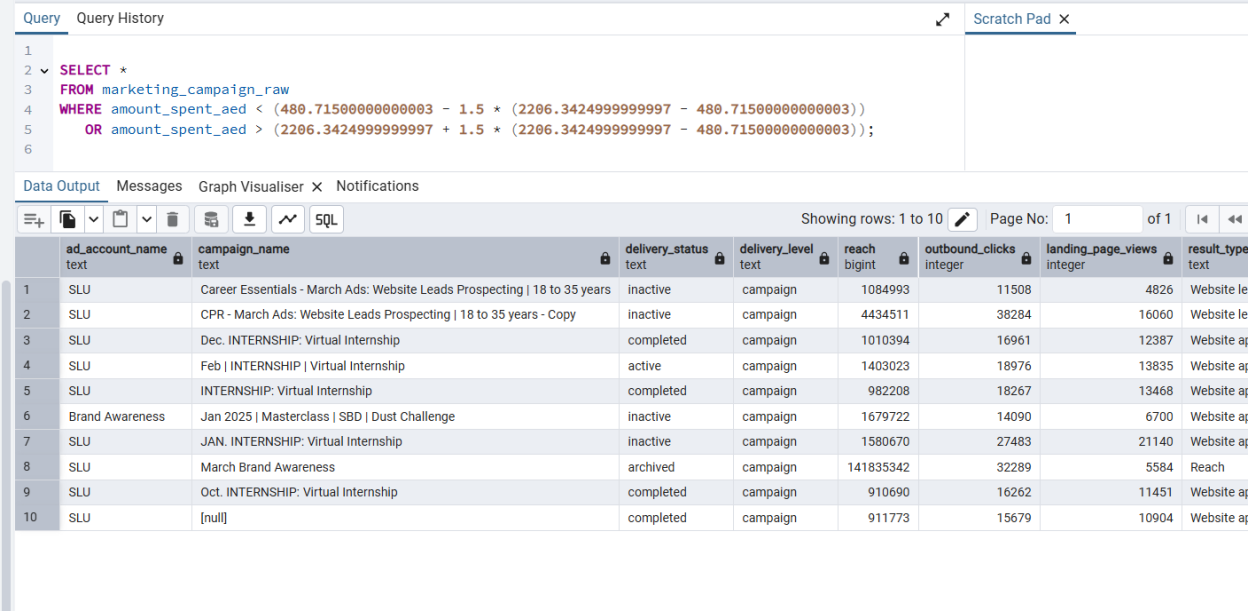
```
SELECT
  PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY amount_spent_aed) AS q1,
  PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY amount_spent_aed) AS q3
FROM marketing_campaign_raw;
```



	q1 double precision	q3 double precision
1	480.71500000000003	2206.3424999999997

Q1 = 480.71500000000003, Q3 = 2206.3424999999997

```
SELECT *
FROM marketing_campaign_raw
WHERE amount_spent_aed < (480.71500000000003 - 1.5 * (2206.3424999999997 -
480.71500000000003))
   OR amount_spent_aed > (2206.3424999999997 + 1.5 * (2206.3424999999997 -
480.71500000000003));
```



```
SELECT *
FROM marketing_campaign_raw
WHERE amount_spent_aed < (480.71500000000003 - 1.5 * (2206.3424999999997 - 480.71500000000003))
   OR amount_spent_aed > (2206.3424999999997 + 1.5 * (2206.3424999999997 - 480.71500000000003));
```

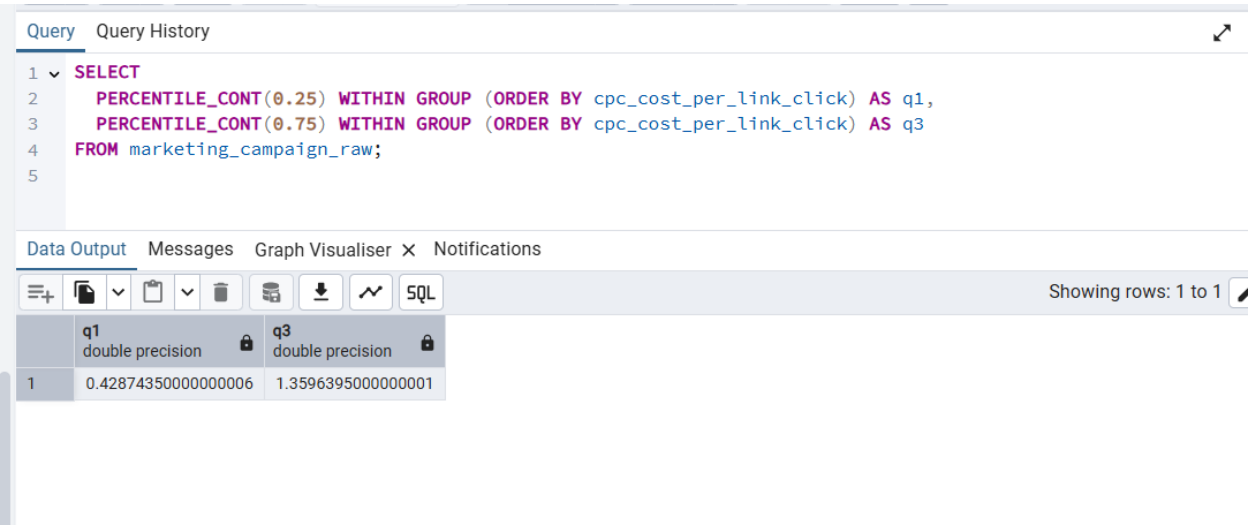
Showing rows: 1 to 10 Page No: 1 of 1

	ad_account_name text	campaign_name text	delivery_status text	delivery_level text	reach bigint	outbound_clicks integer	landing_page_views integer	result_type text
1	SLU	Career Essentials - March Ads: Website Leads Prospecting | 18 to 35 years	inactive	campaign	1084993	11508	4826	Website le
2	SLU	CPR - March Ads: Website Leads Prospecting | 18 to 35 years - Copy	inactive	campaign	4434511	38284	16060	Website le
3	SLU	Dec. INTERNSHIP: Virtual Internship	completed	campaign	1010394	16961	12387	Website ap
4	SLU	Feb | INTERNSHIP | Virtual Internship	active	campaign	1403023	18976	13835	Website ap
5	SLU	INTERNSHIP: Virtual Internship	completed	campaign	982208	18267	13468	Website ap
6	Brand Awareness	Jan 2025 | Masterclass | SBD | Dust Challenge	inactive	campaign	1679722	14090	6700	Website ap
7	SLU	JAN. INTERNSHIP: Virtual Internship	inactive	campaign	1580670	27483	21140	Website ap
8	SLU	March Brand Awareness	archived	campaign	141835342	32289	5584	Reach
9	SLU	Oct. INTERNSHIP: Virtual Internship	completed	campaign	910690	16262	11451	Website ap
10	SLU	[null]	completed	campaign	911773	15679	10904	Website ap

---

◆ 7. `cpc_cost_per_link_click` — Detect Outliers

```
SELECT
  PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY cpc_cost_per_link_click) AS
q1,
  PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY cpc_cost_per_link_click) AS q3
FROM marketing_campaign_raw;
```



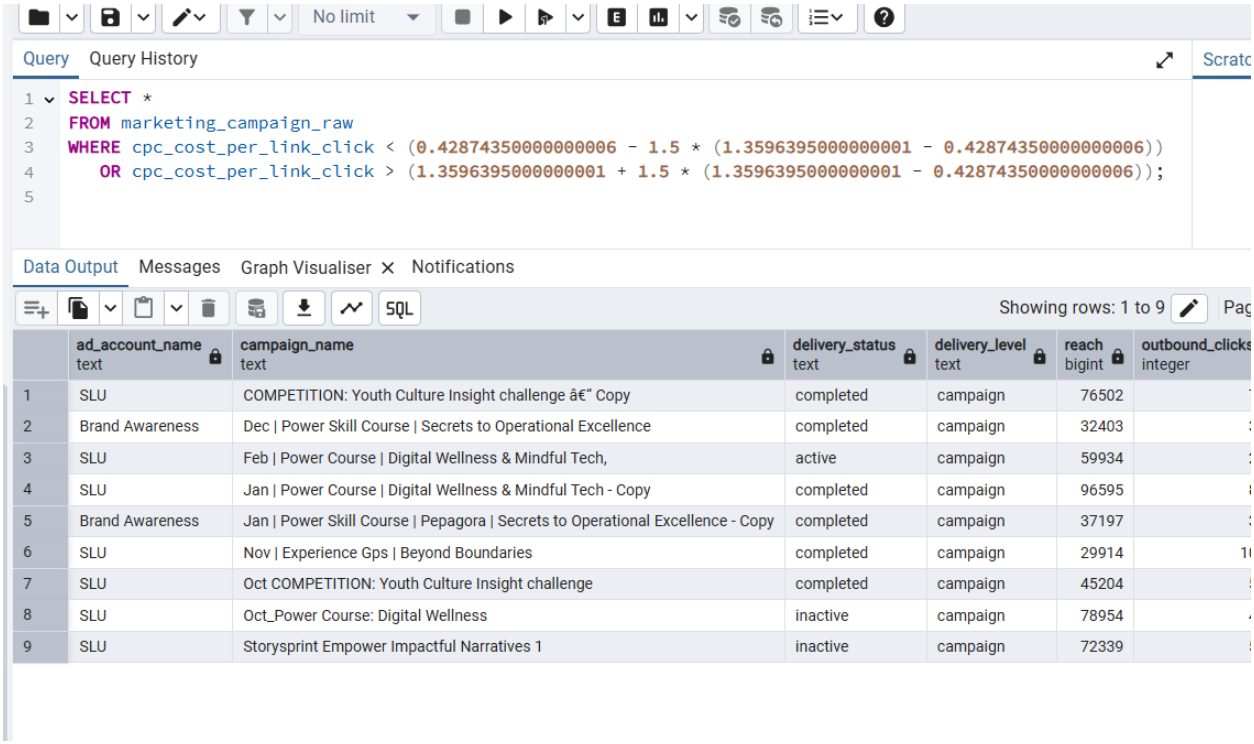
```
SELECT
  PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY cpc_cost_per_link_click) AS q1,
  PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY cpc_cost_per_link_click) AS q3
FROM marketing_campaign_raw;
```

Showing rows: 1 to 1

	q1 double precision	q3 double precision
1	0.42874350000000006	1.3596395000000001

```
Q1 = 0.42874350000000006 , Q3 = 1.3596395000000001

SELECT *
FROM marketing_campaign_raw
WHERE cpc_cost_per_link_click < (0.42874350000000006 - 1.5 *
(1.3596395000000001 - 0.42874350000000006))
```

```
   OR cpc_cost_per_link_click > (1.3596395000000001 + 1.5 *
(1.3596395000000001 - 0.42874350000000006));
```



Query Query History

```
1  SELECT *
2  FROM marketing_campaign_raw
3  WHERE cpc_cost_per_link_click < (0.42874350000000006 - 1.5 * (1.3596395000000001 - 0.42874350000000006))
4     OR cpc_cost_per_link_click > (1.3596395000000001 + 1.5 * (1.3596395000000001 - 0.42874350000000006));
5
```

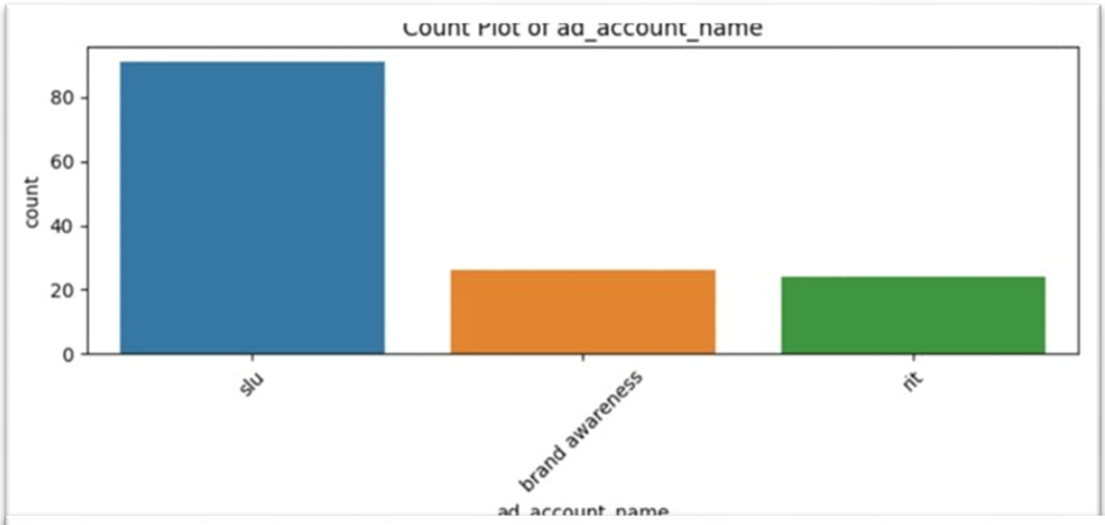
Data Output | Messages | Graph Visualiser | Notifications

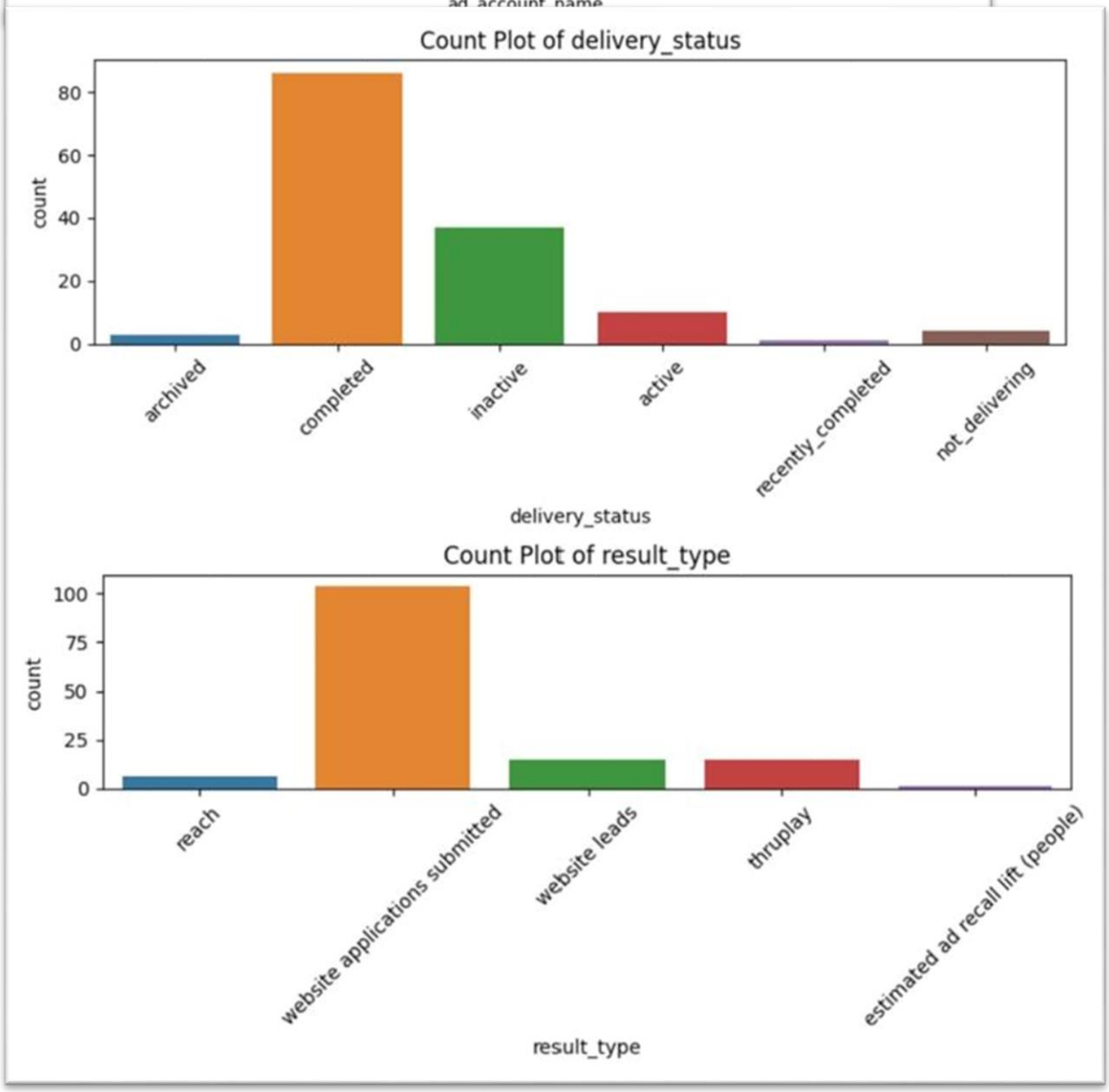
Showing rows: 1 to 9

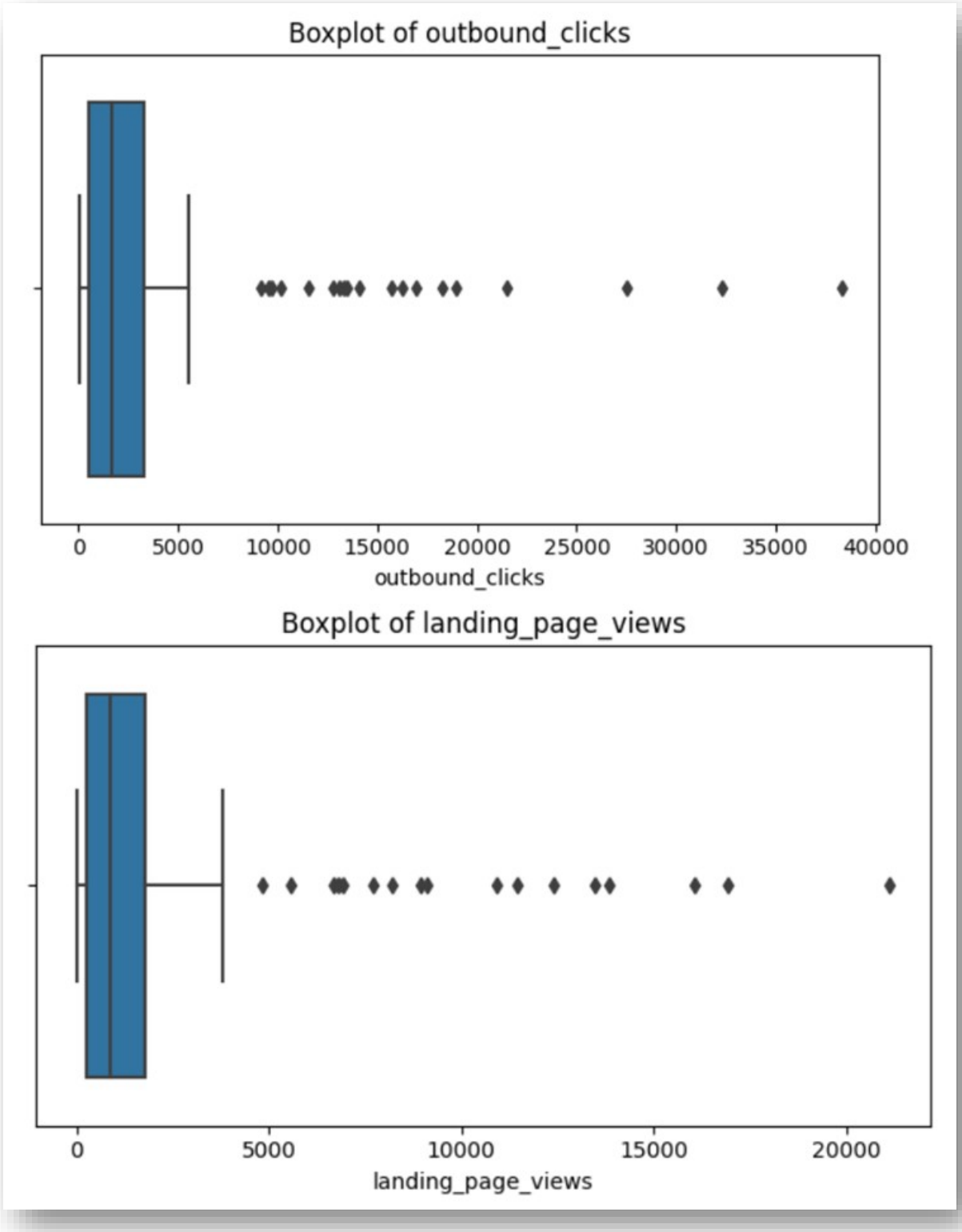
	ad_account_name text	campaign_name text	delivery_status text	delivery_level text	reach bigint	outbound_clicks integer
1	SLU	COMPETITION: Youth Culture Insight challenge â€" Copy	completed	campaign	76502	
2	Brand Awareness	Dec | Power Skill Course | Secrets to Operational Excellence	completed	campaign	32403	
3	SLU	Feb | Power Course | Digital Wellness & Mindful Tech,	active	campaign	59934	
4	SLU	Jan | Power Course | Digital Wellness & Mindful Tech - Copy	completed	campaign	96595	
5	Brand Awareness	Jan | Power Skill Course | Pepagora | Secrets to Operational Excellence - Copy	completed	campaign	37197	
6	SLU	Nov | Experience Gps | Beyond Boundaries	completed	campaign	29914	
7	SLU	Oct COMPETITION: Youth Culture Insight challenge	completed	campaign	45204	
8	SLU	Oct_Power Course: Digital Wellness	inactive	campaign	78954	
9	SLU	Storysprint Empower Impactful Narratives 1	inactive	campaign	72339	

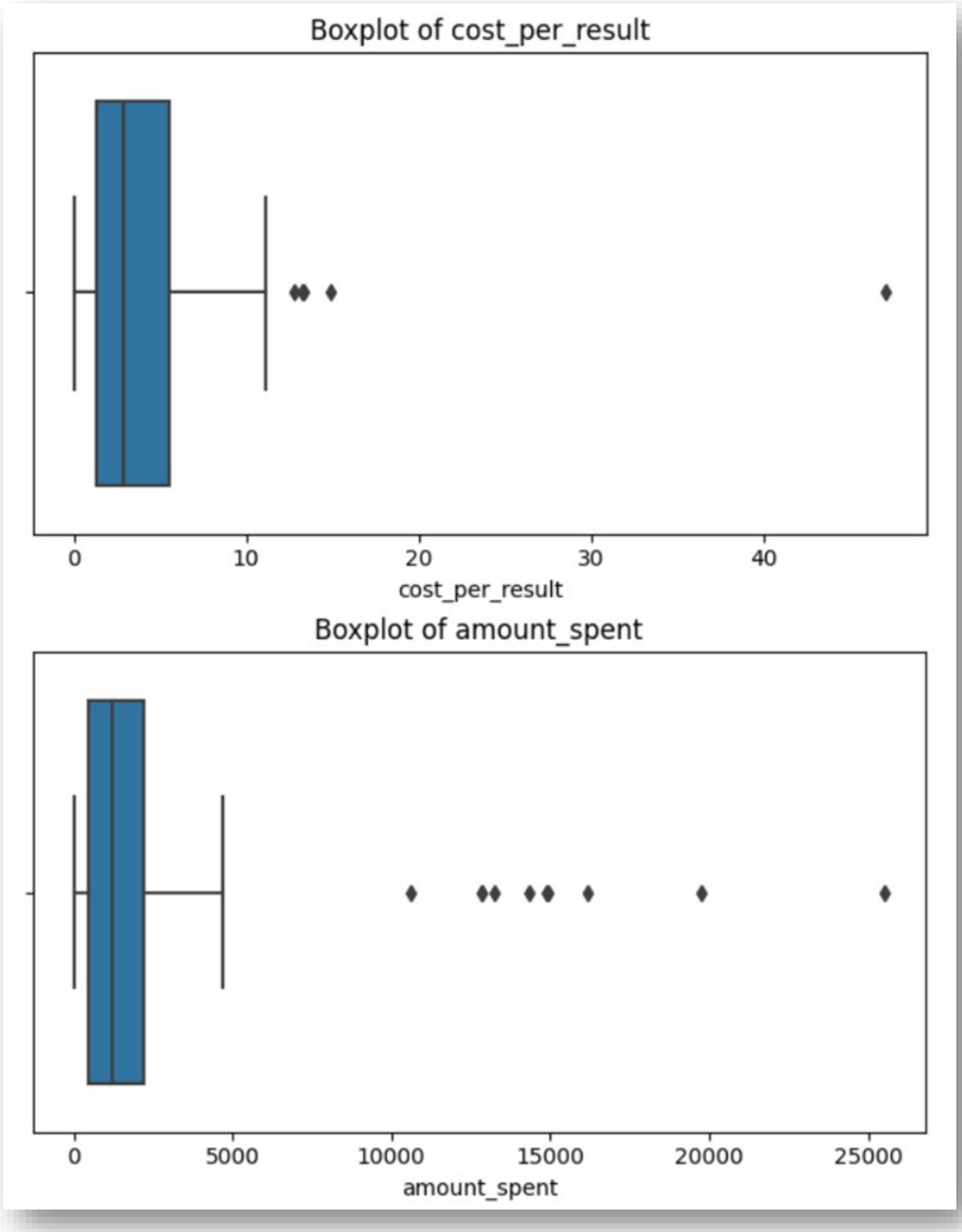
Step 5: Visualize Data Trends

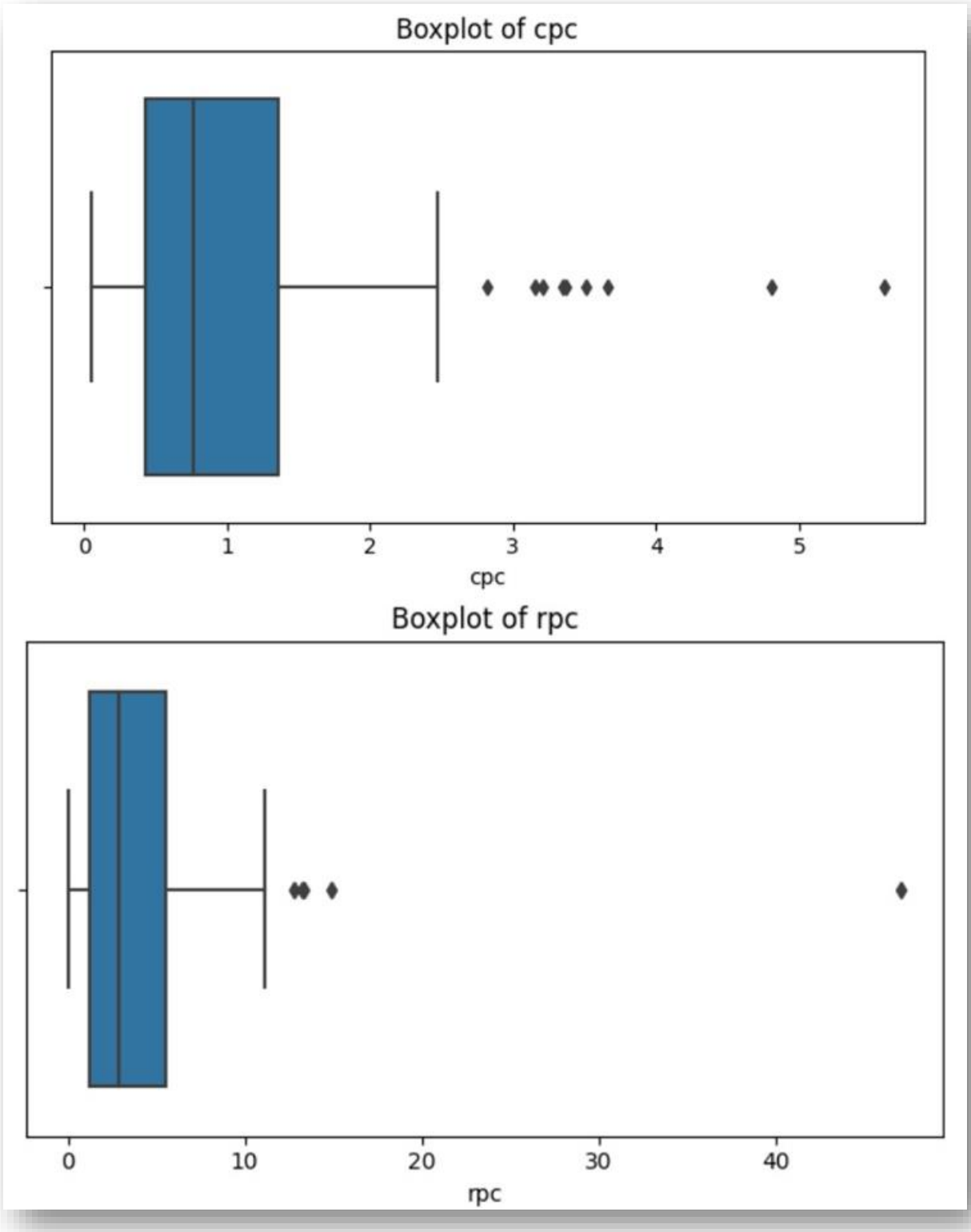
- Use graphs like histograms, line charts, and box plots to identify patterns.
- Analyze relationships between different variables.


Count Plot of ad_account_name
count
0
20
40
60
80
slu
brand awareness
nit
ad_account_name

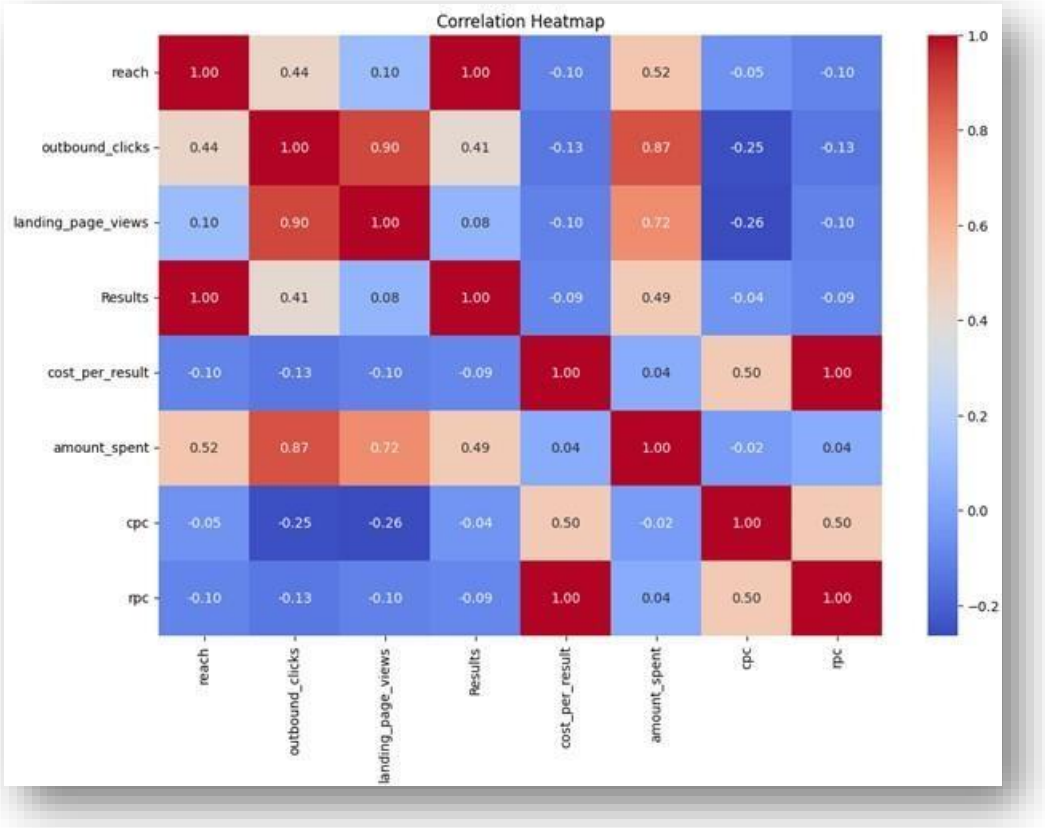

Count Plot of delivery_status
count
0
20
40
60
80
archived
completed
inactive
active
recently_completed
not_delivering
delivery_status
Count Plot of result_type
count
0
25
50
75
100
reach
website applications submitted
website leads
thruplay
estimated ad recall lift (people)
result_type


Boxplot of outbound_clicks
0
5000
10000
15000
20000
25000
30000
35000
40000
outbound_clicks
Boxplot of landing_page_views
0
5000
10000
15000
20000
landing_page_views


Boxplot of cost_per_result
0
10
20
30
40
cost_per_result
Boxplot of amount_spent
0
5000
10000
15000
20000
25000
amount_spent


Boxplot of cpc
0
1
2
3
4
5
cpc
Boxplot of rpc
0
10
20
30
40
rpc

❖ Heat Map


Correlation Heatmap
reach 1.00 0.44 0.10 1.00 -0.10 0.52 -0.05 -0.10
outbound_clicks 0.44 1.00 0.90 0.41 -0.13 0.87 -0.25 -0.13
landing_page_views 0.10 0.90 1.00 0.08 -0.10 0.72 -0.26 -0.10
Results 1.00 0.41 0.08 1.00 -0.09 0.49 -0.04 -0.09
cost_per_result -0.10 -0.13 -0.10 -0.09 1.00 0.04 0.50 1.00
amount_spent 0.52 0.87 0.72 0.49 0.04 1.00 -0.02 0.04
cpc -0.05 -0.25 -0.26 -0.04 0.50 -0.02 1.00 0.50
rpc -0.10 -0.13 -0.10 -0.09 1.00 0.04 0.50 1.00
reach
outbound_clicks
landing_page_views
Results
cost_per_result
amount_spent
cpc
rpc
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

Dataset: Cognito_Raw

Performing the Exploratory Data Analysis using Python

Step 1: Understand the Dataset

- View the first few rows to get an overview of the data.
- Check column names, data types, and the total number of records.

Research how to inspect a table's structure and data in Visual Studio Code using Python.

Pandas is a package in Python used to read data, manipulation and analysis.

Python Codes Used:

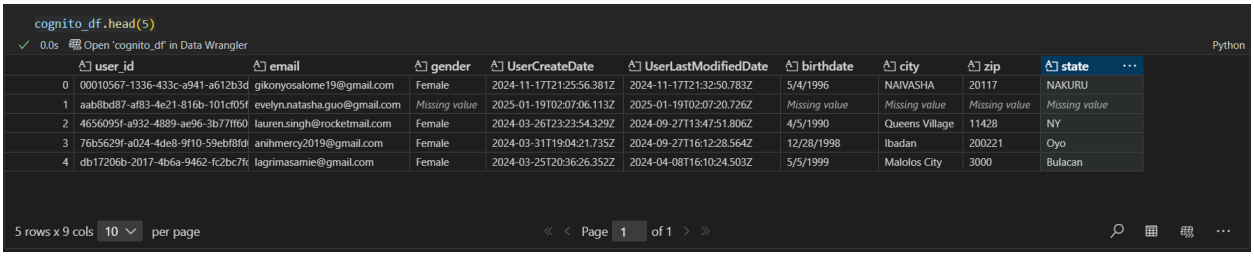
1.Reading the CSV File:

```
cognito_df = pd.read_csv(r"E:\ECCELERATE\Data\Cognito_Raw2(in).csv")
```

2.View First Few Rows:

```
cognito_df.head(5)
```

🔍 *This gives you a quick peek at the dataset — ideal for spotting structure or obvious issues.*



```
cognito_df.head(5)
```

✓ 0.0s 📓 Open 'cognito_df' in Data Wrangler — Python

	user_id	email	gender	UserCreateDate	UserLastModifiedDate	birthdate	city	zip	state
0	00010567-1336-433c-a941-a612b3d	gikonyosalome19@gmail.com	Female	2024-11-17T21:25:56.381Z	2024-11-17T21:32:50.783Z	5/4/1996	NAIVASHA	20117	NAKURU
1	aab8bd87-af83-4e21-816b-101cf05f	evelyn.natasha.guo@gmail.com	*Missing value*	2025-01-19T02:07:06.113Z	2025-01-19T02:07:20.726Z	*Missing value*	*Missing value*	*Missing value*	*Missing value*
2	4656095f-a932-4889-ae96-3b77ff60	lauren.singh@rocketmail.com	Female	2024-03-26T23:23:54.329Z	2024-09-27T13:47:51.806Z	4/5/1990	Queens Village	11428	NY
3	76b5629f-a024-4de8-9f10-59ebf8fd	anihmercy2019@gmail.com	Female	2024-03-31T19:04:21.735Z	2024-09-27T16:12:28.564Z	12/28/1998	Ibadan	200221	Oyo
4	db17206b-2017-4b6a-9462-fc2bc7f	lagrimasamie@gmail.com	Female	2024-03-25T20:36:26.352Z	2024-04-08T16:10:24.503Z	5/5/1999	Malolos City	3000	Bulacan

5 rows x 9 cols 10 per page « ‹ Page 1 of 1 › »

---

3. Check Column Names & Data Types:

```
cognito_df.info()
```

This tells you what each column is, and whether it's stored as text, numeric, timestamp, etc.

```
cognito_df.info()
✓ 0.0s
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 129178 entries, 0 to 129177
Data columns (total 9 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   user_id               129178 non-null  object
 1   email                 129178 non-null  object
 2   gender                86316 non-null   object
 3   UserCreateDate        129178 non-null  object
 4   UserLastModifiedDate  129178 non-null  object
 5   birthdate             86316 non-null   object
 6   city                  86305 non-null   object
 7   zip                   86251 non-null   object
 8   state                 86154 non-null   object
dtypes: object(9)
memory usage: 8.9+ MB
```

Here in the image we can see some of the date columns are in object datatype, so we have to change its datatype into DATETIME

Datatype is changed and the dataset is stored in the new variable final_cognito_df

```
final_congnito_df["UserCreateDate"] = pd.to_datetime(final_congnito_df["UserCreateDate"])
final_congnito_df["UserLastModifiedDate"] = pd.to_datetime(final_congnito_df["UserLastModifiedDate"])
final_congnito_df["birthdate"] = pd.to_datetime(final_congnito_df["birthdate"])
✓ 0.5s
```

After Changing the Datatype:

```
final_congnito_df.info()
✓ 0.0s
<class 'pandas.core.frame.DataFrame'>
Index: 86096 entries, 0 to 129174
Data columns (total 9 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   user_id               86096 non-null  object
 1   email                 86096 non-null  object
 2   gender                86096 non-null  object
 3   UserCreateDate        86096 non-null  datetime64[ns, UTC]
 4   UserLastModifiedDate  86096 non-null  datetime64[ns, UTC]
 5   birthdate             86096 non-null  datetime64[ns]
 6   city                  86096 non-null  object
 7   zip                   86096 non-null  object
 8   state                 86096 non-null  object
dtypes: datetime64[ns, UTC](2), datetime64[ns](1), object(6)
memory usage: 6.6+ MB
```

---

4. Count Total Rows & Columns:

`cognito_df.shape`

```
cognito_df.shape
✓ 0.0s
(129178, 9)
```

Step 2: Summarize the Data

- Generate basic statistics like mean, median, min, max, and standard deviation to understand the distribution.

Explore how to compute summary statistics in Python.

Since the `CognitoRaw(in).csv` dataset contains no numeric field column to which we can do basic statistics ,we are converting the birthdate column to age and we'll compute :

- Minimum
- Maximum
- Mean (average)
- Median
- Standard deviation
- Count (just for completeness)

Python Code for Summary Statistics (VSCode):

Converting Birthdate into Age

```
#Converting 'birthdate' column to age
cognito_df['birthdate'] = pd.to_datetime(cognito_df['birthdate'], errors='coerce')
today = pd.to_datetime("today")
cognito_df['age'] = (today - cognito_df['birthdate']).dt.days // 365
```

1. Basic Descriptive Stats:

```
#Drop rows with missing age

df_age = cognito_df.dropna(subset=['age'])

#Describe the 'age' column

print(df_age['age'].describe())
```

```
#Describe the 'age' column
df_age['age'].describe()
```

✓ 0.3s

	# age
count	86316.0
mean	25.649833171138607
std	5.280843731899279
min	3.0
25%	22.0
50%	25.0
75%	28.0
max	101.0

8 rows x 1 cols 10 per page

What These Codes Will Show:

Statistic	Description
`MIN(Age)`	Smallest cohort size
MAX(Age)	Largest cohort size
AVG(Age)	Average cohort size
`STDDEV(Age)`	Standard deviation — shows how spread out the sizes are
`COUNT(Age)`	Total number of cohort records
`MEDIAN(Age)`	Middle value of all sizes (better indicator than average if there are outliers)

Step 3: Identify Missing and Duplicate Values

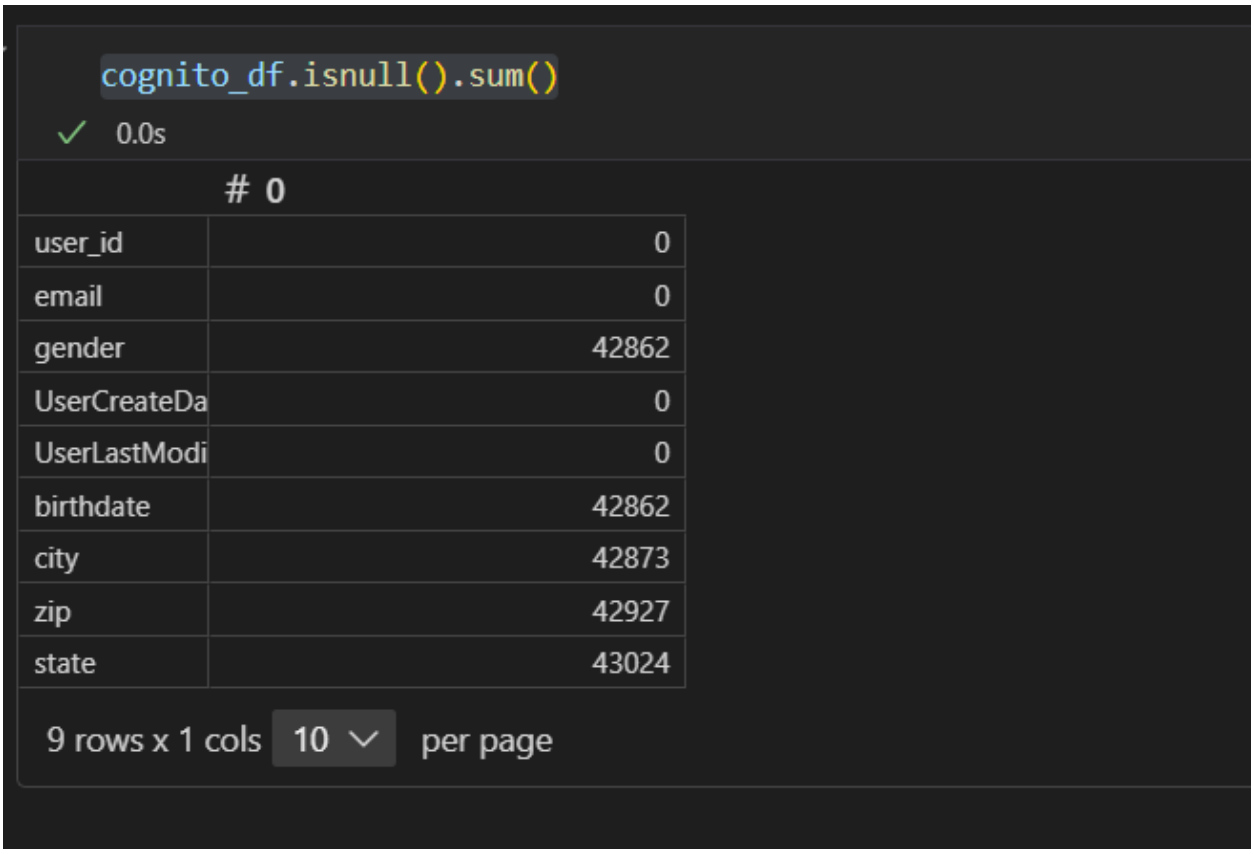
- Detect missing data to determine if any cleaning is needed.
- Identify duplicate records that may need removal.

Look up methods to check for missing values and duplicates in VS-Code.

Part 1: Detect Missing (NULL) Values

1. Missing Count for Each Column

```
cognito_df.isnull().sum()
```



```
cognito_df.isnull().sum()
```

✓ 0.0s

	# 0
user_id	0
email	0
gender	42862
UserCreateDa	0
UserLastModi	0
birthdate	42862
city	42873
zip	42927
state	43024

9 rows x 1 cols 10 per page

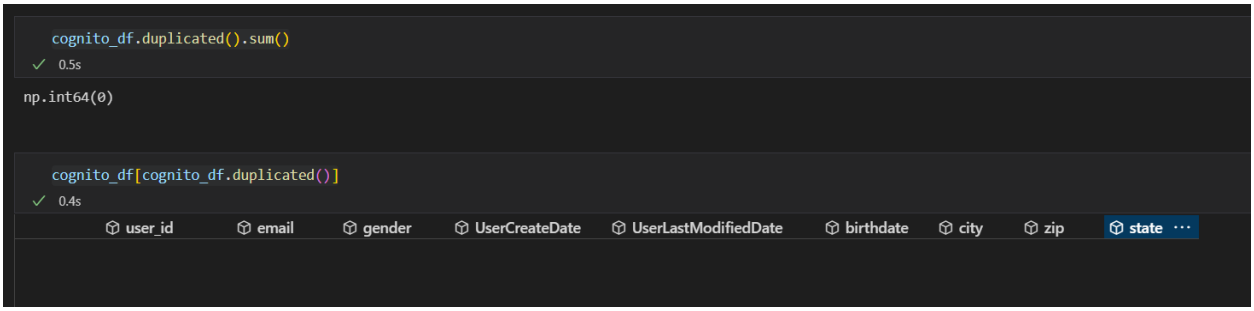
Part 2: Detect Duplicate Records

2. Fully Duplicate Rows (every column repeated)

```
cognito_df.duplicated().sum()
```

```
cognito_df[cognito_df.duplicated()]
```

- **No duplications in this.**



What to Look For:

Feature	Insight
NULL Values	Indicate incomplete or missing data that may need cleaning
Fully Duplicate Rows	Often accidental — likely safe to remove

Step 4: Spot Outliers and Anomalies

- Use techniques like the Interquartile Range (IQR) to find unusual values that might distort analysis.

Research how outliers are detected using SQL and statistical methods.

Interquartile Range (IQR) Method Overview

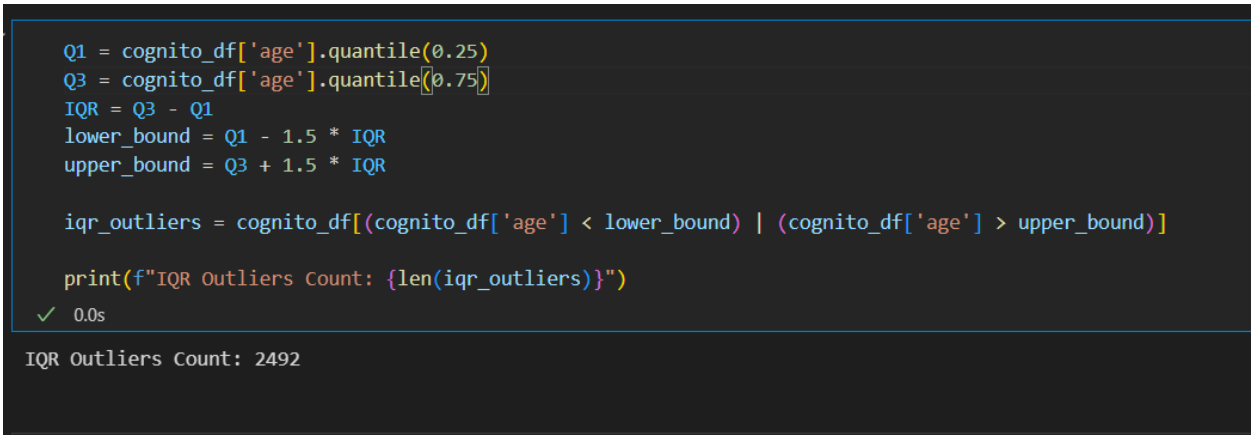
Steps:

5. Calculate Q1 (25th percentile) and Q3 (75th percentile)
6. Compute IQR = Q3 − Q1
7. Define outlier bounds:
    - Lower Bound = Q1 − 1.5 × IQR
    - Upper Bound = Q3 + 1.5 × IQR
8. Flag values outside these bounds as outliers

PostgreSQL Queries to Detect Outliers

1. Get Q1, Q3, and IQR

```
Q1 = cognito_df['age'].quantile(0.25)

Q3 = cognito_df['age'].quantile(0.75)

IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR

upper_bound = Q3 + 1.5 * IQR

iqr_outliers = cognito_df[(cognito_df['age'] < lower_bound) |
(cognito_df['age'] > upper_bound)]
print(f"IQR Outliers Count: {len(iqr_outliers)}")
```



What to Look For:

Metric	Purpose
Q1 & Q3	Measure the range where most data lies
Outliers	Any value falling far outside this range
Next Step	Review outliers — correct, cap, or remove as needed

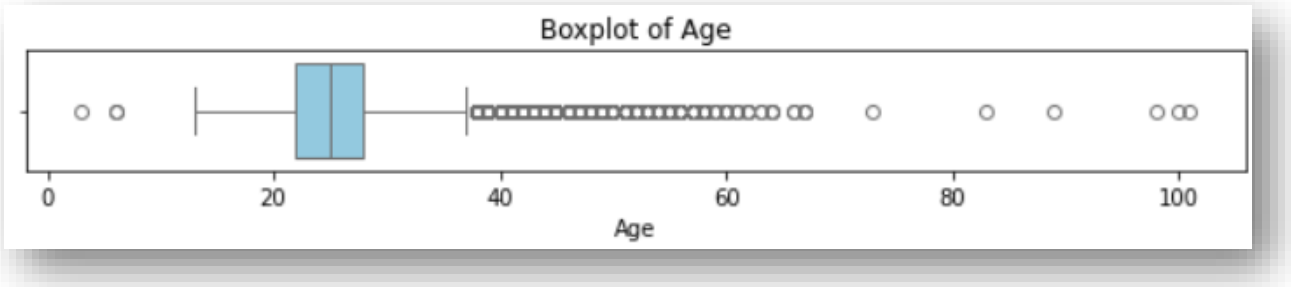
---

Step 5: Visualize Data Trends

- Use graphs like histograms, line charts, and box plots to identify patterns.
- Analyze relationships between different variables.

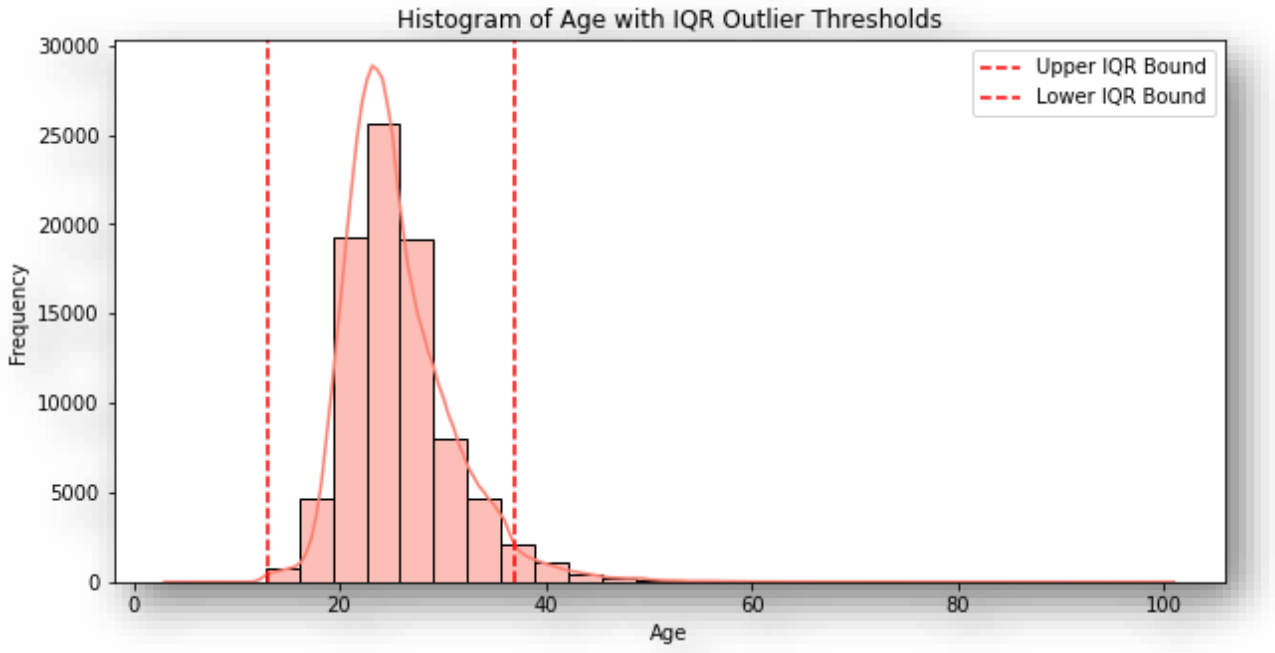
Find out how to create basic visualizations for data analysis.

1.Visualization of Outliers:

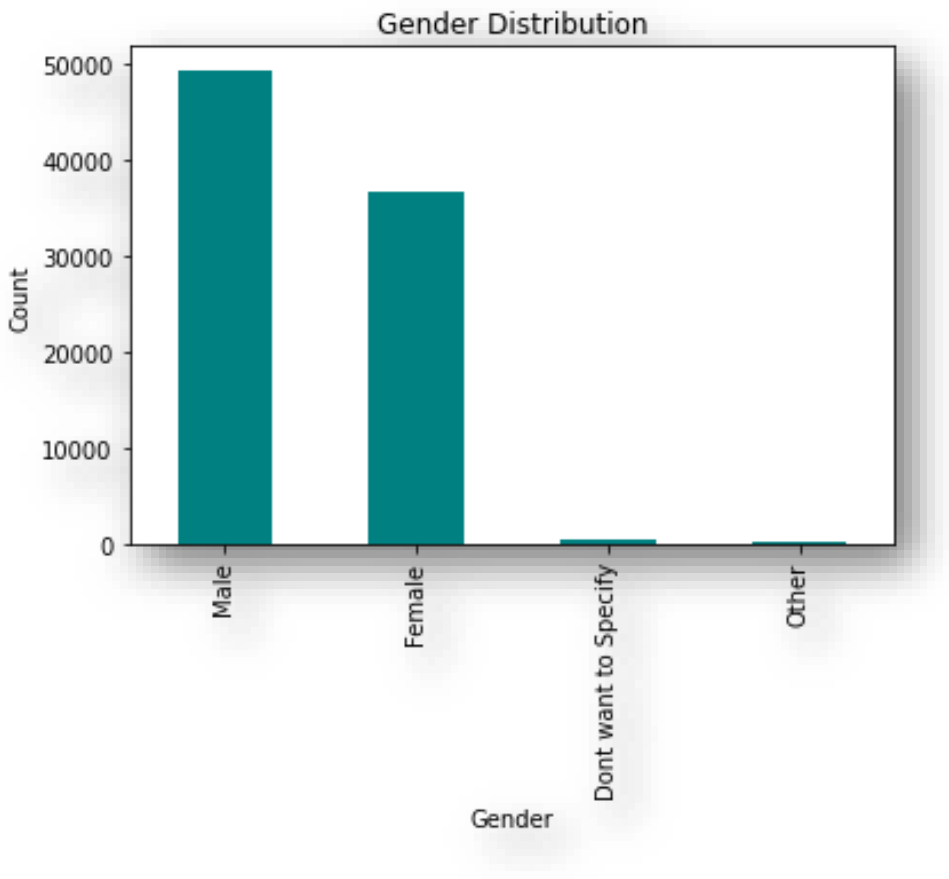


---

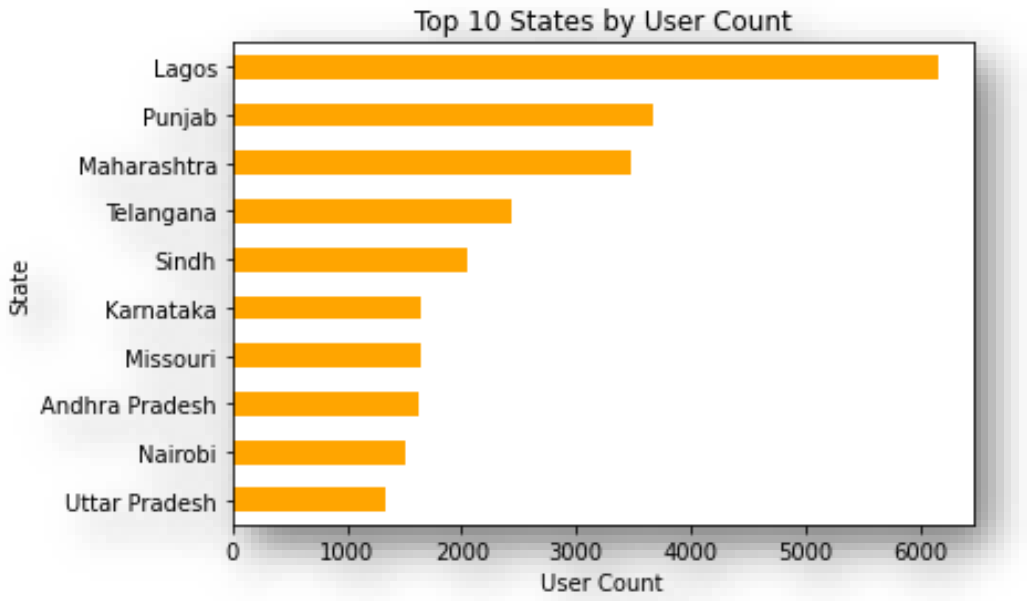
2. Distribution of User based on Age:



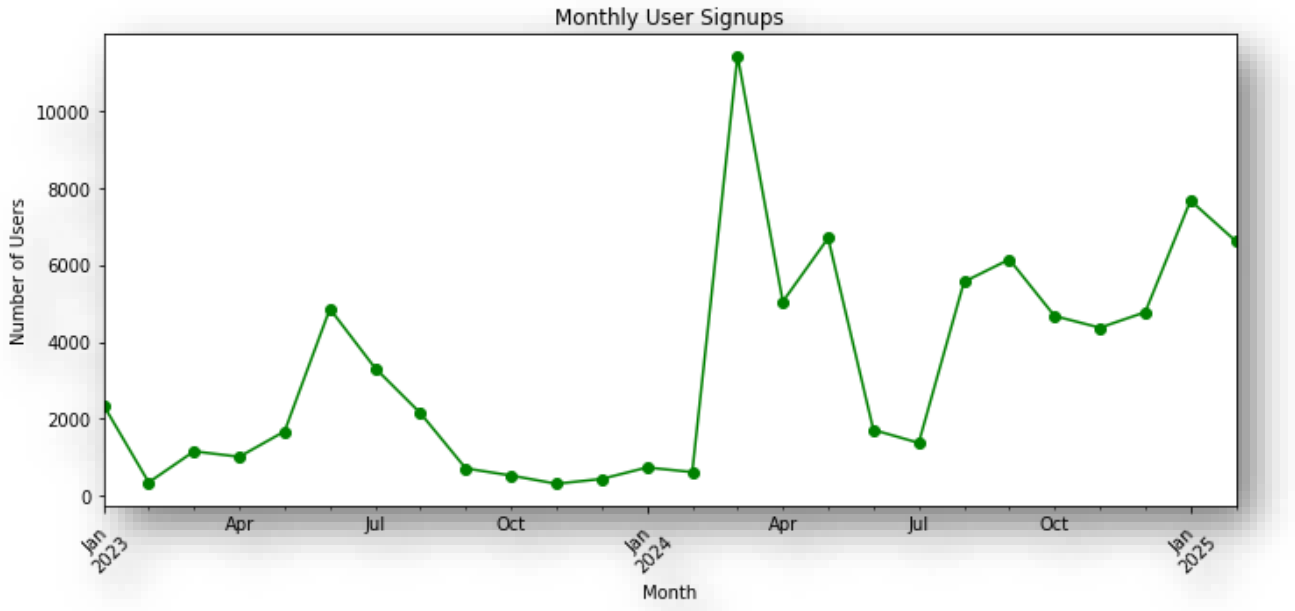
3.Gender Distribution



4.Users by State (Top 10 States)



5.Account Creation Trend Over the Specified Period



Dataset: Learner_Raw

Performing the Exploratory Data Analysis

Step 1: Understand the Dataset

- View the first few rows to get an overview of the data.
- Check column names, data types, and the total number of records.

Research how to inspect a table's structure and data in Visual Studio Code.

Pandas is a package in Python used to read data, manipulation and analysis.

Objective:

Understand the structure of the dataset by inspecting:

- First few rows
- Column names
- Data types
- Total number of records

Codes in VS-Code:

1.Reading the CSV File:

```
learner_df = pd.read_csv(r"E:\ECCELERATE\Data\Learner_Raw(in).csv")
```

---

1. View First Few Rows:

```
learner_df.head(5)
```

This gives a quick snapshot of the data format and sample values.

```
learner_df.head(5)
```

✓ 0.0s Open 'learner_df' in Data Wrangler

	learner_id	country	degree	institution	major
0	Learner#00004f18-8b86-4fe4-ad7e-(	Nigeria	Undergraduate Student	Federal University of Technology Ow	Civil Engineering
1	Learner#00006478-745f-49bf-b126-(	Nigeria	*Missing value*	*Missing value*	*Missing value*
2	Learner#00010567-1336-433c-a941-	Kenya	Graduate Student	UNICAF UNIVERSITY	Environmental Sustainability
3	Learner#00011c80-0c5c-4601-9696-	Bangladesh	*Missing value*	*Missing value*	*Missing value*
4	Learner#000141a7-4c82-401f-a2e6-(	Nigeria	*Missing value*	*Missing value*	*Missing value*

5 rows x 5 cols 10 per page « ‹ Page 1 of 1 › »

---

2. Check Column Names and Data Types:

`learner_df.info()`

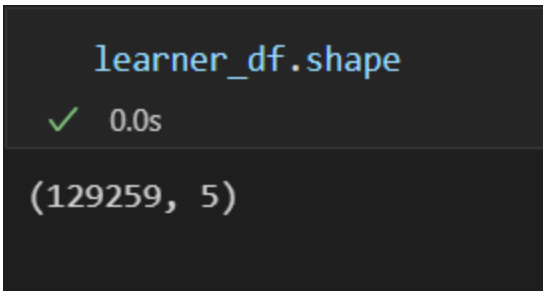
Lists each column with its data type (e.g., text, integer, timestamp).

```
learner_df.info()
✓ 0.1s
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 129259 entries, 0 to 129258
Data columns (total 5 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   learner_id   129259 non-null  object
 1   country      126984 non-null  object
 2   degree       76566 non-null   object
 3   institution  76186 non-null   object
 4   major        76388 non-null   object
dtypes: object(5)
memory usage: 4.9+ MB
```

3. Get Total Number of Rows & Columns:

`Learner_df.shape`

 Confirms how many records are present in the dataset.



Step 2: Summarize the Data

- Generate basic statistics like mean, median, min, max, and standard deviation to understand the distribution.

Since the Learner_Raw.csv dataset contains no numeric fields its not possible to do basic statistics

---

Step 1: Identify Missing and Duplicate Values

- Detect missing data to determine if any cleaning is needed.
- Identify duplicate records that may need removal.

Look up methods to check for missing values and duplicates in Visual Studio Code

Objective:

- Detect missing (NULL) values
- Identify fully or partially duplicate records

Part A: Check for Missing Values

1. Count of NULLs for All Columns

```
learner_df.isnull().sum()
```

🧠 *Compare filled counts with total rows to detect missing values.*

```
learner_df.isnull().sum()
```

✓ 0.0s

	# 0
learner_id	0
country	2275
degree	52693
institution	53073
major	52871

Part B: Check for Duplicates

3. Fully Duplicate Records

```
learner_df.duplicated().sum()
```

```
learner_df[learner_df.duplicated()]
```

📌 *These rows are identical in all fields — likely safe to remove.*

```
learner_df.duplicated().sum()
```

✓ 0.0s

```
np.int64(0)
```

```
learner_df[learner_df.duplicated()]
```

✓ 0.1s

learne... ⋯	country	degree	institution	major

- *No full duplicate rows*

What to Do Next:

Issue Type	Meaning
NULLs	May indicate missing inputs, timing issues, or data pipeline errors
Full Duplicates	Usually safe to delete unless time-sensitive duplicates are allowed

Step 4: Spot Outliers and Anomalies

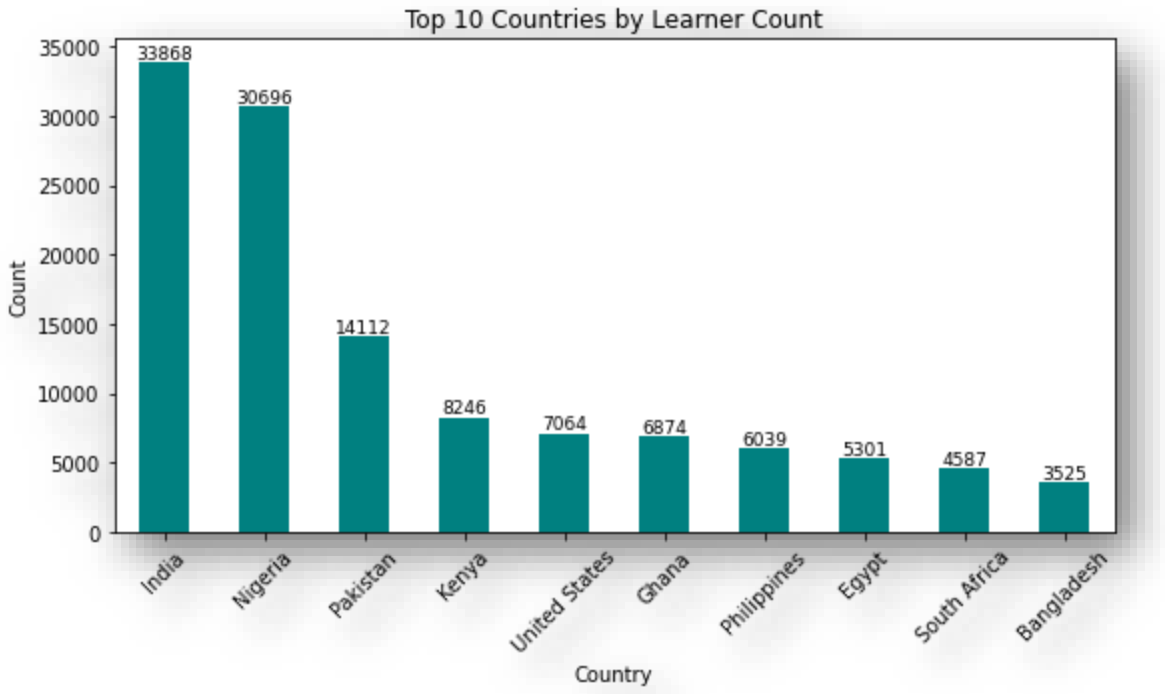
- Use techniques like the Interquartile Range (IQR) to find unusual values that might distort analysis.
- Objective of finding the outliers is to find which record is being odd from the other records
- But here all the columns are categorial so we can’t do usual IQR method to find the outliers.
- If we go for rare, uncommon, low frequency occurrence data as outliers we may lose data of users.
- So we can’t do outlier analysis for this dataset

Step 5: Visualize Data Trends

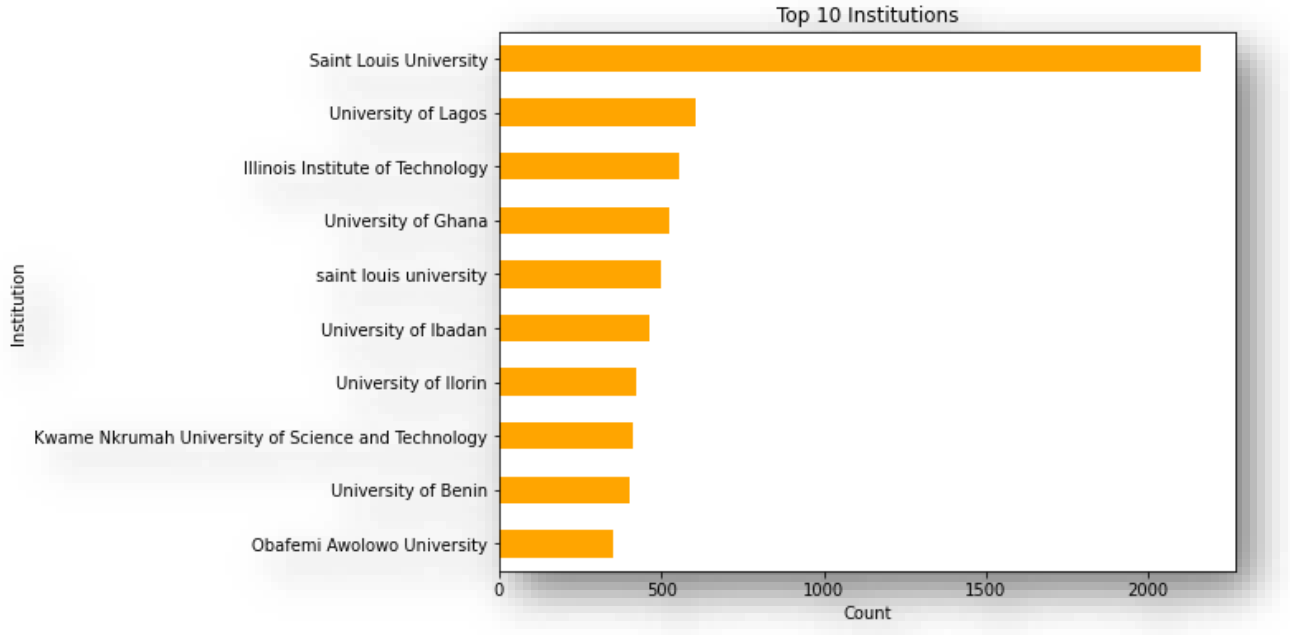
- Use graphs like histograms, line charts, and box plots to identify patterns.
- Analyze relationships between different variables.

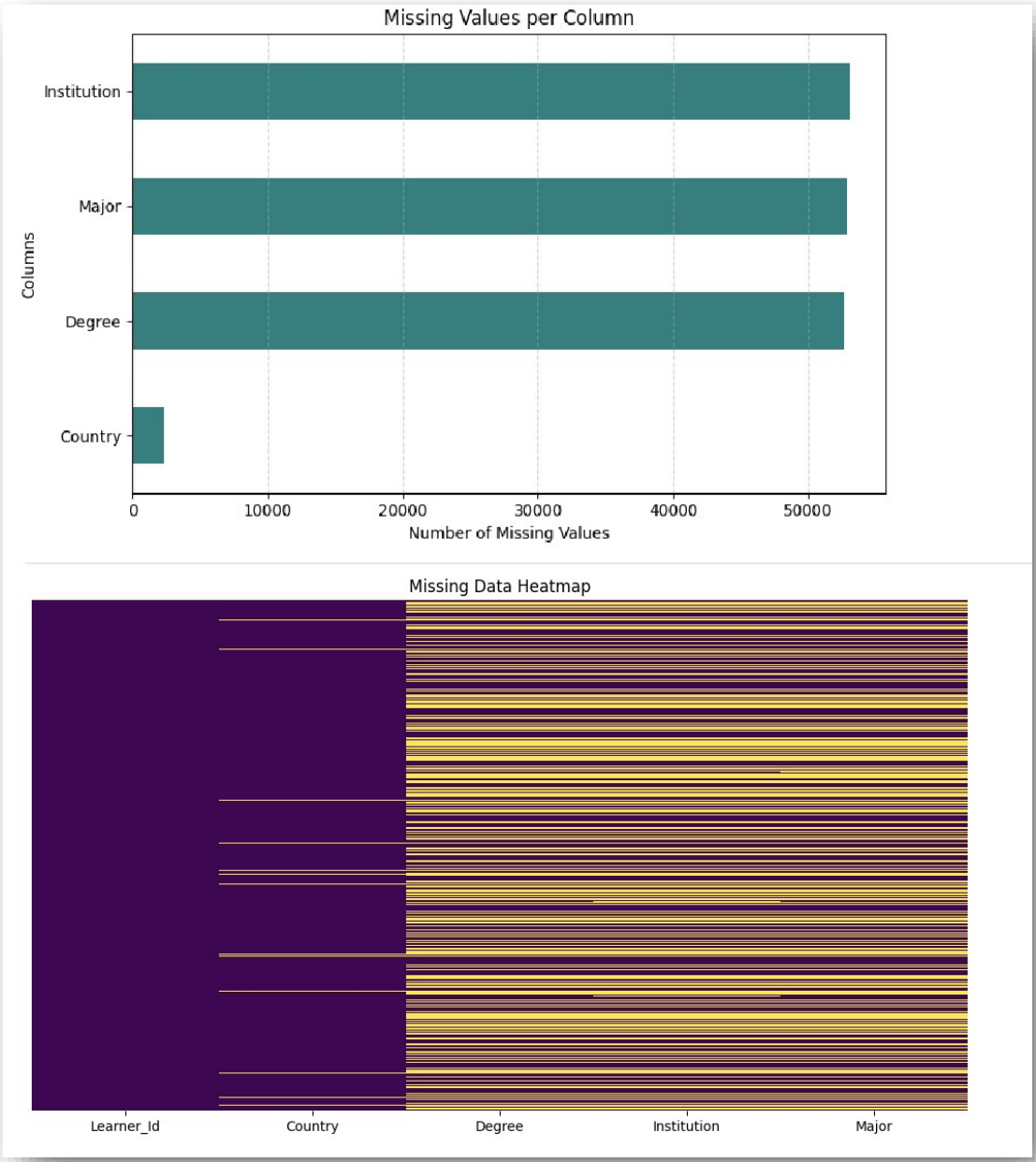
Find out how to create basic visualizations for data analysis.

1.Top Countries by Learner Count (Top 10 Countries)



2.Top 10 Institutions




Missing Values per Column
Institution
Major
Degree
Country
Columns
0
10000
20000
30000
40000
50000
Number of Missing Values
Missing Data Heatmap
Learner_id
Country
Degree
Institution
Major

Dataset: Opportunity_Raw

Performing the Exploratory Data Analysis

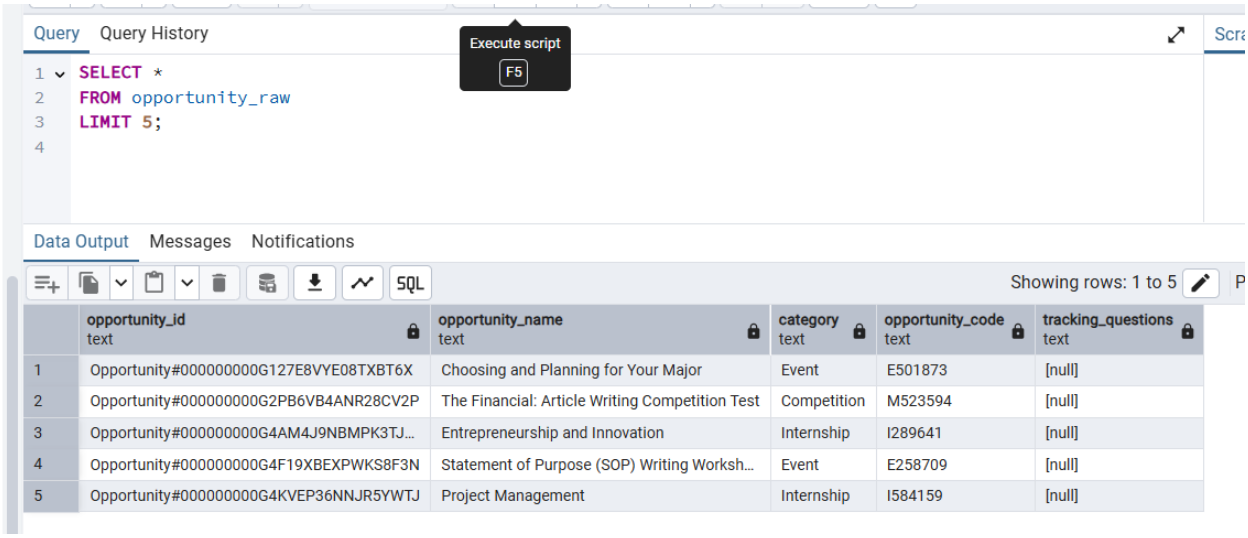
Step 1: Understand the Dataset

- View the first few rows to get an overview of the data.
- Check column names, data types, and the total number of records.

Research how to inspect a table's structure and data in Visual Studio Code.

View First Few Rows (Top 5 Records)

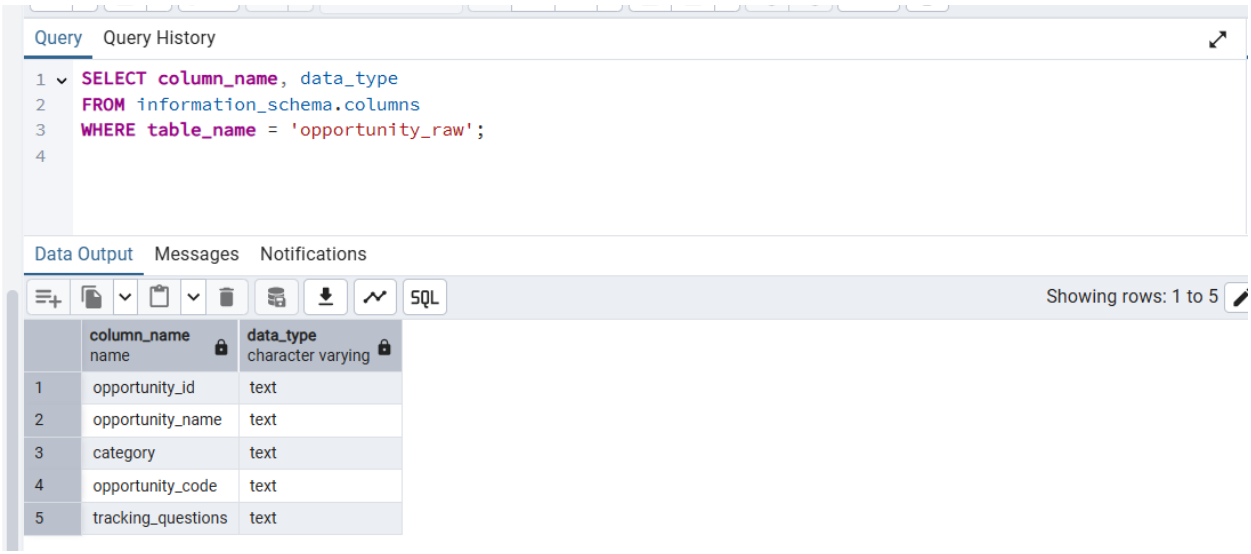
```
SELECT *
FROM opportunity_raw
LIMIT 5;
```



	opportunity_id text	opportunity_name text	category text	opportunity_code text	tracking_questions text
1	Opportunity#000000000G127E8VYE08TXBT6X	Choosing and Planning for Your Major	Event	E501873	[null]
2	Opportunity#000000000G2PB6VB4ANR28CV2P	The Financial: Article Writing Competition Test	Competition	M523594	[null]
3	Opportunity#000000000G4AM4J9NBMPK3TJ...	Entrepreneurship and Innovation	Internship	I289641	[null]
4	Opportunity#000000000G4F19XBEXPWKS8F3N	Statement of Purpose (SOP) Writing Worksh...	Event	E258709	[null]
5	Opportunity#000000000G4KVEP36NNJR5YWTJ	Project Management	Internship	I584159	[null]

Check Column Names & Data Types

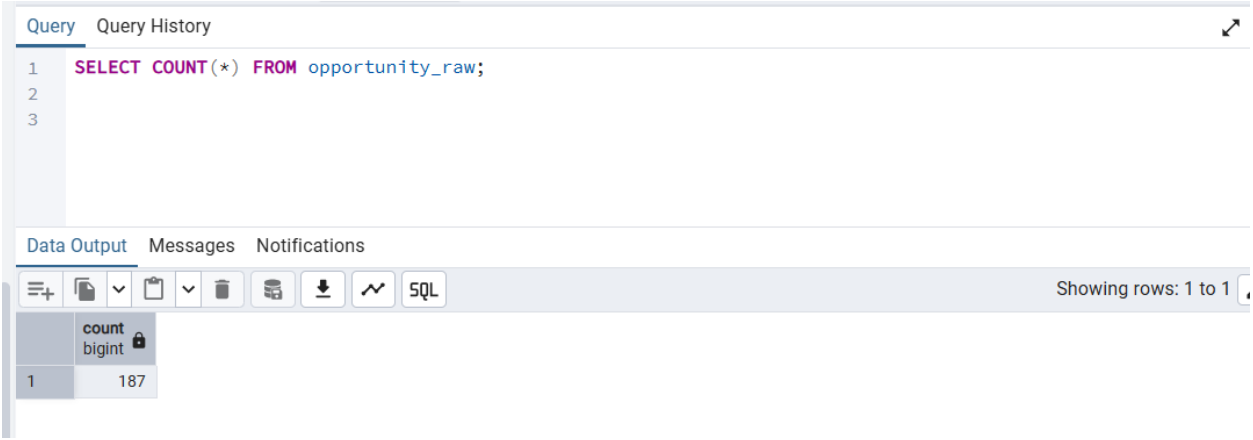
```
SELECT column_name, data_type
FROM information_schema.columns
WHERE table_name = 'opportunity_raw';
```



	column_name name	data_type character varying
1	opportunity_id	text
2	opportunity_name	text
3	category	text
4	opportunity_code	text
5	tracking_questions	text

Count Total Rows

```
SELECT COUNT(*) FROM opportunity_raw;
```



	count bigint
1	187

Step 2: Summarize the Data

- Generate basic statistics like mean, median, min, max, and standard deviation to understand the distribution.

Explore how to compute summary statistics in SQL.

Objective:

• Generate basic statistics like mean, median, min, max, and standard deviation to understand the distribution.
• Explore how to compute summary statistics in SQL.

Not Applicable to This Dataset

The `opportunity_raw` dataset **does not contain any numeric columns**, which means **standard statistical functions such as**:

- `MIN(size)`
- `MAX(size)`
- `AVG(size)`
- `STDDEV(size)`
- `MEDIAN(size)`

are not applicable to this table.

All columns — such as `opportunity_id`, `opportunity_name`, `category`, `opportunity_code`, and `tracking_questions` — are of **textual (string) data type**, not numeric.

Step 3: Identify Missing and Duplicate Values

- Detect missing data to determine if any cleaning is needed.
- Identify duplicate records that may need removal.

Look up methods to check for missing values and duplicates in PostgreSQL.

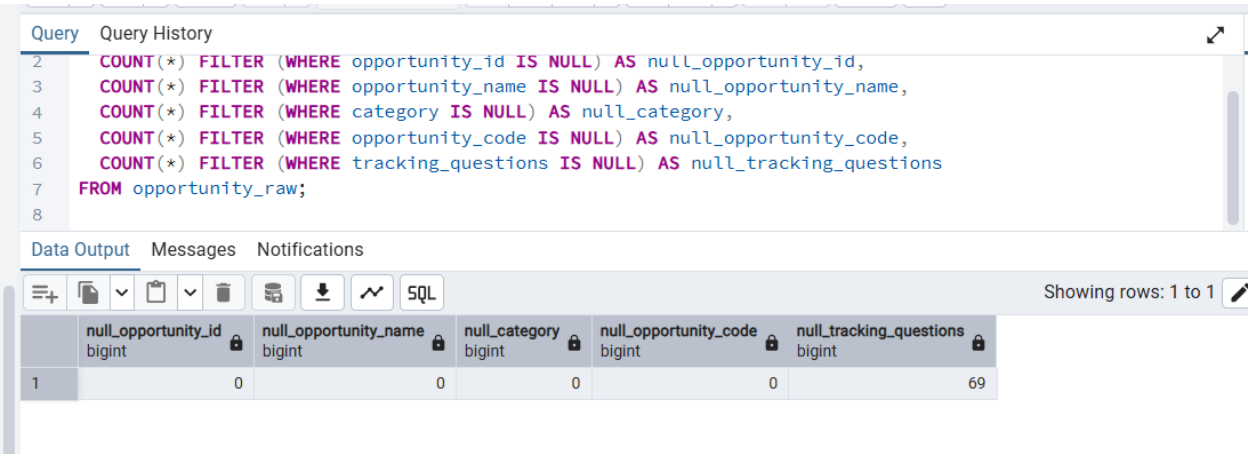
Objective:

• Detect missing data to determine if any cleaning is needed
• Identify duplicate records that may need removal
• Use PostgreSQL queries only — no Python

◆ **Part A: Check for Missing (NULL) Values**

Query:

```
SELECT
  COUNT(*) FILTER (WHERE opportunity_id IS NULL) AS null_opportunity_id,
  COUNT(*) FILTER (WHERE opportunity_name IS NULL) AS null_opportunity_name,
  COUNT(*) FILTER (WHERE category IS NULL) AS null_category,
  COUNT(*) FILTER (WHERE opportunity_code IS NULL) AS null_opportunity_code,
  COUNT(*) FILTER (WHERE tracking_questions IS NULL) AS
null_tracking_questions
FROM opportunity_raw;
```

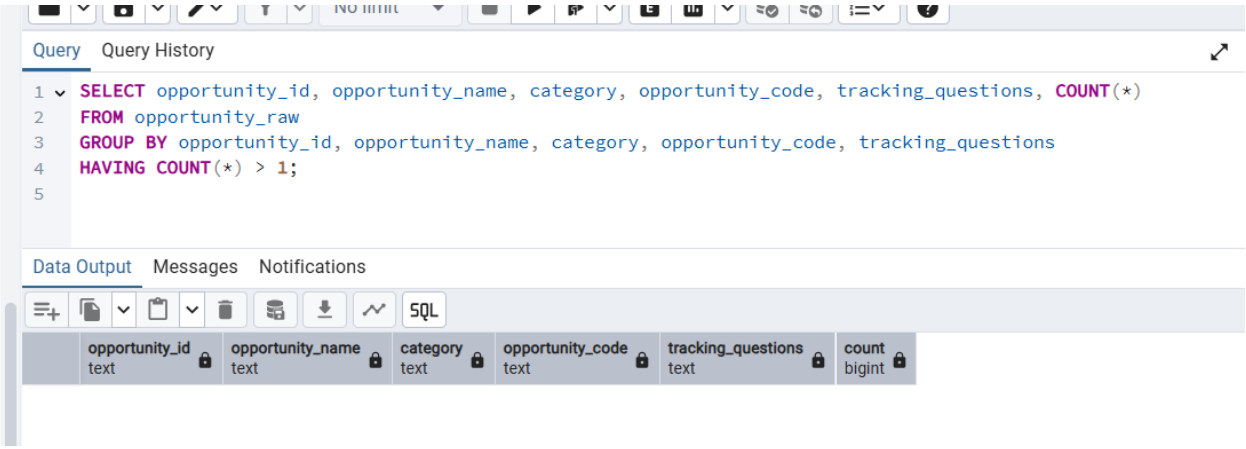


- **`tracking_questions` is completely empty (NULL in all records), likely optional or inactive.**

◆ Part B: Check for Fully Duplicate Records

Query:

```
SELECT opportunity_id, opportunity_name, category, opportunity_code,
tracking_questions, COUNT(*)
FROM opportunity_raw
GROUP BY opportunity_id, opportunity_name, category, opportunity_code,
tracking_questions
HAVING COUNT(*) > 1;
```



Result Interpretation:

- **0 fully duplicate records** were found.
- No rows are exact matches across all columns.

What to Do Next:

Issue Type	Recommended Action
`tracking_questions` (all NULLs)	Investigate if this column is still needed or was never populated
Full Duplicates	✅ None found — no cleanup needed

Summary:

Step 3 confirms that the dataset has **no duplicate records** and only **one column (`tracking_questions`) with 100% NULL values**.
No urgent cleaning is needed — but tracking data can be archived or flagged for review.

Step 4: Spot Outliers and Anomalies

- Use techniques like the Interquartile Range (IQR) to find unusual values that might distort analysis.

Research how outliers are detected using SQL and statistical methods.

Objective:

• Use techniques like the Interquartile Range (IQR) to detect unusual values
• First: **Determine if outlier detection is relevant for this dataset**

Is Outlier Detection Applicable?

Requirement	`opportunity_raw` Status
Contains **numeric data**	✗ No — all columns are of type `TEXT`
Has measurable values (e.g. size, price, score)	✗ No — only identifiers, names, and categories
Columns suitable for IQR/mean/std-based checks	✗ None present

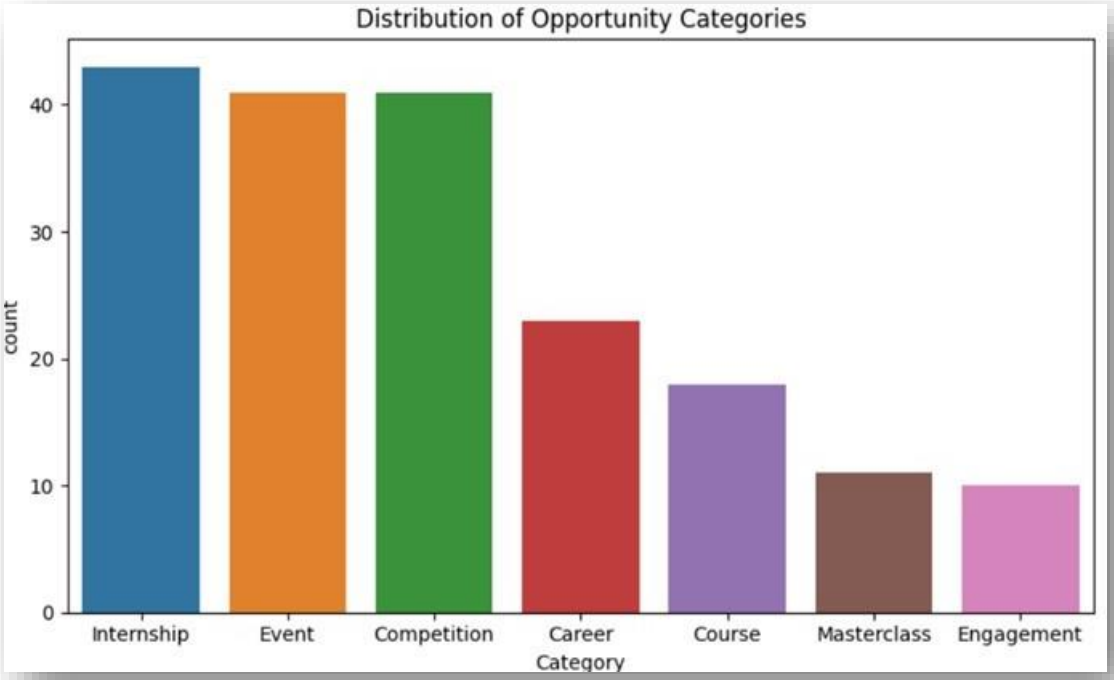
---

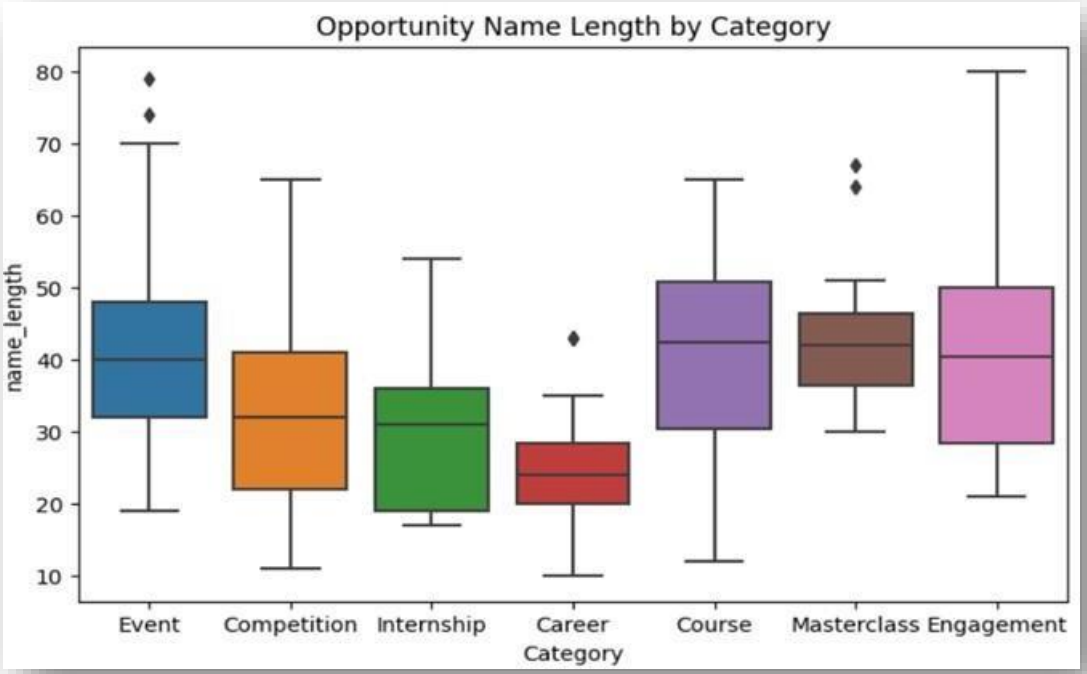
Conclusion:

✗ **Outlier detection is not applicable** to the `opportunity_raw` dataset.
The dataset contains only **categorical and identifier fields**, with no numeric data to evaluate using statistical methods like **IQR, z-scores, or standard deviation**.

Step 5: Visualize Data Trends

- Use graphs like histograms, line charts, and box plots to identify patterns.
- Analyze relationships between different variables.


Distribution of Opportunity Categories
count
0
10
20
30
40
Internship
Event
Competition
Career
Course
Masterclass
Engagement
Category


Opportunity Name Length by Category
name_length
10
20
30
40
50
60
70
80
Event
Competition
Internship
Career
Course
Masterclass
Engagement
Category

Key Findings & Next Steps for Data Cleaning and Transformation

Contain all Datasets

1. Learner_Raw Dataset

Key Findings:

- **Primary Key `learner_id` is valid** — unique and clean.
- **High missing values** in:
  - `degree`: ~41%
  - `institution`: ~41%
  - `major`: ~41%
  - `country`: 2,275 missing entries.
- **Inconsistent text formatting**:
  - Degree types: "Bachelor", "Bachelors", "BS", etc.
  - Major: Case inconsistencies (e.g., “computer science” vs “Computer Science”).
- No valid **Foreign Key (FK)** relationships with `Cognito_Raw` or `LearnerOpportunity_Raw`.

Next Steps for Cleaning:

- **Normalize text** (standardize casing for degree and major).
- **Deduplicate vague entries** (e.g., “null”, “Other”).
- **Consider imputation**:
  - Categorical mode for `degree`, `institution`, `major` if used in modeling.
- **Retain as independent dimension table** until foreign key alignment is confirmed.

---

2. Opportunity_Raw Dataset

Key Findings:

- `opportunity_id`: **Valid Primary Key**.
- `opportunity_code`: Appears unique; possible alternate key.
- `tracking_questions`: **~90% NULL** → likely deprecated.
- Duplicate `opportunity_name` values exist.

Next Steps for Cleaning:

- **Drop `tracking_questions`** unless confirmed relevant.

- **Deduplicate or classify `opportunity_name`** if required for reporting.
- **Standardize `category`** (Event, Internship, Competition) for consistency.
- Prepare for **join with `LearnerOpportunity_Raw`** via `opportunity_id`.

---

3. Cognito_Raw2 Dataset

Key Findings:

- `user_id`: Clean, unique — **strong PK**.
- `email`: Mostly complete, but **duplicate risk**.
- Missing/inconsistent in:
  - `gender` (~22% missing)
  - `birthdate` (~10%)
  - `city`, `zip`, `state` (~15%)
- Date fields (`UserCreateDate`, `UserLastModifiedDate`) stored as **strings**.
- `custom:learner_id` does **not match `learner_id`** in `Learner_Raw`.

Next Steps for Cleaning:

- **Convert** all date fields to proper `datetime` format.
- **Standardize**:
  - `gender`, `city`, `state` values
- **Validate uniqueness** of email if used as a secondary ID.
- Assess if **`user_id` or `email`** can act as bridge to learner-level datasets.

---

4. Cohort_Raw Dataset

Key Findings:

- `cohort_code`: Clean, appears **unique** — candidate PK.
- `cohort_id`: **Redundant**, same placeholder in all rows.
- `start_date`, `end_date`: Stored in **epoch timestamp (ms)** → unreadable as-is.
- `size`: Numerical, clean — but **outliers suspected** (e.g., 100,000).

Next Steps for Cleaning:

- **Drop `cohort_id`** — not useful.
- **Convert `start_date` and `end_date`** to human-readable datetime.
- **Analyze `size` outliers**; define thresholds for valid cohort sizes.
- Confirm whether size reflects **actual enrollment** or **capacity**.

---

5. LearnerOpportunity_Raw Dataset

Key Findings:

- **186 exact duplicates** found.
- `assigned_cohort`: Missing in ~11.7% of records.
- `apply_date`: Valid ISO format but stored as **text** — 188 missing.
- `status`: Numeric codes (e.g., 1070, 1010) — **undocumented** meanings.
- `enrollment_id` is **not unique** — consider composite key with `learner_id`.

Next Steps for Cleaning:

- **Remove exact duplicate rows**.
- **Decode `status` values** — obtain mapping documentation.
- **Convert `apply_date`** to datetime.
- **Impute `assigned_cohort`** if used in cohort-level analysis.
- **Treat `enrollment_id + learner_id`** as composite primary key.

---

6. Marketing_Campaign_Raw Dataset

Key Findings:

- No single primary key — **possible composite key**: `Campaign name + Result type + Reporting starts`.
- **5 duplicate rows**.
- Missing data in critical fields (4–6%):
  - `reach`, `results`, `clicks`, `cost per result`, etc.
- **Outliers**:
  - `Reach`: Max = 141M+, Mean = 1.7M
  - `Results`: Max = 182M
- `Reporting starts`: Only **1 unique value** — not useful for time-series.

Next Steps for Cleaning:

- **Remove duplicate rows**.
- **Impute or drop missing** numeric fields (consider imputing with 0 or median, depending on business context).
- **Investigate and handle outliers** (cap or segment).
- **Ignore or enrich `Reporting starts`** — may require full timeline data.
- Standardize categorical fields (e.g., `Delivery status`, `Result type`).

Executive Summary: Cross-Dataset Next Steps

Task	Applies To	Priority
Normalize and standardize text	Learner_Raw, Cognito_Raw2	High
Convert text to datetime	Cohort_Raw, Cognito_Raw2, LearnerOpp	High
Remove duplicates	LearnerOpp_Raw, Marketing_Campaign_Raw	High
Handle outliers	Cohort_Raw, Marketing_Campaign_Raw	Medium
Decode numeric codes	LearnerOpp_Raw (`status`)	High
Drop deprecated columns	Opportunity_Raw (`tracking_questions`)	High
Impute missing data	Across all datasets	High
Validate key relationships	All datasets (joins, PK/FK alignment)	High