User Guide: Observability Analytics Query Builder

Authors: Joy Wang Sam Birus

Last update: October 2025

Table of content

Introduction

We are excited to announce that Observability Analytics (OA) query builder is now General Available!

If you are a Cloud Ops's Observability Analytics user (also known as Log Analytics), this feature is for you!

Observability query builder providing below key benefits:

- **No more writing SQL query:** Generating queries and getting results through a guided UI, no longer need to write SQL for most of your analytics questions. And you can continue writing SQL if you like.
- **Easy to extract, cast, and transform JSON data:** Query builder is designed to streamline some of the complexities of working with Observability Analytics data, such as extracting, casting, and transforming JSON data.
- **Toggle between builder and SQL**: You can toggle between query builder and automatically generated SQL, and make edits with the SQL to achieve more complex query.
- **Work with Log Scopes**: Query builder enables you to query with log scope. So you can keep the scope consistent when doing deeper analysis and troubleshooting in log explorer.

This user guide will cover steps for enabling the query builder and provide some examples for common use cases.

Enable Observability Analytics

In order to use the query builder, first enable Log Analytics.

Query Builder UI Layout

Log view, trace views, analytics views, metrics view viewer: show the list of available views

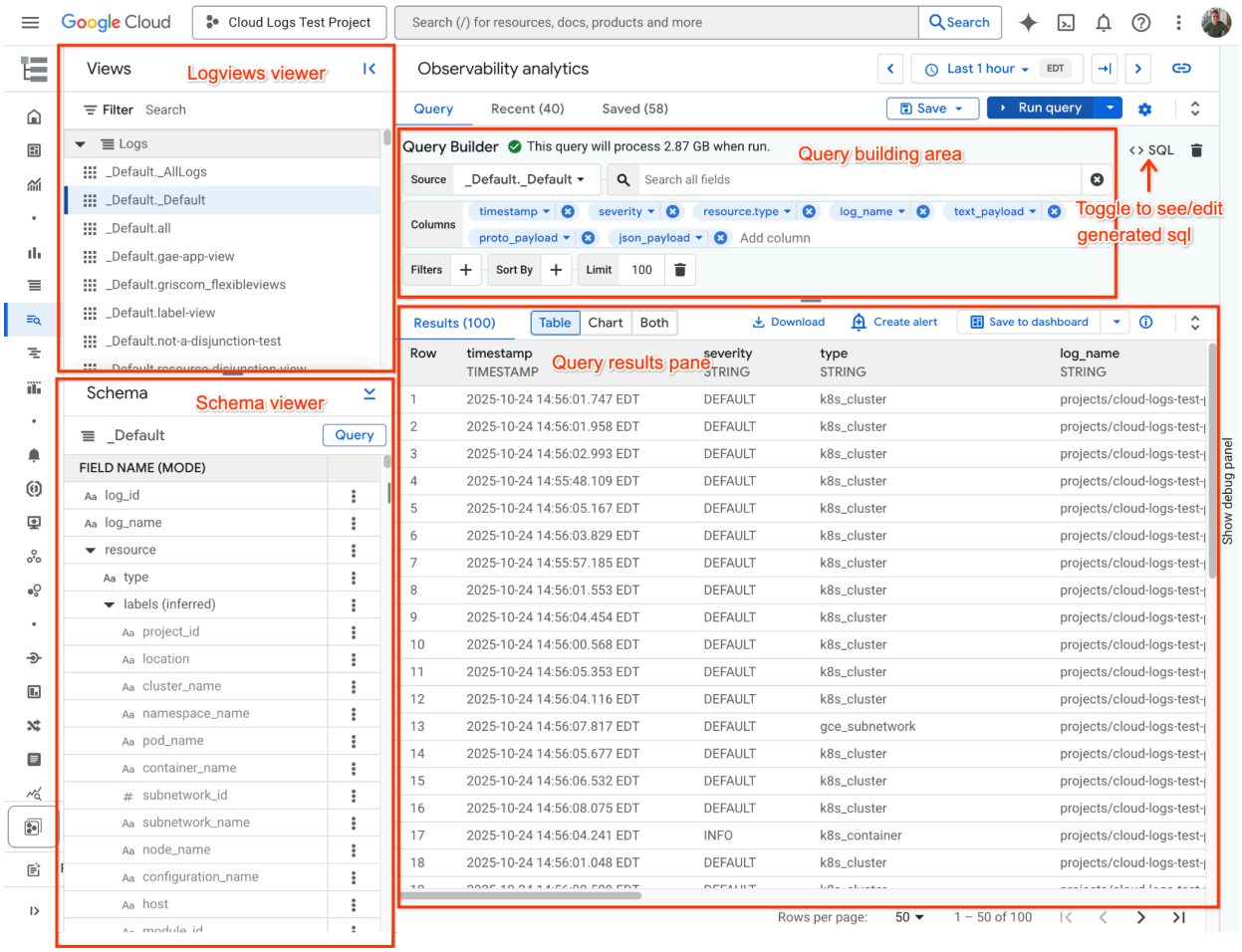
Log Schema viewer: Display log schema, including nested fields

Query Building area:

- **View select:** Selecting log view, **log scope**, metrics view, trace view and analytics views,
- Construct query and answer questions by selecting fields, filters, aggregations, sortings, apply regex, alias, casting value type and time function.

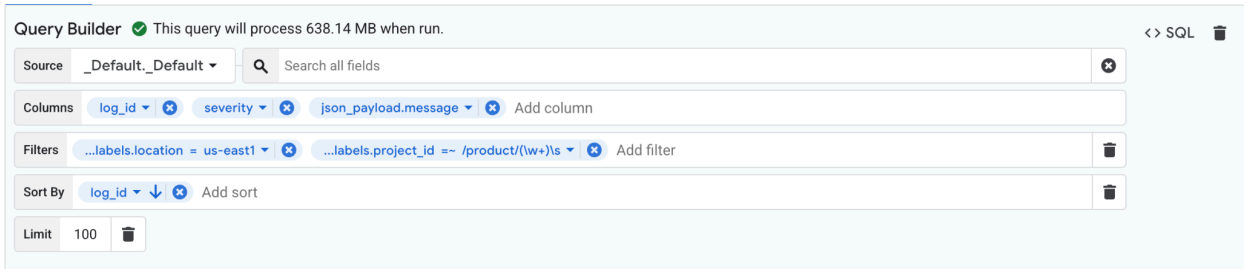
Query result plane:

- Display results from the construct.
- Data time range is automatically applied from the time range picker on the top right



Query Builder UI Component

The query builder UI is split into several fields.



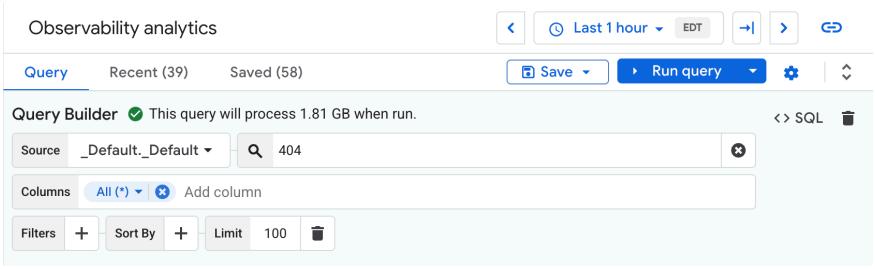
- “Source” - allows you to pick the base view for the query, which will be presented in the FROM section. In addition to supporting log views and analytics views, there is also support for Log Scopes, which allow querying multiple views at once.
- “Search all fields” - Provides easy filtering using a simple text based search against the entire row, similar to the Logs Explorer’s search bar.
- “Columns” - Allows you to specify the columns that will appear in the output table
- “Filters” - Allows you to apply filters against fields of the source data
- “Sort by” - Allows you to specify the columns to sort by

- “Limit” - Set the number of output rows

Sample Use Cases

Example 1: Search ‘404’ in all fields

This query builder construct allows you to search term ‘404’ across all fields of the log, and return the log as long as there is a match in any field of the log.



Generated SQL query

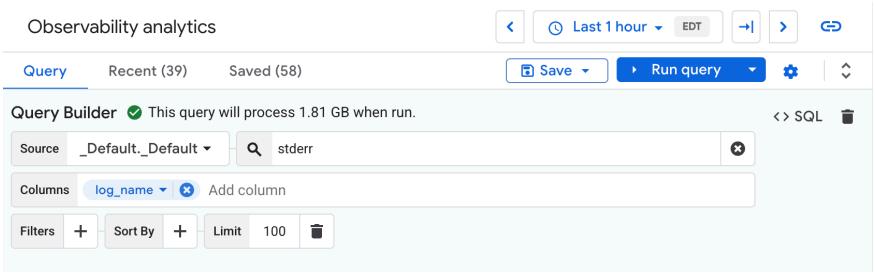
```
SQL
WITH
  scope_query AS (
  SELECT
    *
  FROM
    `query-builder-test-project.global._Default._Default` )
SELECT
  *
FROM
  scope_query
WHERE
  SEARCH(scope_query,
    '404')
LIMIT
  100
```

Expected results

Row	log_id	text_payload	timestamp
	STRING	STRING	TIMESTAMP

1	stdout	10.0.3.1 - - [02/May/2025:18:29:06 +0000] "GET /sendgrid.env HTTP/1.1" 404 153 "-" "l9explore/1.2.2" "-"	2025-05-02 14:29:06.268 EDT

Example 2: Search for 'stderr' across the dataset and return log_name for matching results



Generated SQL query

```sql
WITH
  scope_query AS (
  SELECT
    *
  FROM
    `query-builder-test-project.global._Default._Default` )
SELECT
  log_name
FROM
  scope_query
WHERE
  SEARCH(scope_query,
    'stderr')
LIMIT
  100
```

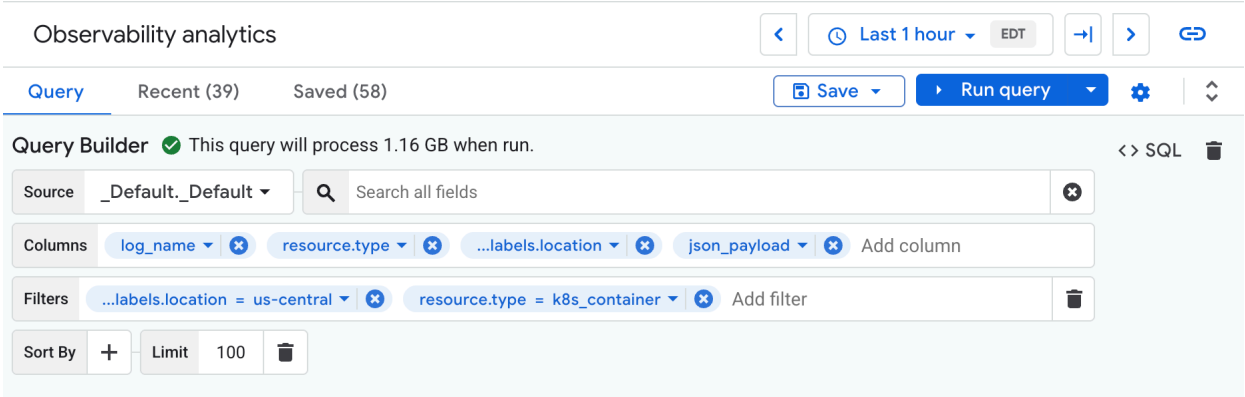
Expected results

Row	log_name STRING
1	projects/query-builder-test-project/logs/stderr

2	projects/query-builder-test-project/logs/stderr

Example 3: Select fields with filters (equality filter)

This query builder construct allows you to select log name, resource type, resource location, JSON payload from logs, where resource type is k8s_container and location is us-central1



Generated SQL query

```
SQL
WITH
  scope_query AS (
  SELECT
    *
  FROM
    `query-builder-test-project.global._Default._Default` )
SELECT
  log_name,
  resource.type,
  JSON_VALUE( resource.labels.location ) AS location,
  json_payload AS json_payload
FROM
  scope_query
WHERE
  JSON_VALUE( resource.labels.location ) = 'us-central'
  AND resource.type = 'k8s_container'
LIMIT
  100
```

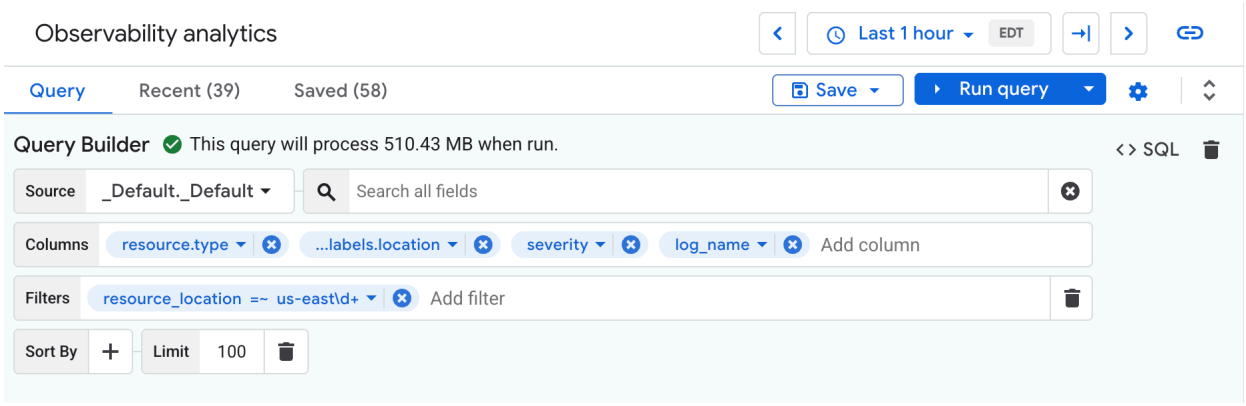
Expected results

Row	log_name STRING	type STRING	oa0_ STRING	json_payload

1	projects/query-builder-test-project/logs/stdout	k8s_container	us-central1	*Null*
2	projects/query-builder-test-project/logs/stdout	k8s_container	us-central1	*Null*
3	projects/query-builder-test-project/logs/stdout	k8s_container	us-central1	*Null*

Example 4: Alias selected fields, with regex filters

Selecting resource type, resource location, severity and log name where resource locations are in all of us-east locations.



Generated SQL query

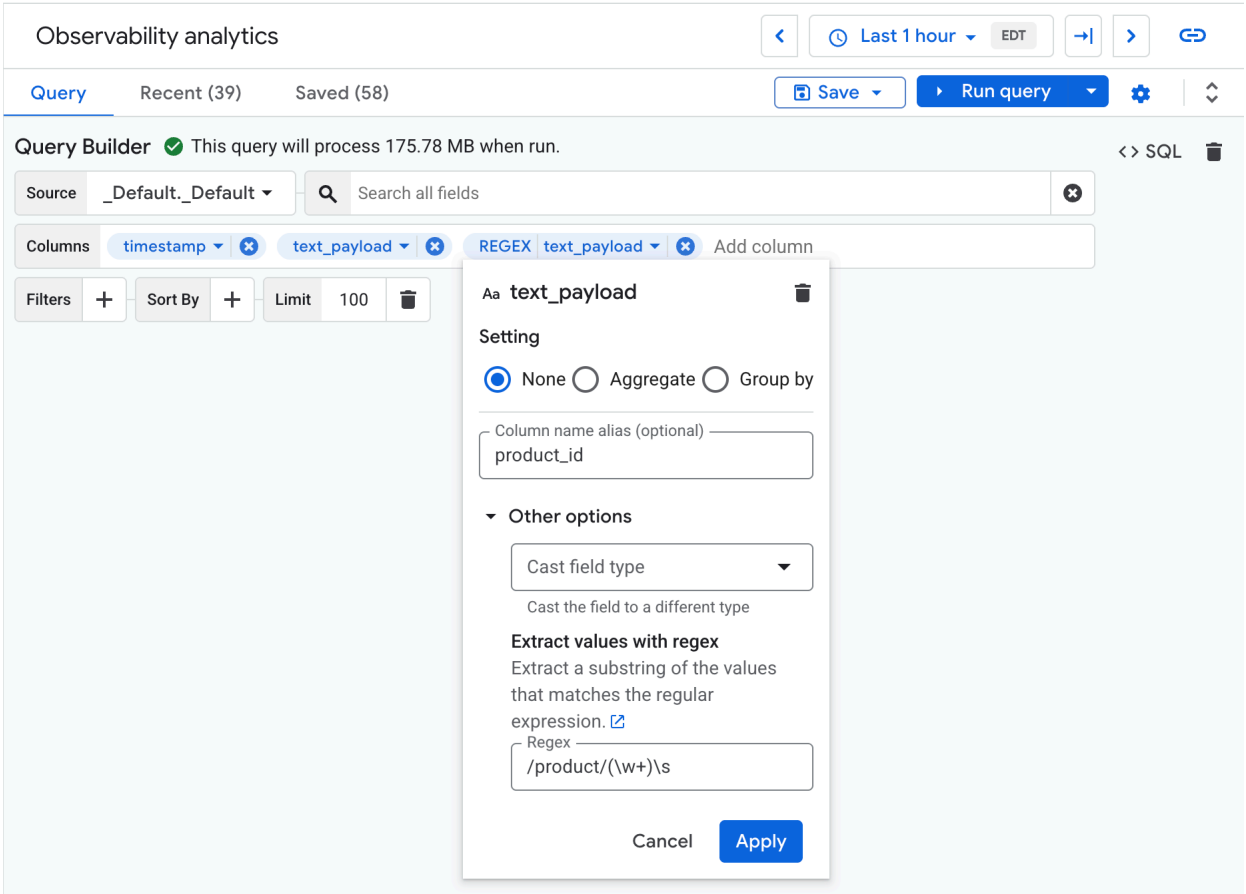
```sql
WITH
  scope_query AS (
  SELECT
    *
  FROM
    `query-builder-test-project.global._Default._Default` )
SELECT
  resource.type AS resource_type,
  JSON_VALUE( resource.labels.location ) AS resource_location,
  severity,
  log_name
FROM
  scope_query
WHERE
  JSON_VALUE( resource.labels.location ) = 'us-east\\d+'
LIMIT
  100
```

Expected results

Row	resource_type STRING	resource_location STRING	severity STRING	log_name STRING
1	k8s_cluster	us-east1-a	DEFAULT	projects/query-builder-test-project/logs/cloudaudit.googleapis.com%2Factivity
2	k8s_cluster	us-east4-b	DEFAULT	projects/query-builder-test-project/logs/cloudaudit.googleapis.com%2Factivity

Example 5: Alias selected field with regex extraction

Selecting timestamp, text_payload, and an aliased product_id from the text_payload column with a regex extraction applied.



Generated SQL query

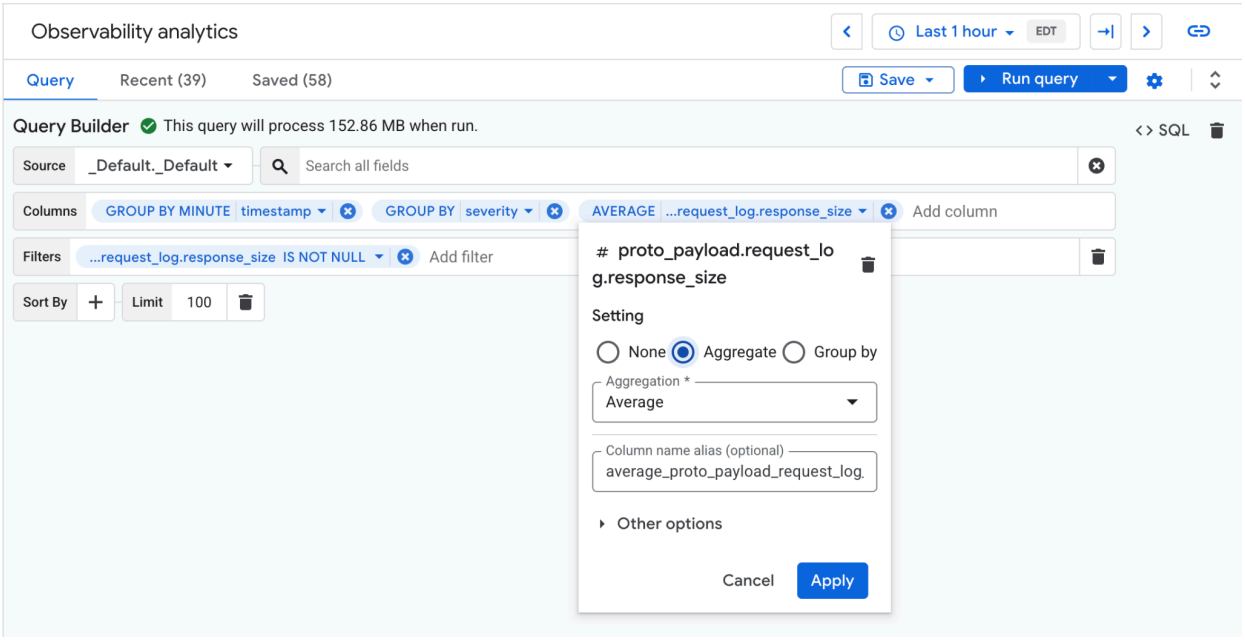
```sql
WITH
  scope_query AS (
  SELECT
    *
  FROM
    `query-builder-test-project.global._Default._Default` )
SELECT
  timestamp,
  text_payload,
  REGEXP_EXTRACT( text_payload, '/product/(\\w+)\\s') AS product_id
FROM
  scope_query
LIMIT
  100
```

Expected results

Row	timestamp	text_payload	product_id
	TIMESTAMP	STRING	STRING
1	2025-05-02 10:16:29.749 EDT	GET /product/1PUX7V6EV0 12345 12345(100.00%) | 11 2 3333 00 | 0.00 0.00	1PUX7V6EV0
2	2025-05-02 10:16:31.755 EDT	GET /product/1PUX7V6EV0 12345 85483(100.00%) | 11 2 3333 00 | 0.00 0.00	1PUX7V6EV0
3	2025-05-02 10:22:44.673 EDT	GET /product/1PUX7V6EV0 12345 12345(100.00%) | 11 2 3333 00 | 0.30 0.30	1PUX7V6EV0

Example 6: Fields with aggregation and grouping, filter with IS NOT NULL

Select average response size and group by severity and timestamp



Generated SQL query

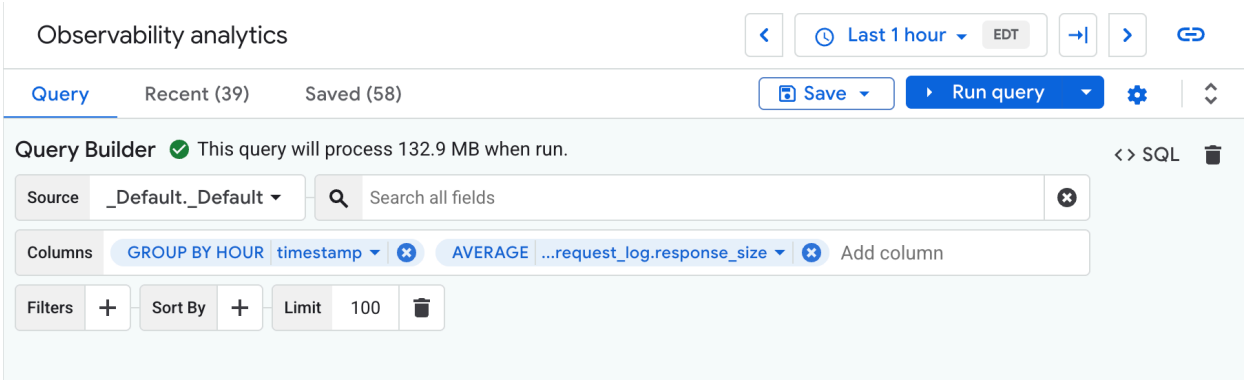
```sql
WITH
  scope_query AS (
  SELECT
    *
  FROM
    `query-builder-test-project.global._Default._Default` )
SELECT
  TIMESTAMP_TRUNC( timestamp, MINUTE ) AS minute_timestamp,
  severity,
  AVG( proto_payload.request_log.response_size ) AS
average_proto_payload_request_log_response_size
FROM
  scope_query
WHERE
  proto_payload.request_log.response_size IS NOT NULL
GROUP BY
  TIMESTAMP_TRUNC( timestamp, MINUTE ),
  severity
LIMIT
  100
```

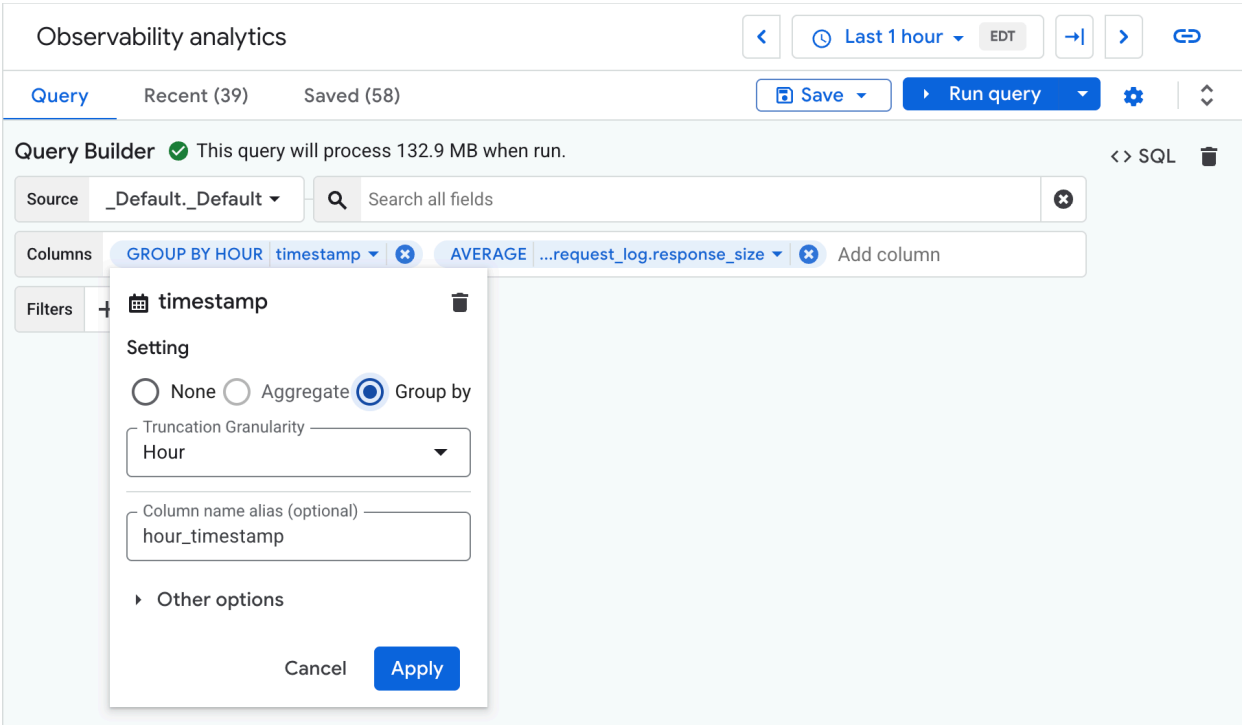
Expected results

Row	timestamp	severity	response_size
	TIMESTAMP	STRING	INTEGER
1	2025-05-02 14:00:00.000 EDT	INFO	431
2	2025-05-02 13:00:00.000 EDT	NOTICE	2540

Example 7: Group by hour truncation from timestamp

Group response size by hours exacted from timestamp





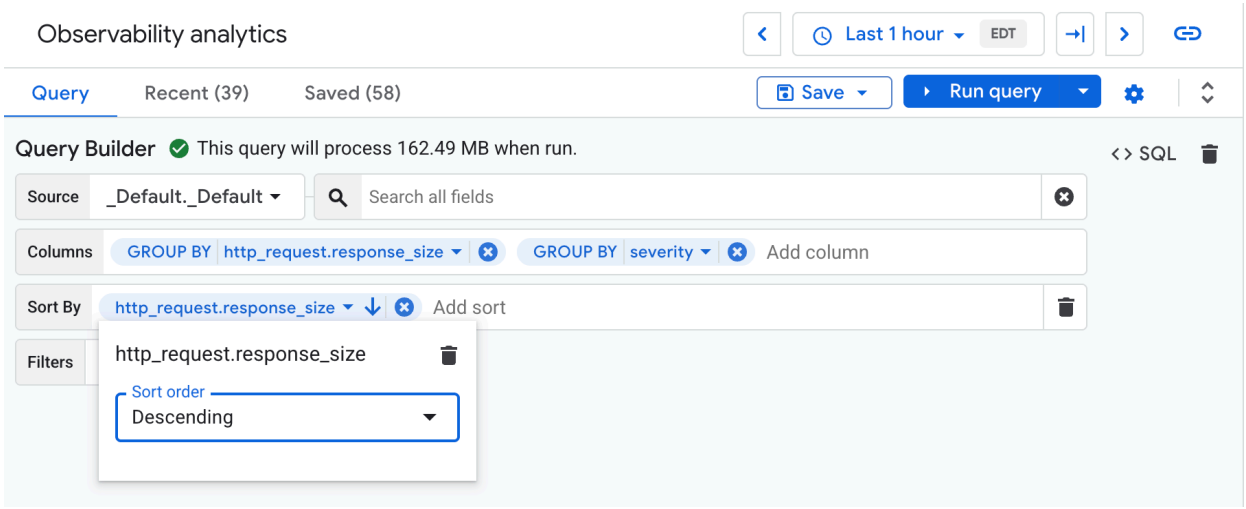
Generated SQL query

```sql
WITH
  scope_query AS (
  SELECT
    *
  FROM
    `query-builder-test-project.global._Default._Default` )
SELECT
  TIMESTAMP_TRUNC( timestamp, HOUR ) AS hour_timestamp,
  AVG( proto_payload.request_log.response_size ) AS
average_proto_payload_request_log_response_size
FROM
  scope_query
GROUP BY
  TIMESTAMP_TRUNC( timestamp, HOUR )
LIMIT
  100
```

Expected results

Row	event_hour	response_size
	TIMESTAMP	FLOAT
1	2025-05-02 14:00:00.000 EDT	756.93939393939
2	2025-05-02 08:00:00.000 EDT	761.92473118279
3	2025-05-02 14:00:00.000 EDT	*Null*

Example 8: Filter and some number of fields with sort



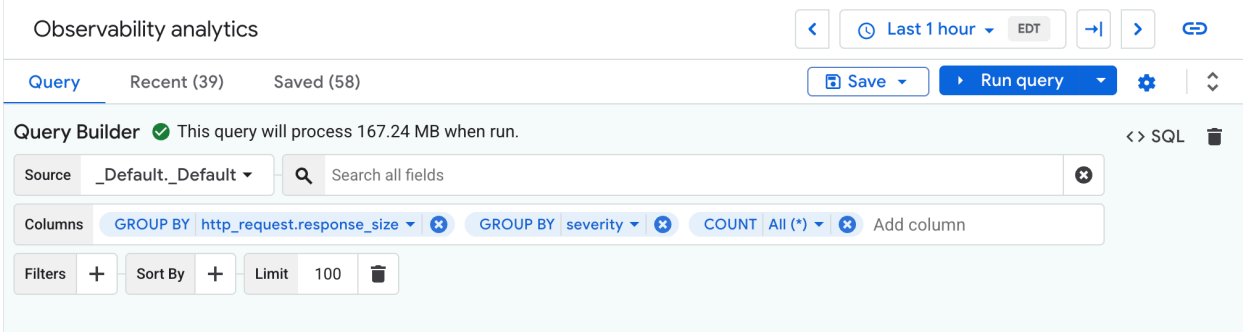
Generated SQL query

```sql
WITH
  scope_query AS (
  SELECT
    *
  FROM
    `query-builder-test-project.global._Default._Default` )
SELECT
  http_request.response_size,
  severity
FROM
  scope_query
GROUP BY
  http_request.response_size,
  severity
ORDER BY
  http_request.response_size DESC
LIMIT
  100
```

Expected Results

Row	severity	response_size
	STRING	INTEGER
1	NOTICE	3076
2	DEBUG	3075
3	NOTICE	*Null*

Example 9: Some number of fields with groupings and a COUNT aggregation



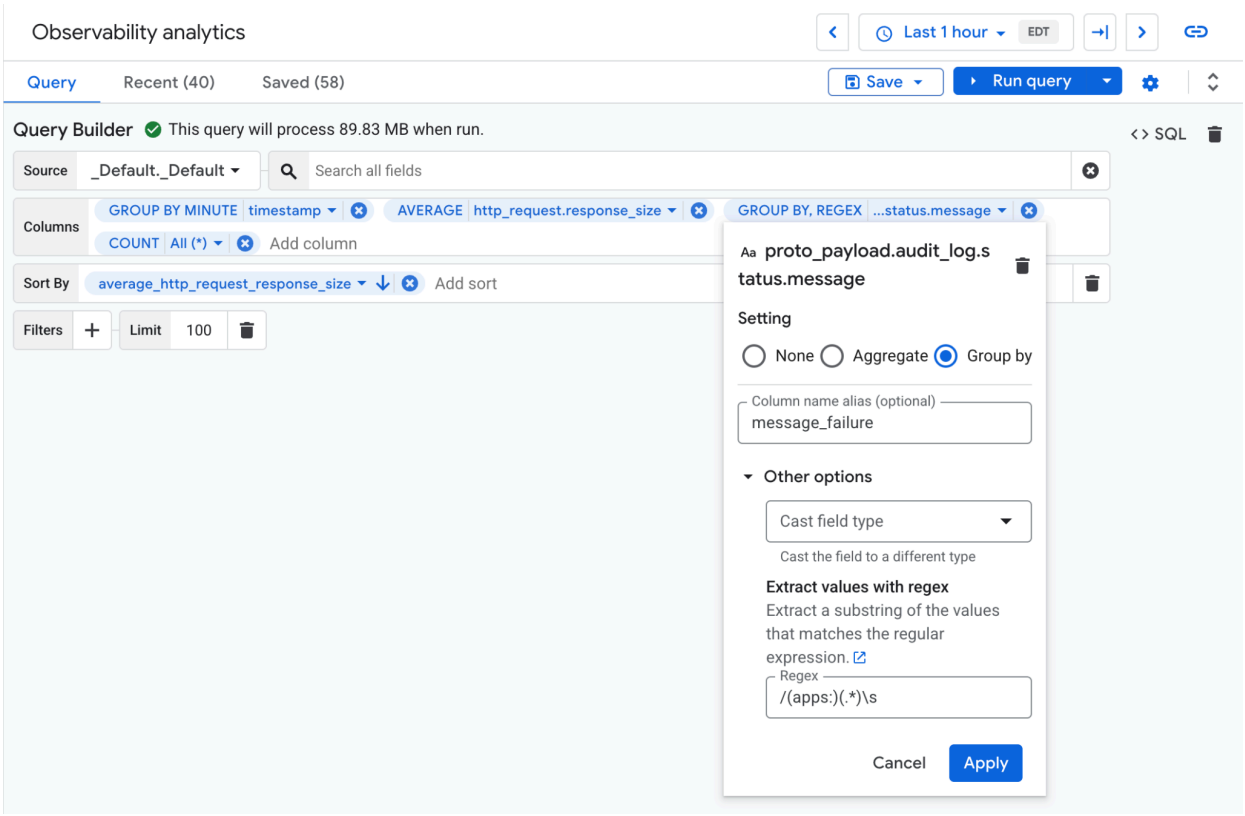
Generated SQL query

```sql
WITH
  scope_query AS (
  SELECT
    *
  FROM
    `query-builder-test-project.global._Default._Default` )
SELECT
  http_request.response_size as response_size,
  severity,
  COUNT( * ) AS count
FROM
  scope_query
GROUP BY
  response_size,
  severity
LIMIT
  100
```

Expected results

Row	severity	response_size	count
	STRING	INTEGER	INTEGER
1	NOTICE	3075	274
2	DEBUG	*Null*	518816
3	WARNING	*Null*	14720

Example 10: Filter and some number of fields with aggregations including aliases, regexp, COUNT, grouping, sort, and search term



Generated SQL query

```SQL
WITH
  scope_query AS (
  SELECT
    *
  FROM
    `query-builder-test-project.global._Default._Default` )
SELECT
  TIMESTAMP_TRUNC( timestamp, MINUTE ) AS minute_timestamp,
  AVG( http_request.response_size ) AS average_http_request_response_size,
  REGEXP_EXTRACT( proto_payload.audit_log.status.message, '/(apps:)(.*)\\s') AS
message_failure,
  COUNT( * ) AS count_all
FROM
  scope_query
GROUP BY
  TIMESTAMP_TRUNC( timestamp, MINUTE ),
  REGEXP_EXTRACT( proto_payload.audit_log.status.message, '/(apps:)(.*)\\s')
ORDER BY
  average_http_request_response_size DESC
```

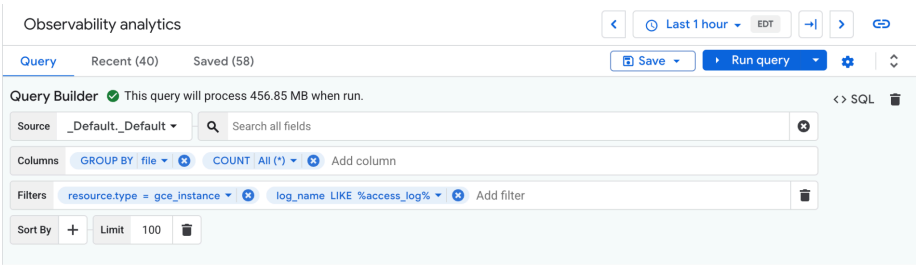
```
LIMIT
  100
```

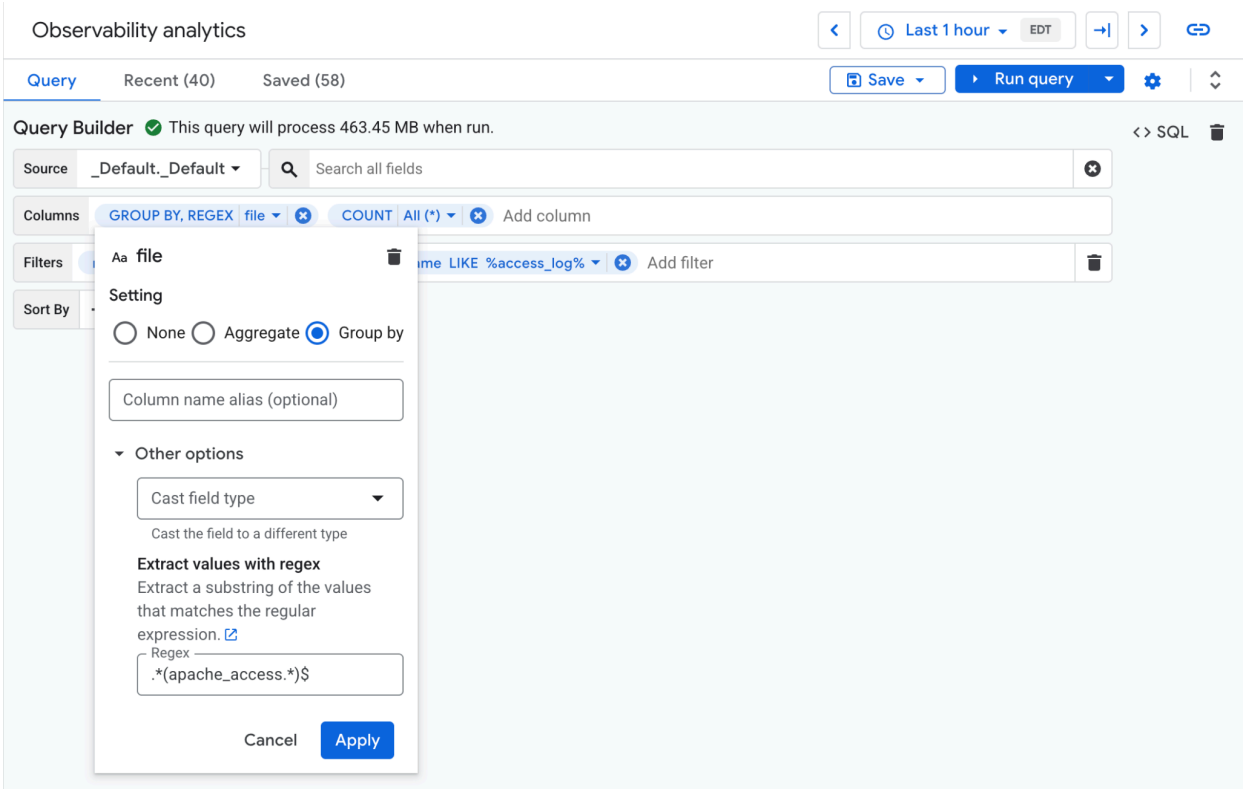
Expected Results

Row	timestamp	response_size	message_failure	count
	TIMESTAMP	FLOAT	STRING	INTEGER
1	2025-05-02 09:00:00.000 EDT	849.8276422764229	app-1-test not found	979292
2	2025-05-02 10:00:00.000 EDT	825.1137295081982	*Null*	978058
3	2025-05-02 02:00:00.000 EDT	814.1943597560974	*Null*	456324

Example 11: Regex extraction - the top 20 files accessed

The following construct shows the top 20 files accessed and shows the use of regex extraction that matches string patterns of “%access_log”.





Generated SQL query

```sql
WITH
  scope_query AS (
  SELECT
    *
  FROM
    `query-builder-test-project.global._Default._Default` )
SELECT
  REGEXP_EXTRACT( json_payload.file, '.*(apache_access.*)$') as file,
  COUNT( * ) AS count_all
FROM
  scope_query
WHERE
  resource.type = 'gce_instance'
  AND log_name LIKE '%access_log%'
GROUP BY
  file
LIMIT
  100
```

Expected result:

Row	file	count
	STRING	INTEGER
1	test_file	12101518

Video Demo: Understand total external traffic by country with VPC flow log

Generated SQL query

```sql
WITH
  scope_query AS (
  SELECT
    *
  FROM
    `test-project.global._Default._Default` )
SELECT
  JSON_VALUE( json_payload.dest_location.country ) AS country,
  SUM( CAST( JSON_VALUE( json_payload.bytes_sent ) AS INT64 ) ) AS
total_bytes_sent,
  SUM( CAST( JSON_VALUE( json_payload.packets_sent ) AS INT64 ) ) AS
total_packets_sent,
  AVG( CAST( JSON_VALUE( json_payload.rtt_msec ) AS INT64 ) ) AS avg_rtt_msec
FROM
  scope_query
WHERE
  log_id = 'compute.googleapis.com/vpc_flows'
  AND JSON_VALUE( json_payload.reporter ) = 'SRC'
  AND JSON_VALUE( json_payload.dest_location.country ) IS NOT NULL
GROUP BY
  JSON_VALUE( json_payload.dest_location.country )
```

```
ORDER BY
  total_bytes_sent DESC
LIMIT
  100
```

Supported Comparison Operators

- =
- !=
- = regex
- !=regex
- >
- >=
- <
- <=
- IS NULL
- IS NOT NULL
- LIKE

To be scheduled

- Support Log scope contain cross project views
- Save query from query builder
- Trace support in Analytics

FAQ

Can I use log scope?

Yes. Log scope is available in the query builder. If you need to write more complex query to the same log scope, you can edit your own query by using the log scope CTE generated by the query builder.

Can I save the query builder construct?

Not yet. Save from query builder is on the roadmap and anticipate to come out in later 2026.

Can I join tables?

No. Join is not supported. But you can use the SQL editor to write SQL.

Does it support “two way mapping” if I write a SQL query?

No. Query builder will auto generate SQL query for valid construct, but it does not support translating your query into builder.