TONalytica: User Guide

v1.1

This tutorial has been funded by TON Footstep #287
Send comments, suggestions, and other inputs to tonalytica@devnull.ae

Disclaimer

Tonalytica is a data analytics platform based on Redash, an open-source business intelligence tool. While Tonalytica leverages the core functionalities and features of Redash to provide a powerful data analysis solution, it is essential to note that Tonalytica is an independent product developed and maintained by our team at DevNull.

Although we are committed to delivering a seamless and efficient user experience with Tonalytica, we want to make it clear that we are not directly affiliated with the creators or maintainers of Redash. Any reference to Redash within the context of Tonalytica is solely to acknowledge the technological foundation upon which our platform is built.

The use of the Redash name, logo, or any associated trademarks is done with the acknowledgment that Redash is a distinct and separate entity from Tonalytica. Redash does not endorse or have any involvement in the development, promotion, or support of Tonalytica.

We extend our appreciation to the Redash community for providing an exceptional open-source project that has been instrumental in the creation of Tonalytica. Our team is dedicated to continually enhancing Tonalytica's capabilities and ensuring that it remains a reliable, user-friendly, and feature-rich data analytics solution for our valued users.

For any inquiries, support, or feedback related to Tonalytica, please reach out to our team through the provided contact channels. We welcome your input and look forward to serving you with the utmost dedication.

Quickstart

This guide will help you get started on TONalytica in minutes. TONalytica has many more features, but these are the basics you'll need to get started. If you're looking for developers guides, check out the documentation.

Prerequisites:
- You'll need to have a TONalytica account to follow along. If you don't have one, you can sign up here. Authorisation exclusively via GitHub.
- We also recommend you have a basic understanding of SQL and Blockchain concepts.

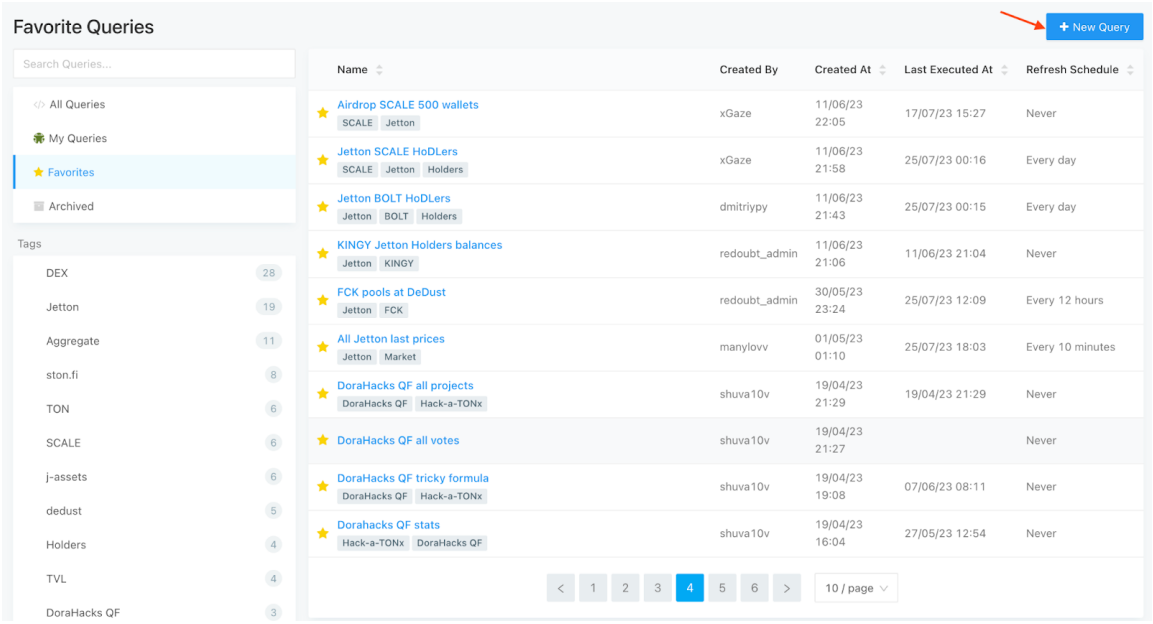
1. Data Sources

The first thing you'll want to do is connect a data source. We've already taken care of that, and you have access to the following databases:

- Main TON DB (accounts, transactions, Jettons, NFTs, pools etc.)
- VENOM (https://venomics.xyz/, testnet)
- Archive Full-Node (available in paid version only)

2. Write A Query

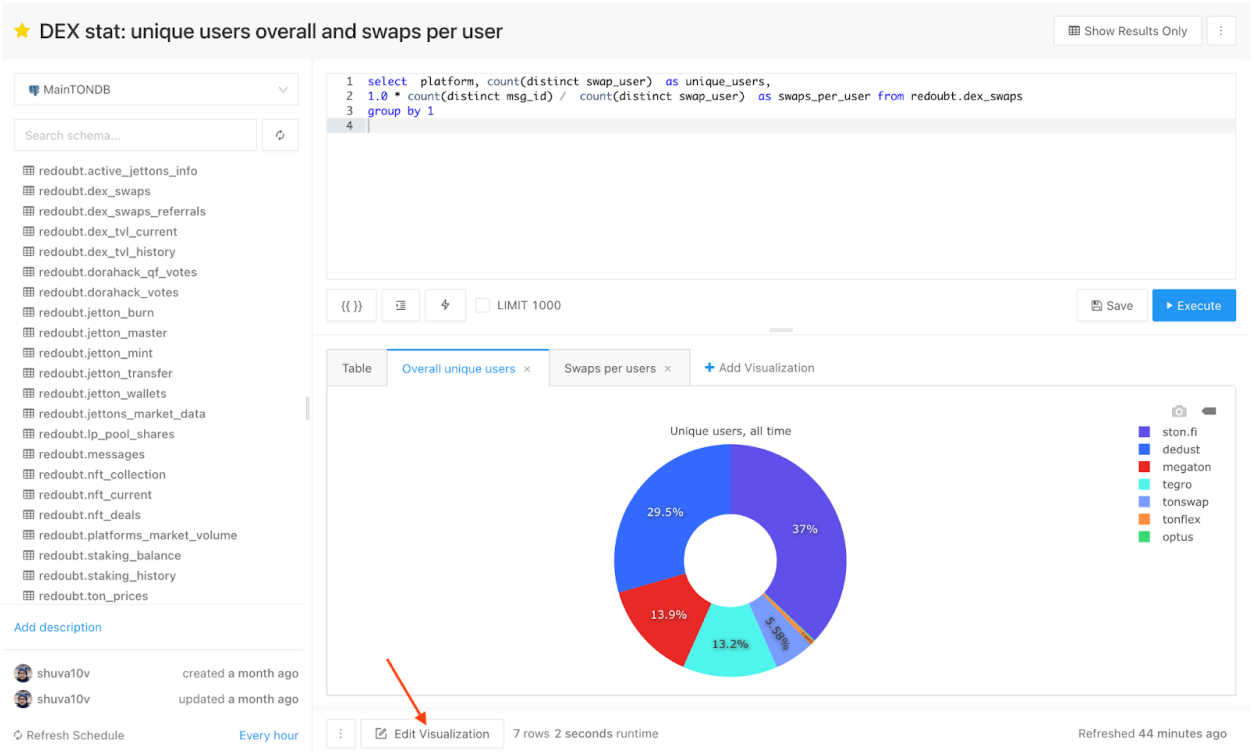
Once you've connected a data source, it's time to write a query: click on "Create" in the navigation bar, and then choose "Query". See the "Writing Queries" page for detailed instructions on how to write queries.



3. Add Visualizations

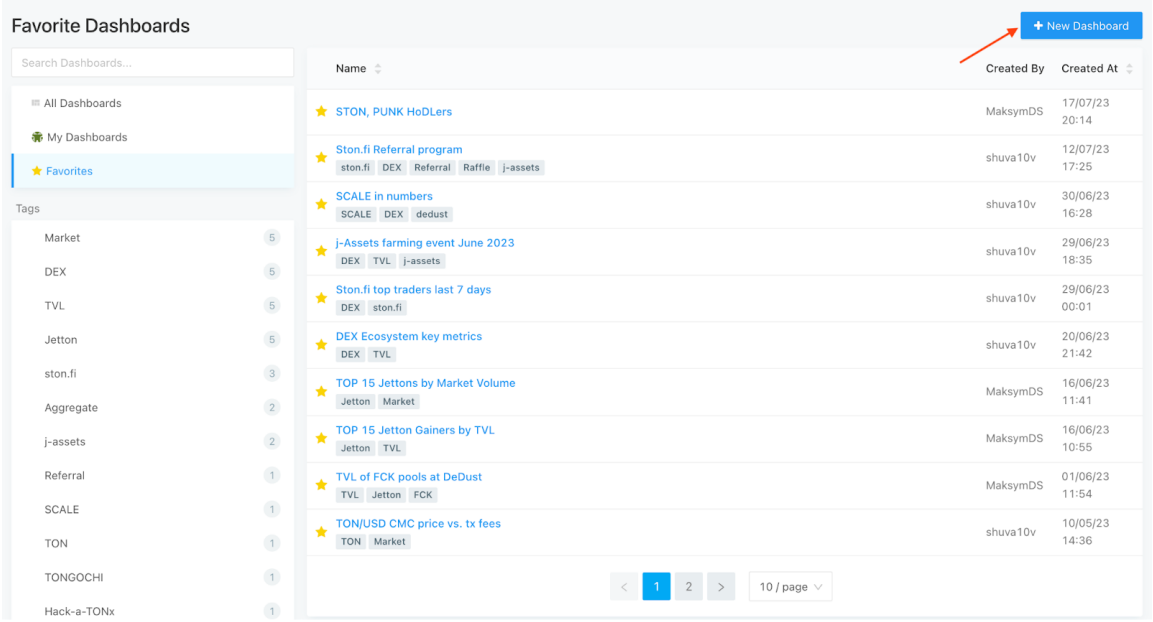
By default, your query results (data) will appear in a simple table. Visualizations are much better to help you digest complex information, so let's visualize your data. TONalytica supports multiple types of visualizations so you should find one that suits your needs.

Click the "New Visualization" button just above the results to select the perfect visualization for your needs. You can view more detailed instructions here.



4. Create A Dashboard

You can combine visualizations and text into thematic & powerful dashboards. Add a new dashboard by clicking on "Create" in the navigation bar, and then choose "Dashboard". Dashboards are visible for your team members or they can be shared publicly. For more details, click here.



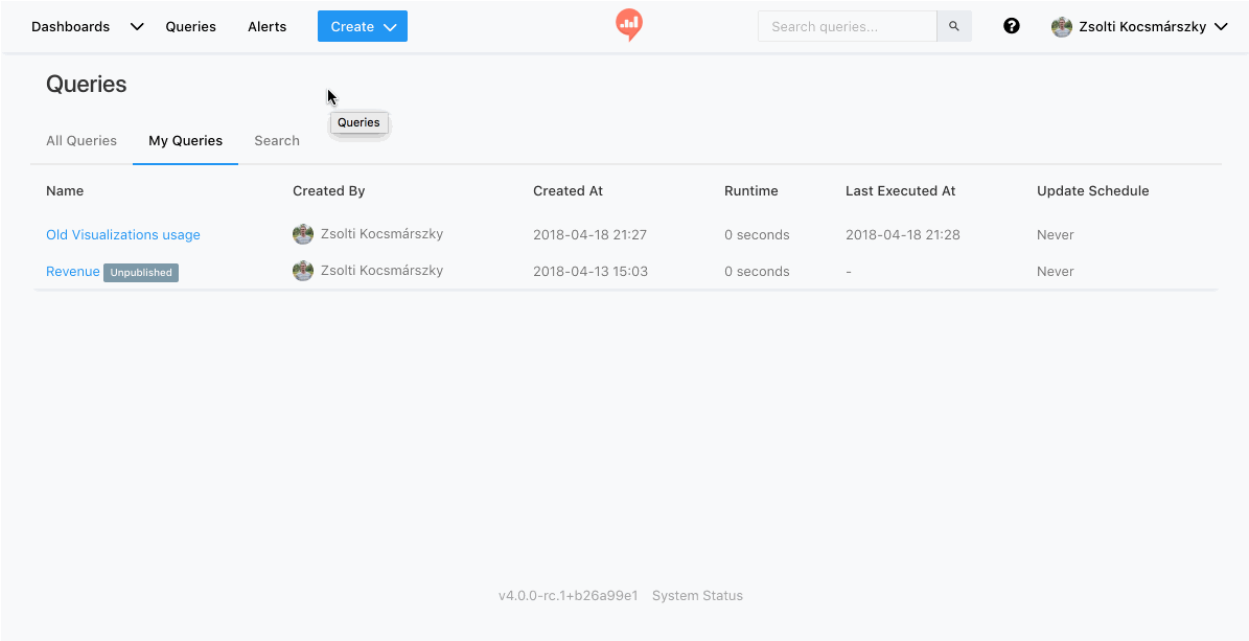
5. Invite Colleagues

TONalytica is better together.

To start enjoying the collaborative nature of TONalytica you'll want to invite your team!

Creating and Editing Queries

The Query Editor is where you can write SQL queries to explore blockchain data. To make a new query, click `Create` in the navbar then select `Query`.



Query Syntax

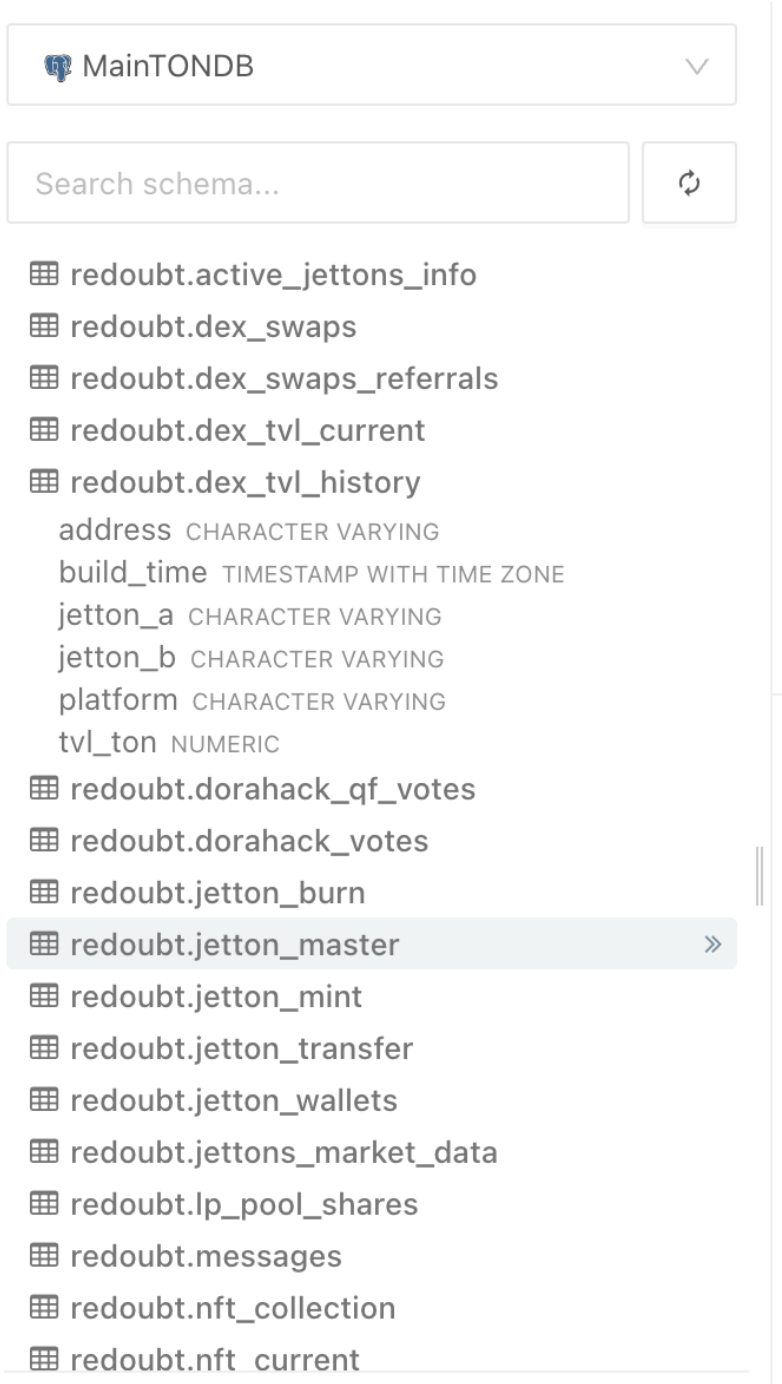
In most cases we use the query language native to the data source. In our case is SQL.

Keyboard Shortcuts

- Execute query: `Ctrl`/`Cmd` + `Enter`
- Save query: `Ctrl`/`Cmd` + `S`
- Toggle Auto Complete: `Ctrl` + `Space`
- Toggle Schema Browser `Alt`/`Option` + `D`

Schema Browser

To the left of the query editor, you will find the Schema Browser:



The schema browser will list all your tables, and when clicking on a table will show its columns. To insert an item into your query, simply click the double arrow on the right side. You can filter the schema with the search box and refresh it by clicking on the refresh button (otherwise it refreshes periodically in the background).

Auto Complete

The query editor also includes an Auto Complete feature that makes writing complicated queries easier. Live Auto Complete is on by default. So you will see table and column suggestions as you type. You can disable Live Auto Complete by clicking the lightning bolt icon beneath the query editor. When Live Auto Complete is disabled, you can still activate Auto Complete by hitting `CTRL` + `Space`.

Live Auto Complete is enabled by default unless your database schema exceeds five thousand tokens (tables or columns). In such cases, you can manually trigger Auto Complete using the keyboard shortcut.

Auto Complete looks for schema tokens, query syntax identifiers (like `SELECT` or `JOIN`) and the titles of Query Snippets

Query Settings

Published vs Unpublished Queries

By default each query starts as an unpublished draft named New Query. It can't be included on dashboards or used with alerts.

To publish a query, change its name or click the `Publish` button. You can toggle the published status by clicking the `Unpublish` button. Unpublishing a query will not remove it from dashboards or alerts. But it will prevent you from adding it to any others.

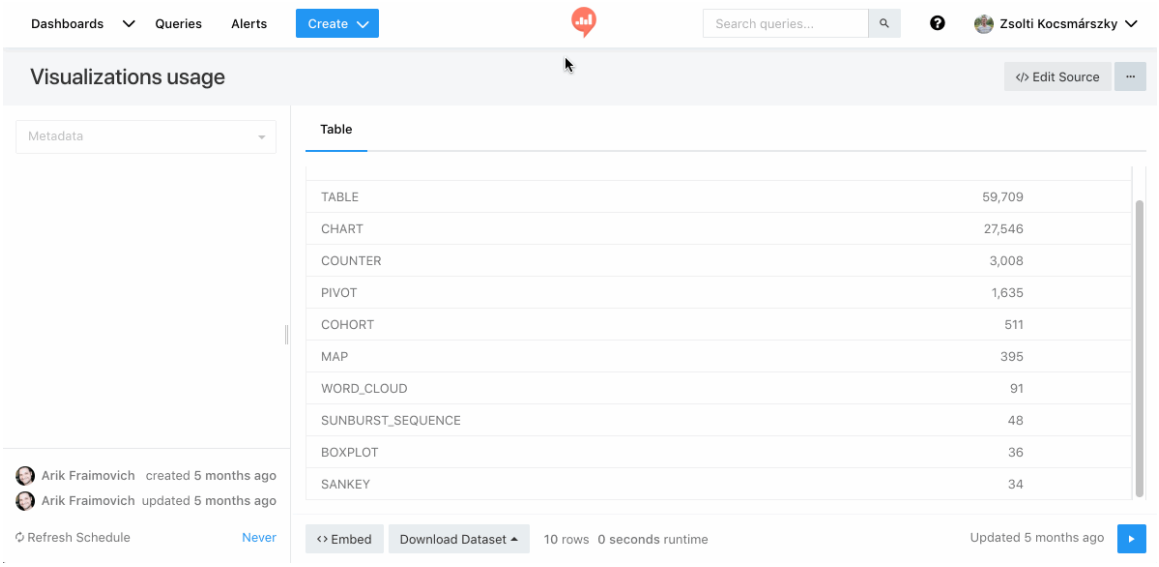
Publishing or un-publishing a query does not affect its visibility. All queries in your organization are visible to all logged-in users.

Archiving a Query

You can't delete queries in TONalytica. But you can archive them. Archiving is like deleting, except direct links to the query still work. To archive a query, open the vertical ellipsis menu at the top-right of the query editor and click Archive.

Duplicating (Forking) a Query

If you need to create a copy of an existing query (created by you or someone else), you can fork it. To fork a query, just click on the Fork button (see example below)



Query Snippets

Copy and Paste are a big part of composing database queries. Because it's much easier to duplicate prior work than to write it from scratch. This is particularly true for common `JOIN` statements or complex `CASE` expressions. As your list of queries in TONalytica grows, however, it can be difficult to remember which queries contain the statement you need right now. Enter Query Snippets.

Insertion Points

`${1:table}` is an insertion point with placeholder text. When TONalytica renders the snippet, the dollar sign `$` and curly braces `{}` will be stripped away and the word `table` will be highlighted for the user to replace.

You can use the placeholder text as a desirable default value for the user to override at runtime.

You designate insertion points by wrapping an integer tab order with a single dollar sign and curly braces `${}`. A text placeholder preceded by a colon `:` is optional but useful for users unfamiliar with your snippet.

An insertion point of zero `${0}` is always the *last* point in the tab order.

Insert A Query Snippet

If you have Live Auto Complete enabled, you can invoke your snippet from the Query Editor by typing the trigger word you defined in the Query Snippet editor. Auto Complete will suggest it like any other keyword in your database.

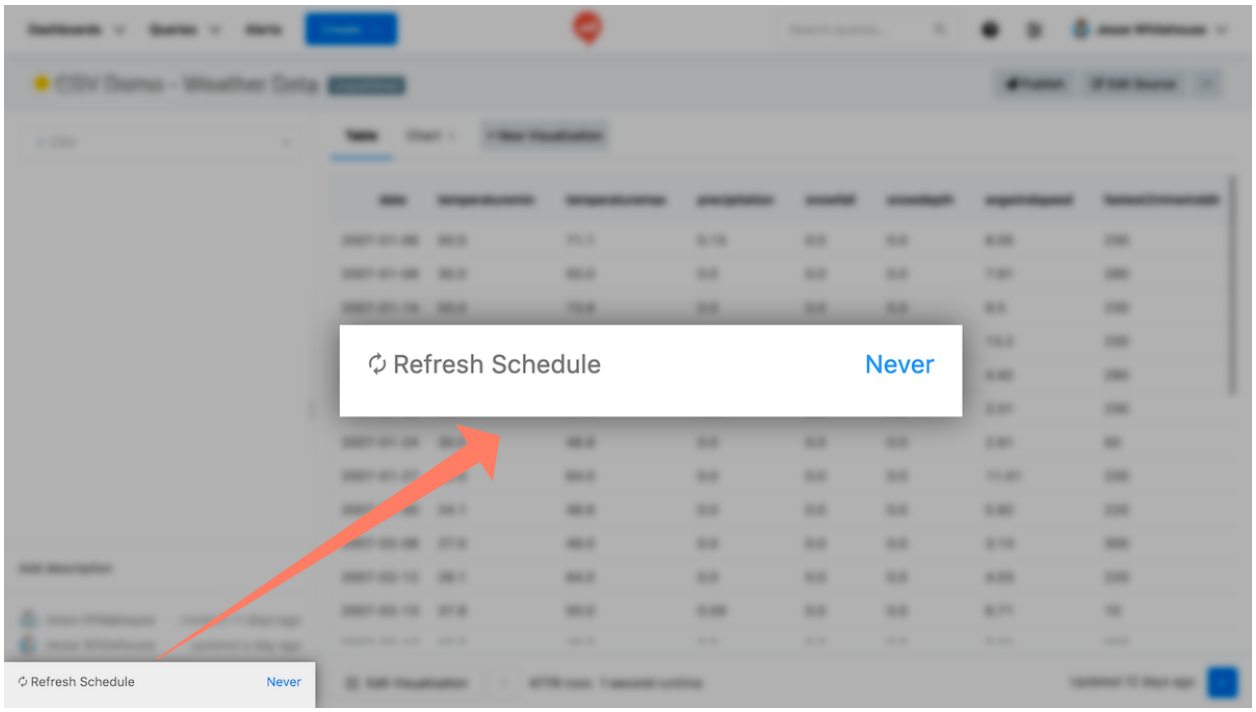
If Live Auto Complete is disabled, you can still invoke Query Snippets by pressing `CTRL + Space` and typing the trigger word for your Query Snippet. This can be necessary if your schema exceeds 5000 tokens.

Here are some other ideas for snippets:

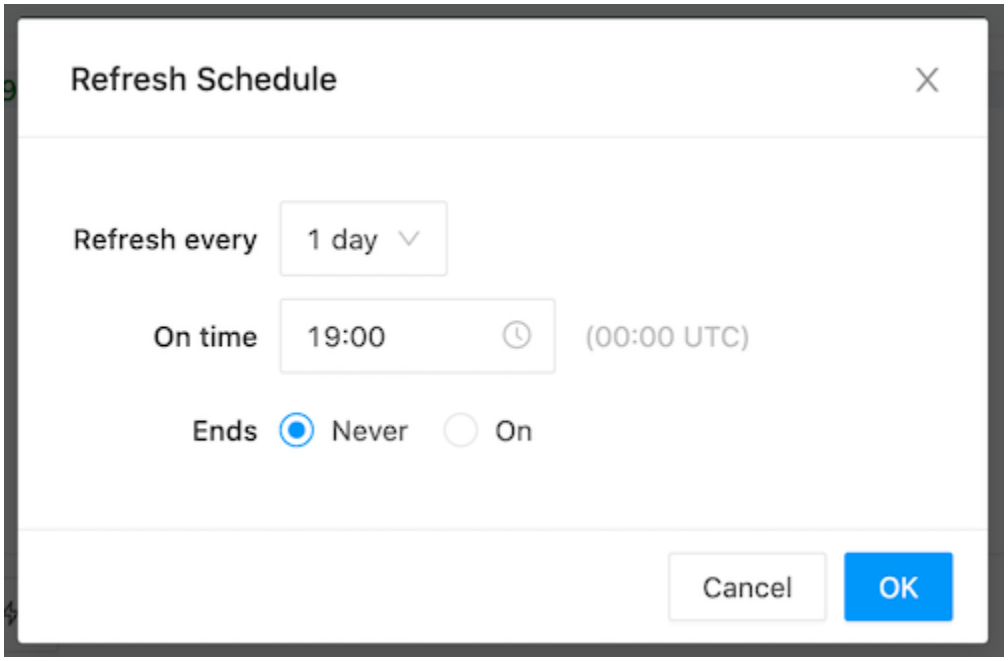
- Frequent `JOIN` statements
- Complicated clauses like `WITH` or `CASE`.

How to schedule a query

You can use scheduled query executions to keep your dashboards updated or to power routine Alerts. By default, your queries will not have a schedule. But this is easy to adjust. In the bottom left corner of the query editor you'll see the schedule area:



Clicking Never will open a picker with allowed schedule intervals.



Your query will run automatically once a schedule is set.

When you schedule queries to run at a certain time-of-day, TONalytica converts your selection to UTC using your computer's local timezone. That means if you want a query to run at a certain time in UTC, you need to adjust the picker by your local offset.

For example, if you want a query to execute at `00:00` UTC each day but your current timezone is CDT (UTC-5), you should enter `19:00` into the scheduler. The UTC value is displayed to the right of your selection to help confirm your math.

Scheduling queries that use parameters is not currently supported. You can use the TONalytica API and a scheduling system like CRON instead.

Scheduled Query Failure Reports

TONalytica added the ability to email query owners once per hour if one or more queries failed. These emails continue until there are no more failures. Failure report emails run on an independent process from the actual query schedules. It may take up to an hour after a failed query execution before TONalytica sends the failure report.

Visualization Types

TONalytica supports several different types of visualizations - A Table is the default view.

Boxplot

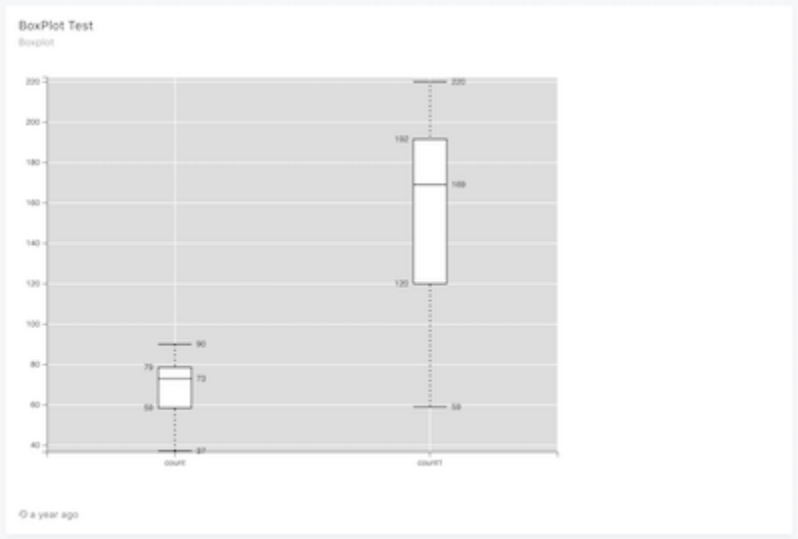


Chart - Line, Bar, Area, Pie, Scatter


Queries over Time
80k
60k
40k
20k
0
Jul 2014
Jan 2015
Jul 2015
Jan 2016
Jul 2016
Jan 2017
Jul 2017
Jan 2018
6 days ago

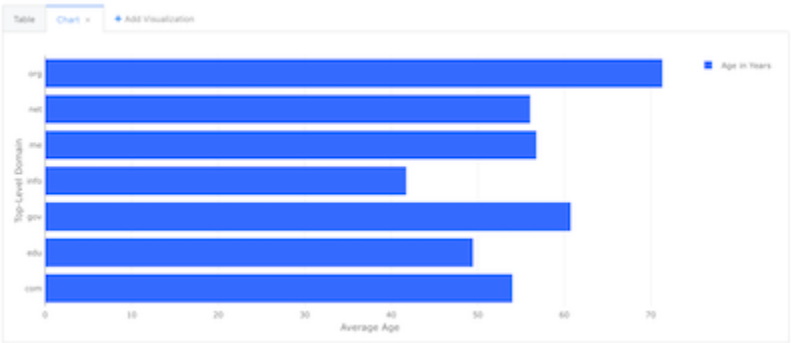
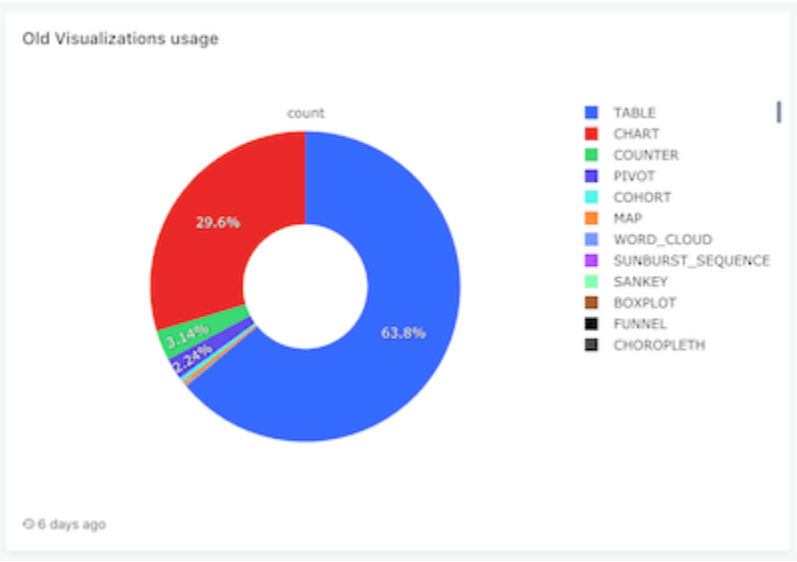
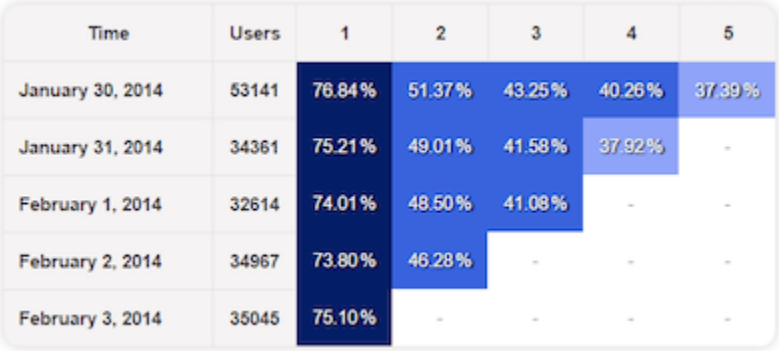


Table
Chart
Add Visualization
Age in Years
org
net
me
info
gov
edu
com
Top-Level Domain
0
10
20
30
40
50
60
70
Average Age


Old Visualizations usage
count
29.6%
3.14%
2.24%
63.8%
TABLE
CHART
COUNTER
PIVOT
COHORT
MAP
WORD_CLOUD
SUNBURST_SEQUENCE
SANKEY
BOXPLOT
FUNNEL
CHOROPLETH
6 days ago

Cohort



Time	Users	1	2	3	4	5
January 30, 2014	53141	76.84%	51.37%	43.25%	40.26%	37.39%
January 31, 2014	34361	75.21%	49.01%	41.58%	37.92%	-
February 1, 2014	32614	74.01%	48.50%	41.08%	-	-
February 2, 2014	34967	73.80%	46.28%	-	-	-
February 3, 2014	35045	75.10%	-	-	-	-

A cohort analysis examines the outcomes of predetermined groups, called cohorts, as they progress through a set of stages. The signature characteristic of a cohort chart is its comparison of the change in a variable across two different time series. For example, a common cohort definition is users by sign-up period and their usage pattern by day. Other examples include:

- Monthly hard drive failure statistics by month
- Weekly supplier delivery performance by week
- Monthly average class GPA's by month

While there are many ways to define the stages of a Cohort analysis, TONalytica's supports Cohorts visualizations with daily, weekly, or monthly stages. Also, TONalytica's cohort charts compare a cohort's measurements in a given period against that group's initial population size.

Date Format

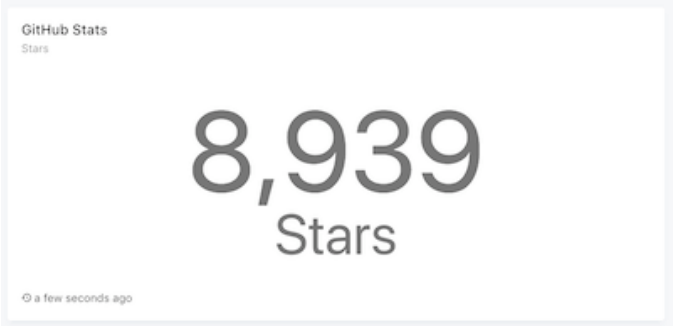
TONalytica expects your input samples to take the following format:

Cohort Date	Period	Count Satisfying Target	Total Cohort Size

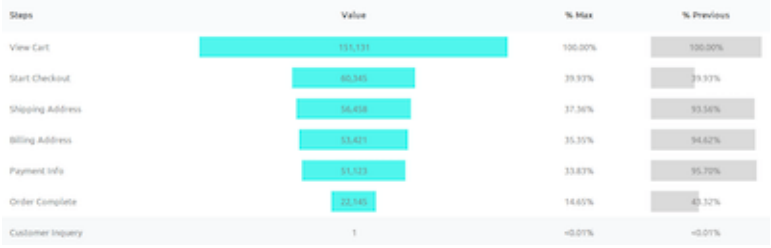
- Cohort Date is the date that uniquely identifies a cohort. Suppose you're visualizing monthly user activity by sign-up date, your cohort date for all users that signed-up in January 2018 would be January 1st, 2018. The cohort date for any user that signed-up in February would be February 1st, 2018.
- Period is a count of how many periods transpired since the cohort date as of this sample. If you are grouping users by sign-up month, then your period will be the count of months since these users signed up. In the above example, a measurement of activity in July for users that signed up in January would yield a period value of 7 because seven periods have transpired between January and July.

- Count Satisfying Target is your actual measurement of this cohort's performance in the given period. In the above example, if thirty users who signed up in January showed activity in July then the Count Satisfying Target would be 30.
- Total Cohort Size is the denominator that TONalytica will use to calculate the percentage of a cohort's target satisfaction for a given period. Continuing the example above, if seventy-two users signed up in January then the Total Cohort Size would be 72. When the visualization is rendered, TONalytica would display the value as 41.67% (32 ÷ 72).

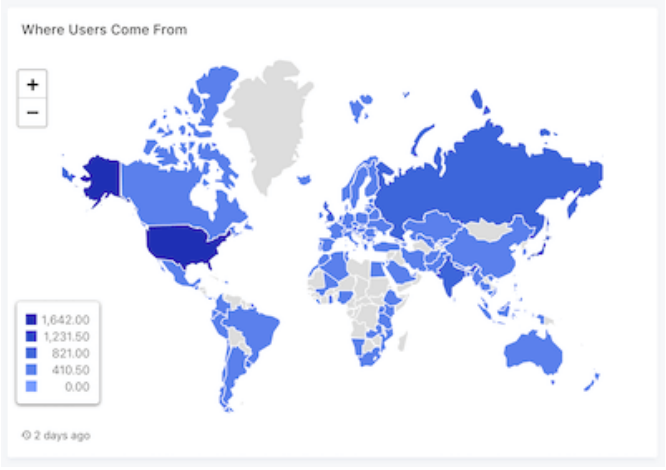
Counter



Funnel



Map



Pivot Table

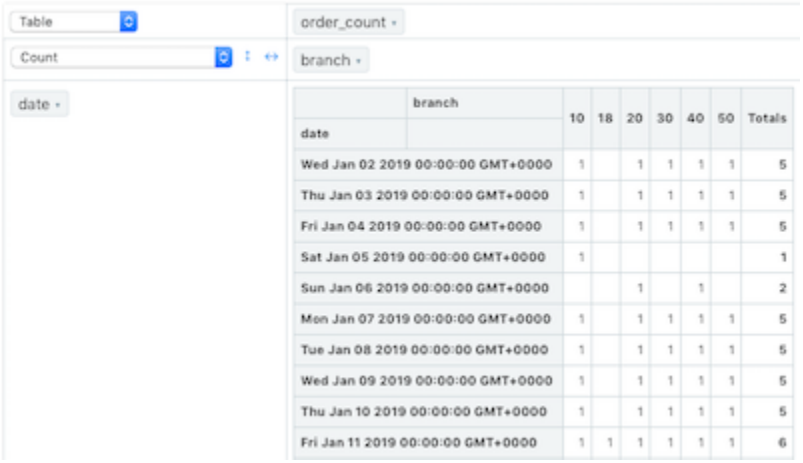
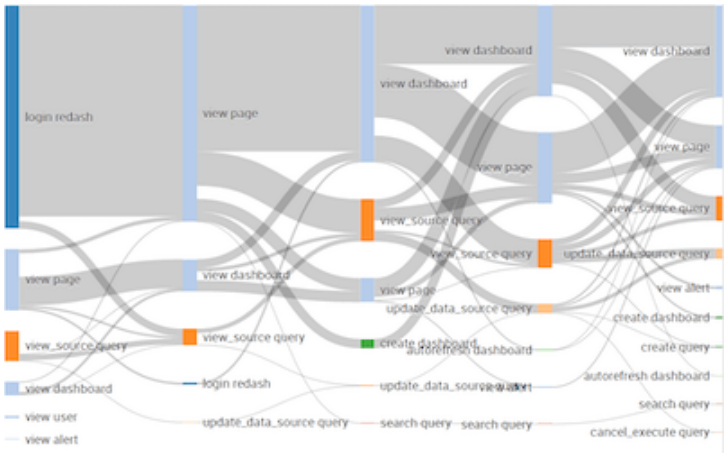


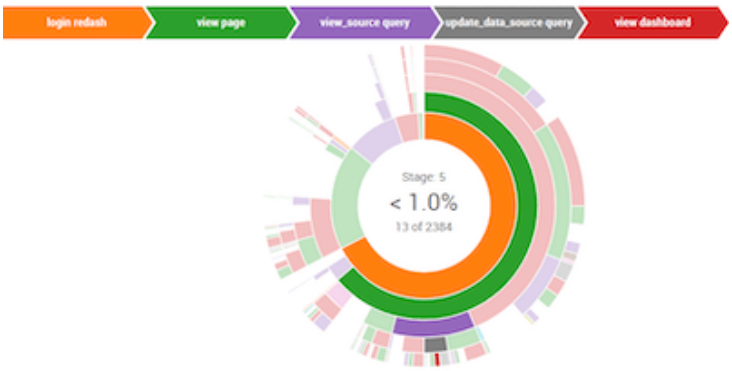
Table | Count | date | order_count | branch

date	branch	10	18	20	30	40	50	Totals
Wed Jan 02 2019 00:00:00 GMT+0000		1		1	1	1	1	5
Thu Jan 03 2019 00:00:00 GMT+0000		1		1	1	1	1	5
Fri Jan 04 2019 00:00:00 GMT+0000		1		1	1	1	1	5
Sat Jan 05 2019 00:00:00 GMT+0000		1						1
Sun Jan 06 2019 00:00:00 GMT+0000				1		1		2
Mon Jan 07 2019 00:00:00 GMT+0000		1		1	1	1	1	5
Tue Jan 08 2019 00:00:00 GMT+0000		1		1	1	1	1	5
Wed Jan 09 2019 00:00:00 GMT+0000		1		1	1	1	1	5
Thu Jan 10 2019 00:00:00 GMT+0000		1		1	1	1	1	5
Fri Jan 11 2019 00:00:00 GMT+0000		1	1	1	1	1	1	6

Sankey



Sunburst

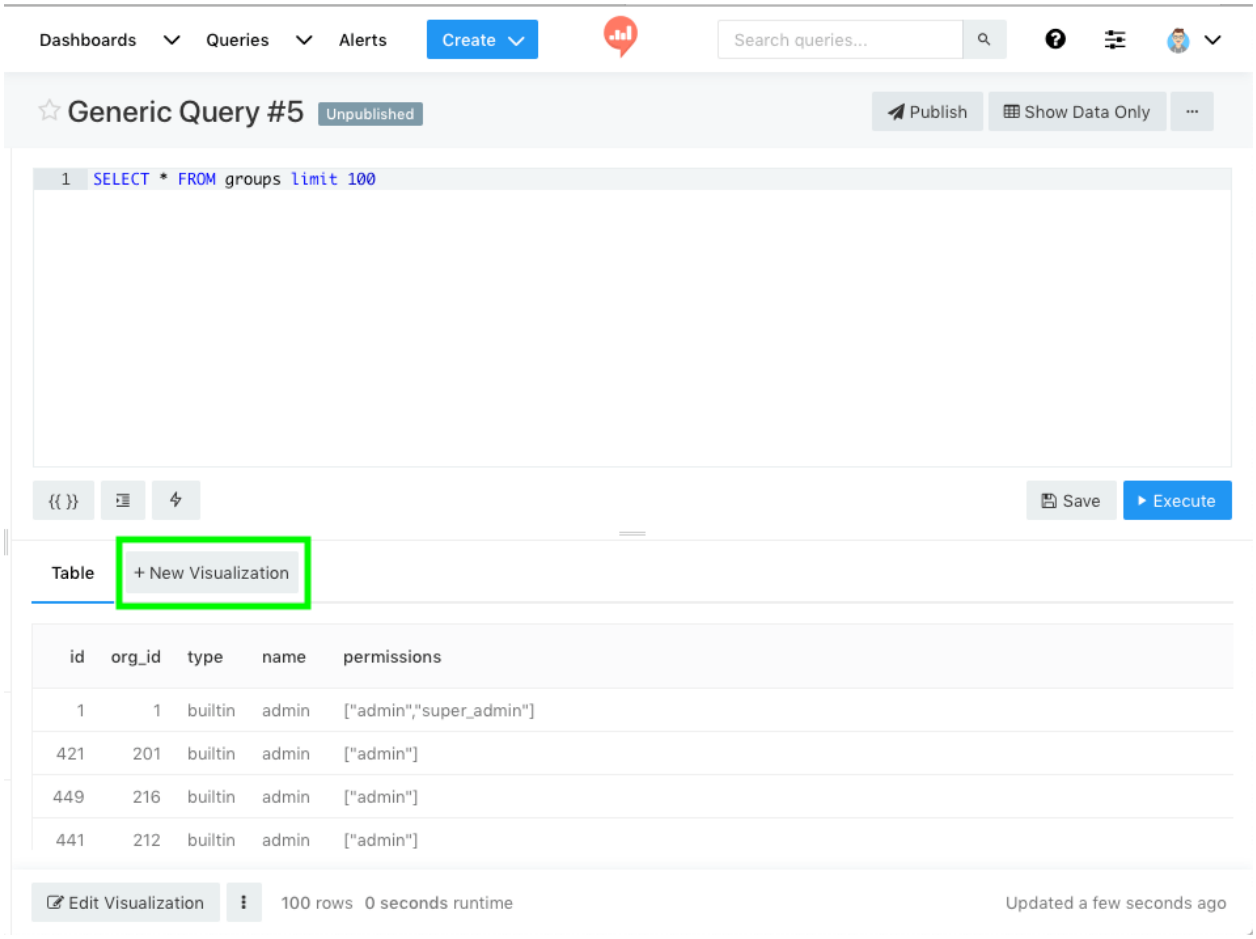


Visualizations How To

Visualizations make your data come to life! You can create Visualizations from any results of a query.

Create a New Visualization

Once your query has finished running for the first time, you can add a visualization by clicking the “New Visualization” button above the results table.

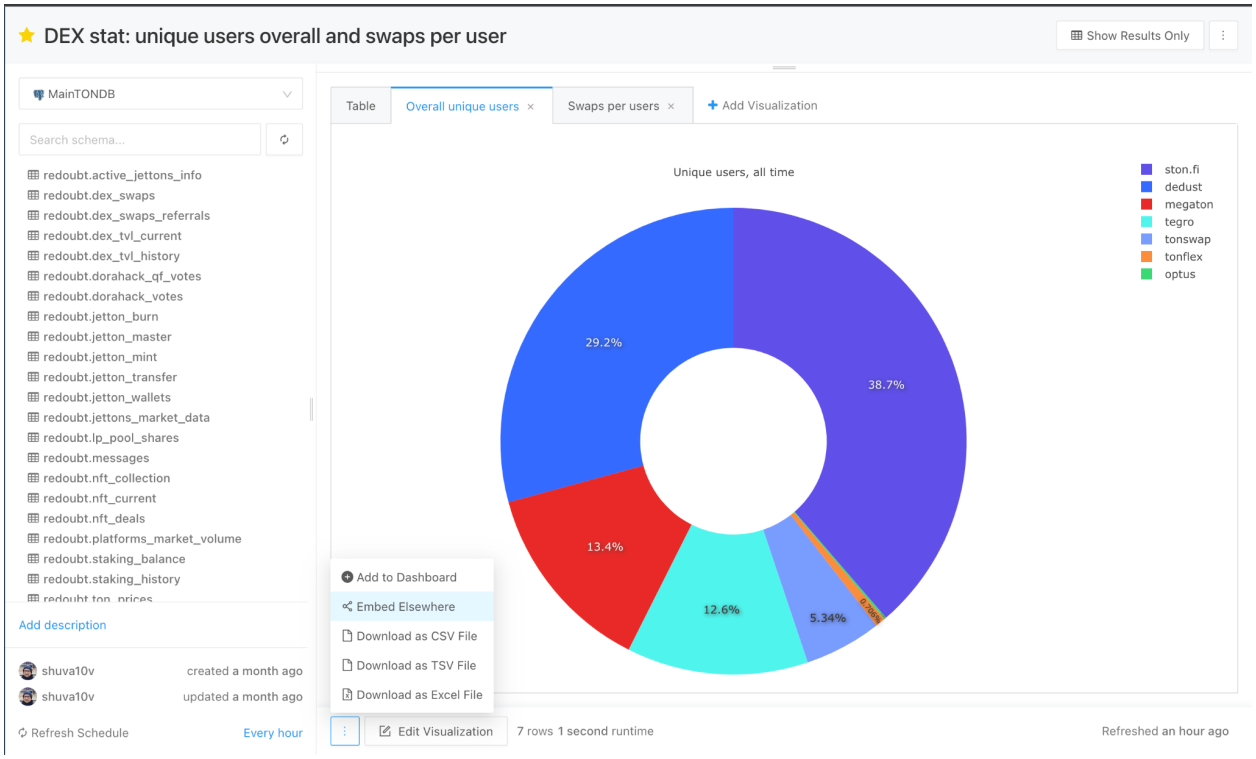


Edit A Visualization

You can modify the settings of an existing visualization from the query editor screen. Click the visualization on the tab bar and you’ll see an `Edit Visualization` option beneath each visualization. Clicking it will open the current settings for that visualization (type, X axis, Y axis, groupings etc.). Hit “Save” to apply your changes or “Cancel” to leave no trace.

Embedding Visualizations

It's easy to embed TONalytica visualizations. Just click the elipsis button beneath any visualization to show further options and select `Embed Elsewhere`.



This will pop up the `<iframe>` code you can drop into your HTML pages.

> Queries with text-type parameters do not support embeds.

Query String Variables for Embeds

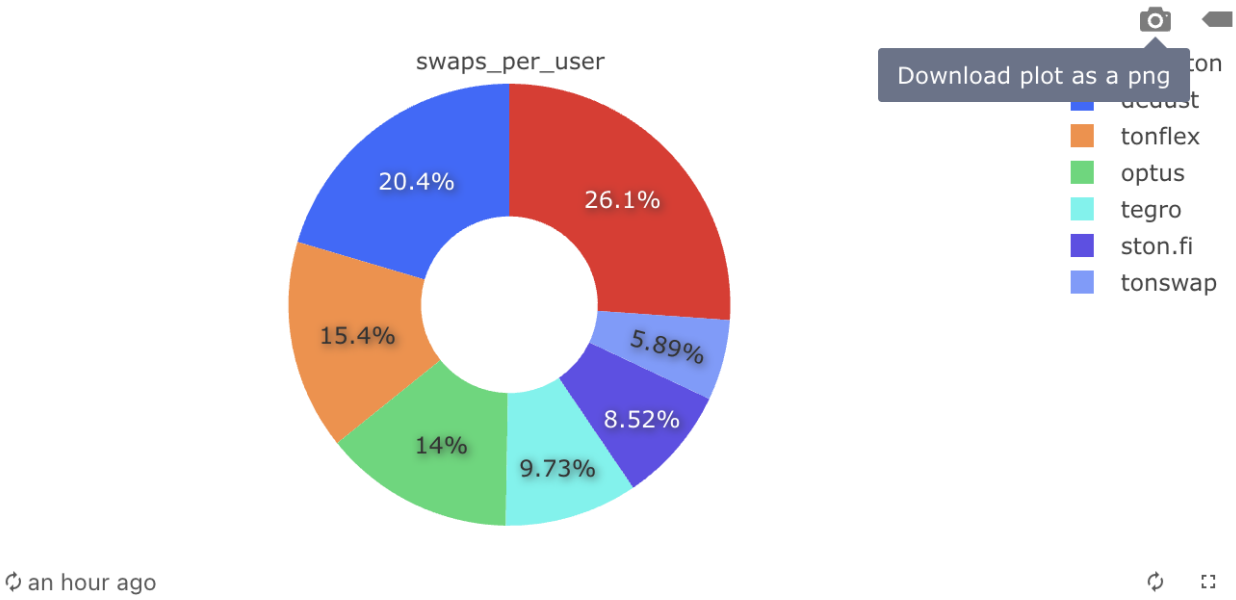
You can append query string variables to your embed URLs:

- `?hide_parameters` hides any parameter selection widgets
- `?hide_header` hides the branded TONalytica header and query title
- `?hide_link` hides the link back to TONalytica
- `?hide_timestamp` hides the timestamp

Downloading A Visualization as an Image File

For chart visualizations, you can also download a local image file. Just hover your mouse near the top right area of the visualization and click the camera icon that appears. A PNG will be downloaded to your device.

Swaps per users – DEX stat: unique users overall and swaps per user



an hour ago

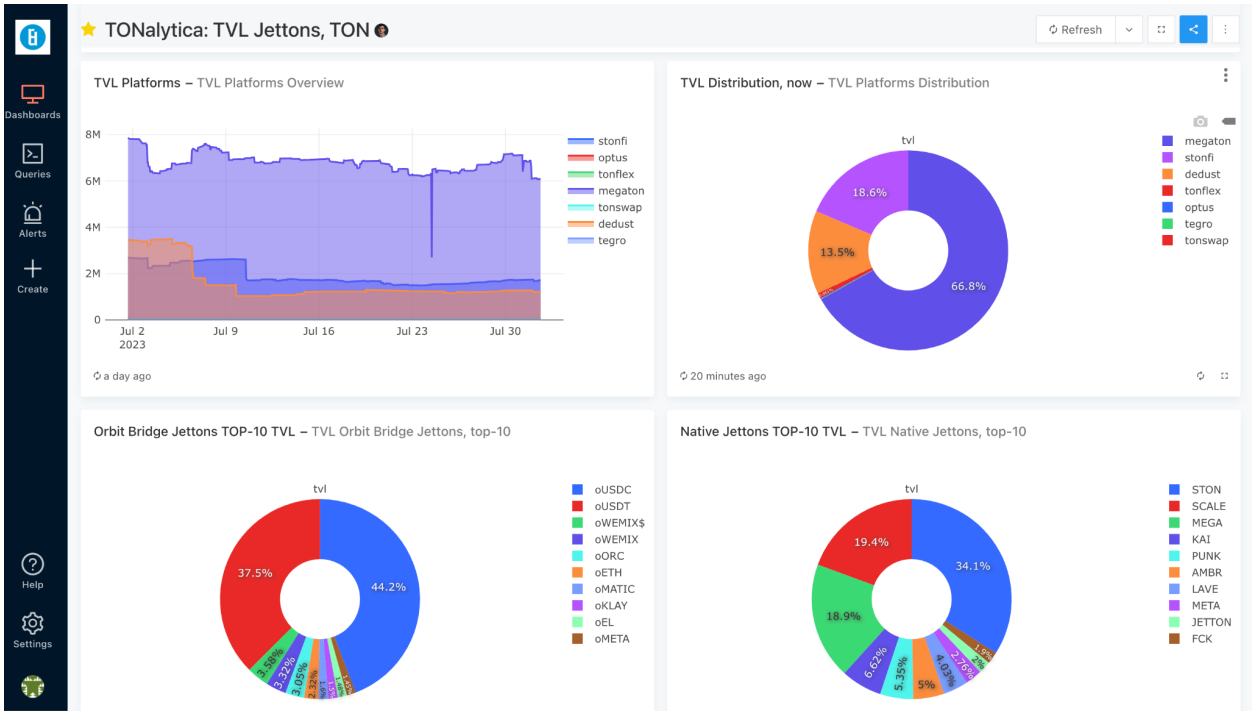
Creating and Editing Dashboards

Dashboards on TONalytica consist of widgets. Widgets can either be Visualizations or Text. It is also possible to embed images or GIFs inside of the text widget.

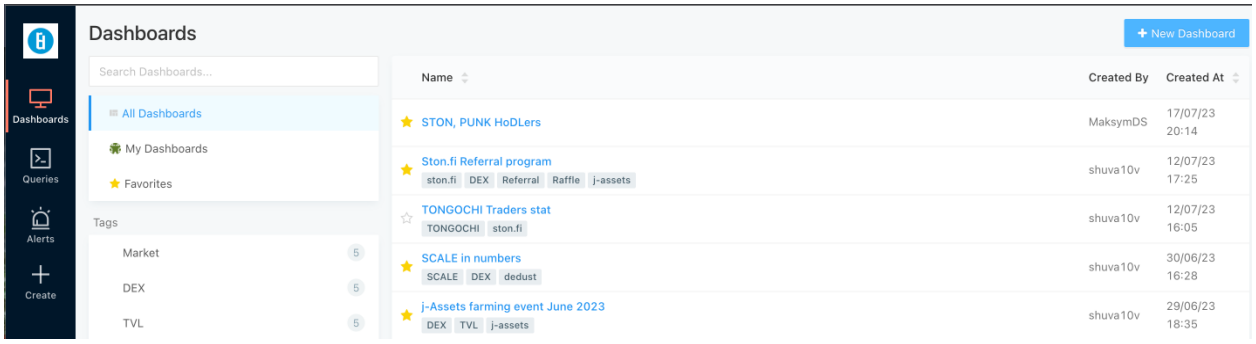
You can freely resize every widget to match the layout you want to create.

Creating a Dashboard

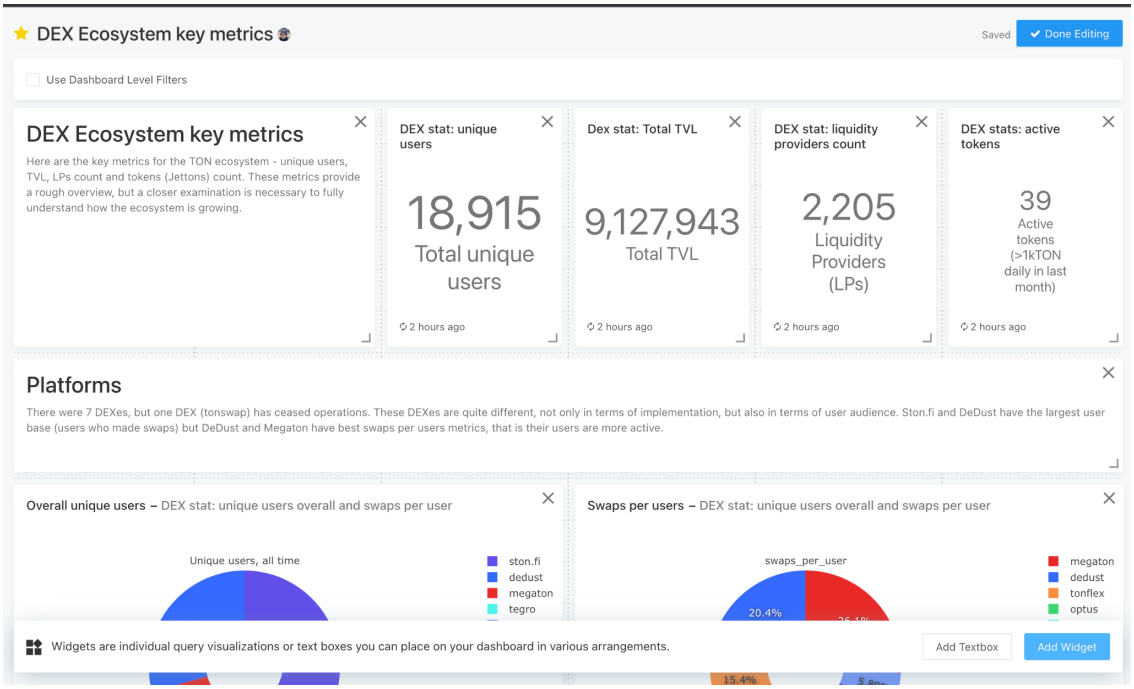
A dashboard lets you combine visualizations and text boxes that provide context with your data.



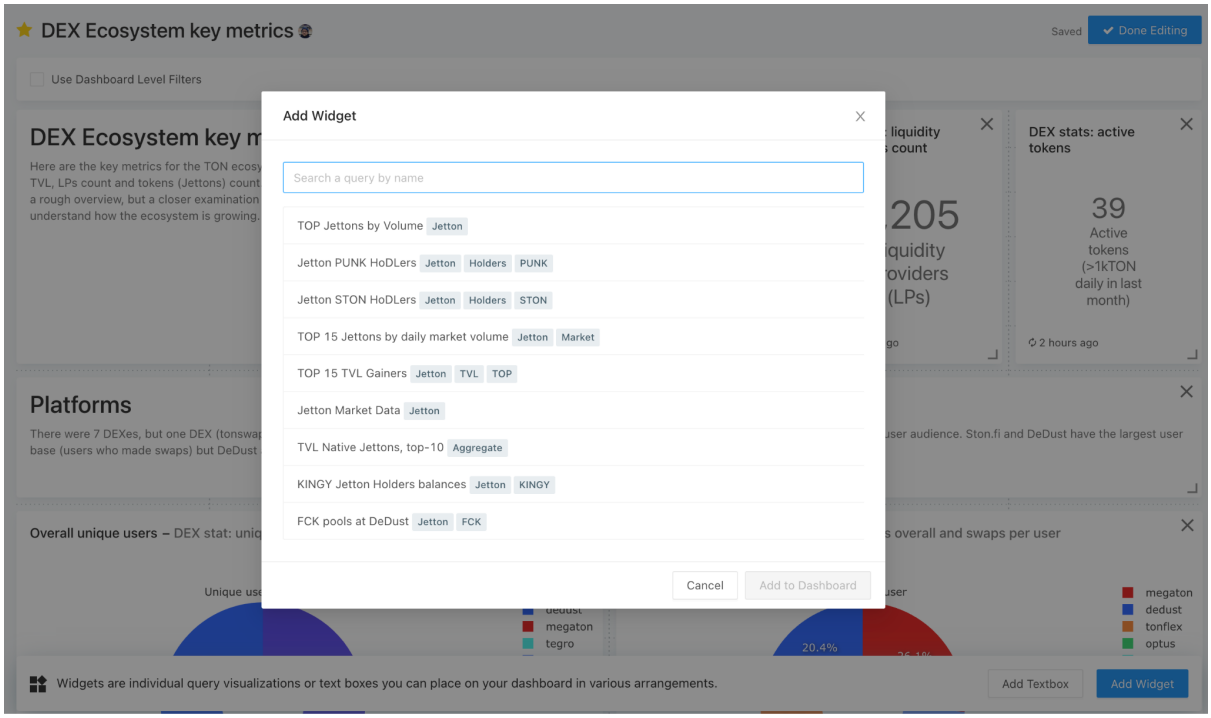
You can create a new dashboard with the Create button from the main navigation menu:



After naming your dashboard, you can add widgets from existing query visualizations or by writing commentary with a text box. Start by clicking the Add Widget button.



Search existing queries or pick a recent one from the pre-populated list:



Dashboard URLs

When you create a dashboard, TONalytica automatically assigns it an `id` number and a URL `slug`. The slug is based on the name of the dashboard. For example a dashboard named “Account Overview” could have this URL:

`https://tonalytica.redoubt.online/dashboards/251-account-overview`

If you change the dashboard name to “Account Over (Old)”, the URL will update to:

`https://tonalytica.redoubt.online/dashboards/251-account-overview-old-`

The dashboard can also be reached using the `/dashboard` endpoint (notice this is singular), which accepts *either* an ID or a slug:

- `https://tonalytica.redoubt.online/dashboard/251`
- `https://tonalytica.redoubt.online/dashboard/account-overview`

Dashboard ids are guaranteed to be unique. But multiple dashboards may use the same name (and therefore `slug`). If a user visits `/dashboard/account-overview` and more than one dashboard exists with that slug, they will be redirected to the earliest created dashboard with that slug.

Picking Visualizations

By default, query results are shown in a table. At the moment it's not possible to create a new visualization from the "Add Widget" menu, so you'll need to open the query and add the visualization there beforehand (instructions).

Adding Text Boxes

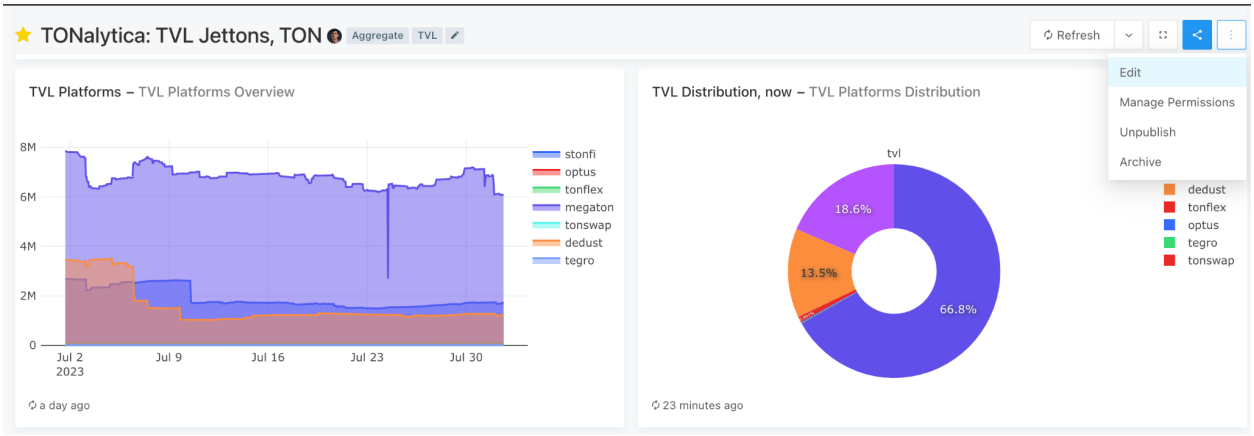
Add a text box to your dashboard using the `Text Box` tab on the Add Widget dialog. You can style the text boxes in your dashboards using Markdown.

You can include static images on your dashboards within your markdown-formatted text boxes. Just use markdown image syntax:`![]( <url for image > )`

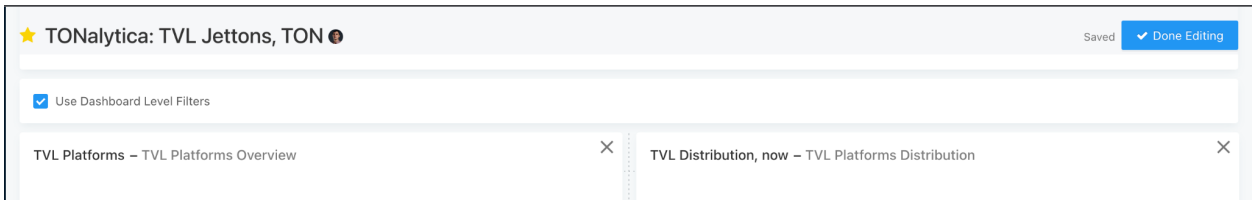
Dashboard Filters

When queries have filters you need to apply filters at the dashboard level as well. Setting your dashboard filters flag will cause the filter to be applied to all Queries.

1. Open dashboard settings:



2. Check the "Use Dashboard Level Filters" checkbox:

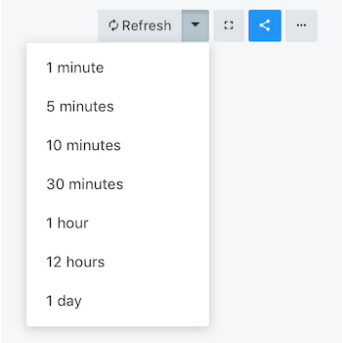


Dashboard Refresh

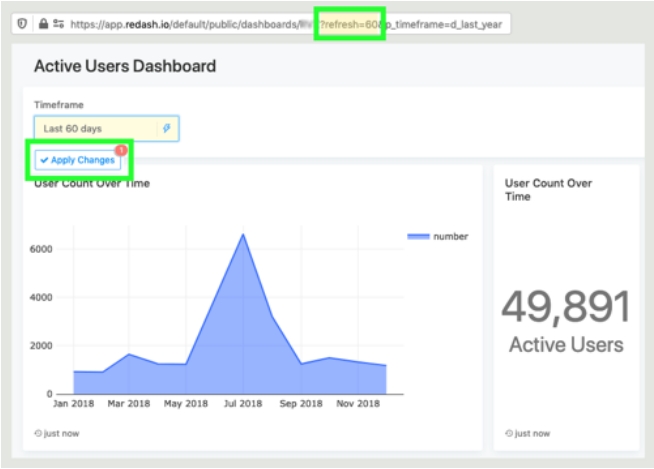
Even large dashboards should load quickly because they fetch their data from a cache that renews whenever a query runs. But if you haven't run the queries recently, your dashboard might be stale. It could even mix old data with new if some queries ran more recently than others.

To force a refresh, click the Refresh button on the upper-right of the dashboard editor. This runs all the dashboard queries and updates its visualizations.

If you want this to happen periodically you can activate Automatic Dashboard Refresh from the UI by clicking the dropdown pictured below. Or you can pass a `refresh` query string variable with your dashboard URL. The allowed refresh intervals are expressed in seconds: 60, 300, 600, 1800, 3600, 43200, and 86400.



Automatic Dashboard Refresh occurs as part of the TONalytica frontend application. Your refresh schedule is only in-effect as long as a logged-in user has the dashboard open in their browser. To guarantee that your queries are executed regularly (which is important for alerts), you should use a Scheduled Query instead.
On public dashboards there is no Refresh button. You can add `refresh` to the query string. And for dashboards with parameters you can trigger a refresh by changing a parameter value and clicking Apply Changes.



Data Tables

TONalytica ingests data from own full-node to directly fill our raw tables for each chain. This data is then decoded using special parsers to provide easier to work with decoded tables. Then we create abstracted tables that standardize and aggregate the data (from all other tables) - giving you the easiest to work. While Blockchain data is cool on it's own and we do our best to prepare, standardize and work with that data, sometimes a bit of off-chain data or augmented on-chain data is needed as well.

Jettons tables

Here is a list of available Jettons related data sources. All tables has internal `id` primary field.

active_jettons_info

Actual information about all active jettons (>10TON TVL):

- `address` - jetton address
- `admin_address`- admin address
- `decimals`- decimals
- `market_volume_ton`- 24h market volume in TON
- `price`- latest price
- `total_tvl`- total TVL in TON
- `symbol`- symbol

jettons_market_data

Aggregated info for all jettons with all history

- `build_time` - data point time
- `address` - jetton address
- `symbol` - jetton symbol
- `price` - jetton price (in TON)
- `market_volume_ton` - 24h market volume in TON
- `active_owners_24` - 24h active traders
- `total_holders` - total holders
-

jetton_master

Information about jettons (master contracts)

- `address` - master contract address
- `admin_address, symbol, name, image, image_data, decimals, metadata_url, description` - see TEP-74, TEP-64

jetton_burn

All burn events (mostly for wrapped TON's)

- `time` - burn time
- `created_lt` - lt of burn message
- `query_id` - query id
- `amount` - amount burnt
- `owner` - burn initiator (actually it could be not the owner, depends on smart contract logic)
- `wallet` - wallet address

jetton_mint

TEP-74 does not specify the use of mint, but TEP-74 recommends using the internal_transfer layout for it (i.e., a message that is handled by wallets to update their balances). In the case of ordinary transfer, internal_transfer should precede the transfer operation. If it is absent, it is not an ordinary transfer but rather a mint operation.

- `time` - mint time
- `created_lt` - lt of mint message
- `amount` - amount minted
- `minter` - address who initiated mint (has to be master contract, otherwise it is some suspicious…)
- `from_address` - address specified as from_address in message layout (should be equals to `minter`, but could be empty as well)
- `wallet` - wallet address

jetton_transfer

TEP-74 transfers.

- `time` - transfer time
- `created_lt` - lt of transfer message
- `successful` - if transfer was successfully processed by wallet smart-contract (i.e. *transfer_internal* was executed without errors)
- `originated_msg_id` - msg_id with external messages which led to this transfer
- `query_id`, `amount`, `response_destination`, `custom_payload`, `forward_ton_amount`, `forward_payload` - see TEP-74
- `source_owner`, `destination_owner` - owners of jettons
- `source_wallet` - jetton wallet which handle transfer operation, should be used to determine jetton kind
- `sub_op` - 32-bit op code from `forward_payload` (if present)

jetton_wallets

- `wallet_address` - jetton wallet address (unique)
- `owner` - wallet owner address (constant)
- `jetton_master` - jetton master address (constant)
- `balance` - actual balance

NFT

nft_current

Current owners and sale ssate of the NFTs

- `address` - NFT address, primary key
- `collection` - NFT collection address
- `owner` - owner
- `price` - item price in case it is on sale

nft_deals

NFT sale deals

- `sale_address` - smart contract address, primary key
- `address` - NFT item address
- `deal_time` - deal time
- `seller` - previous owner
- `buyer` - new owner
- `price` - price of the sale

nft_collection

Information from collection smart contract

- `address` - collection address
- `owner` - admin
- `name, description, image, metadata_url` - metadata fields

DEXes

dex_tvl_current

Current TVL for all pools

- `address` - pool address
- `platform` - platform name
- `jetton_a` - first jetton in the pair
- `jetton_b` - second jetton in the pair
- `tvl_ton` - TVL in TON

dex_tvl_history

The same as previous one, but with all history (see `build_time` field)

dex_swaps

- `msg_id` - unique swap id
- `platform` -
- `swap_dst_token` - destination token address
- `swap_src_token` - source token address
- `swap_dst_amount` - destination token amount
- `swap_src_amount` - source token amount
- `swap_time` - swap time
- `swap_user` - user initiated swap

staking_balance

Current staking balances for all nominators pools

- `pool` - pool address
- `address` - nominator address
- `balance` - balance in nanoTON

Alerts

Setting Up An Alert

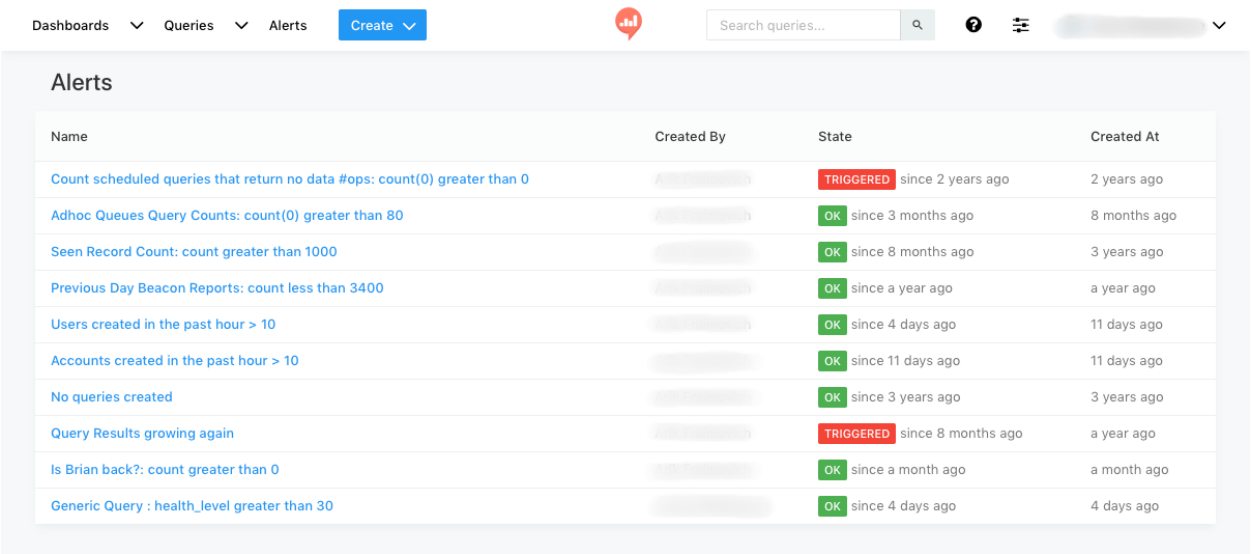
TONalytica alerts notify you when a field returned by a Scheduled Query meets a threshhold. Use them to monitor your business. Or integrate them with tools like Zapier or IFTTT to kickoff workflows such as user onboarding or support tickets. Alerts complement scheduled queries, but their criteria are checked after every execution.

A query schedule is not required but is *highly recommended* for alerts. If you add an alert to a non-scheduled query you will be notified only if a user executes the query manually and the alert criteria are met.

Alerts don't work for queries with parameters.

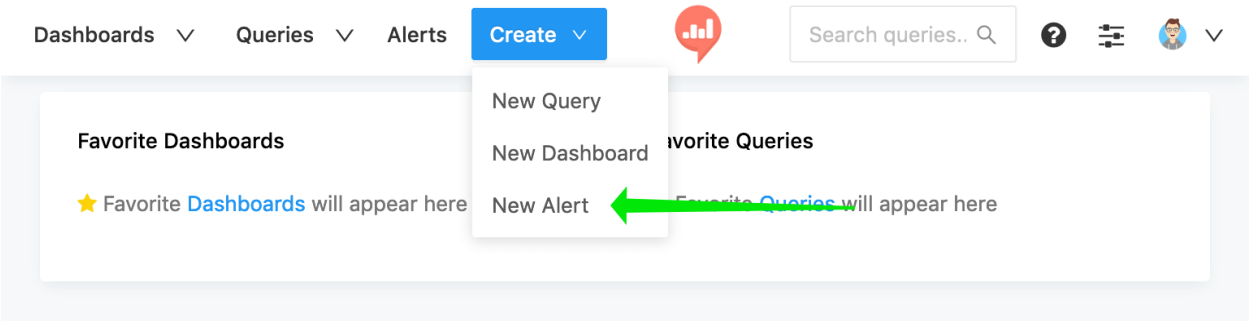
To see a list of current Alerts, click Alerts on the navbar. By default, they are sorted in reverse chronological order by the Created At column. You can reorder the list by clicking the column headings.

- Name shows the string name of each alert. You can change this at any time.
- Created By shows the user that created this Alert.
- State shows whether the Alert status is `UNKNOWN`, `TRIGGERED`, or `OK`.

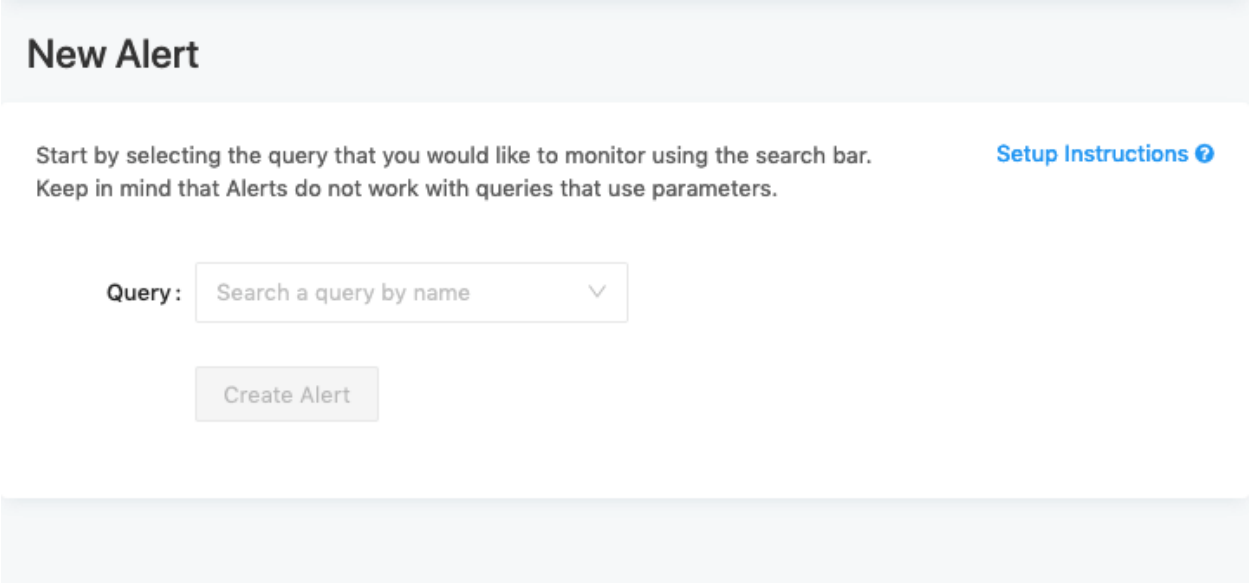


Usage

Click the Create button in the navbar and then click New Alert.

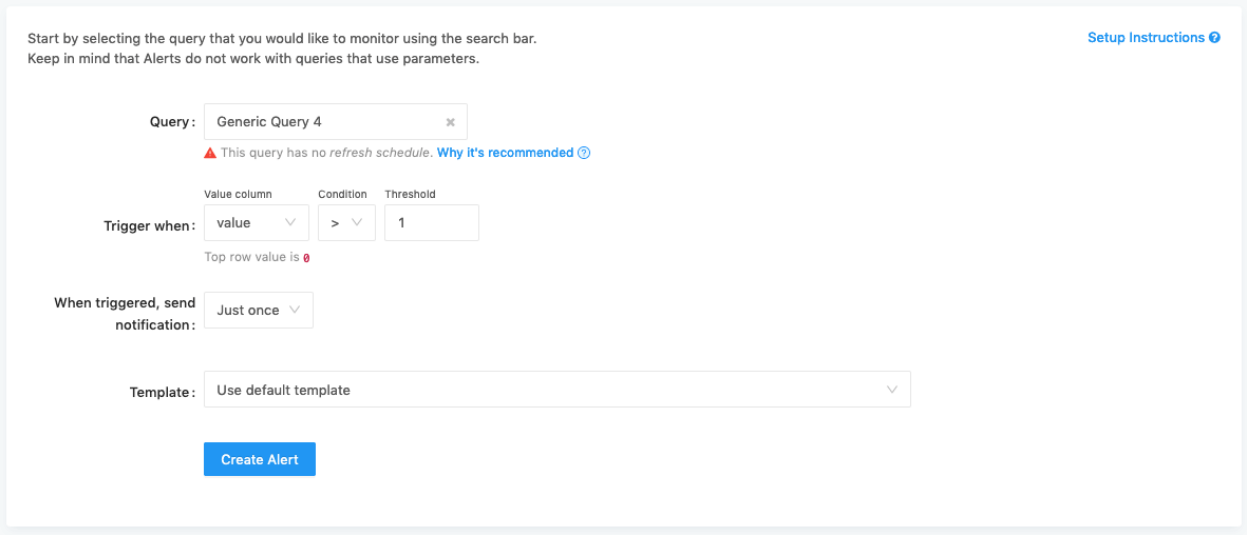


Search for a target query. If you don't see the one you want, make sure it is published and does not use parameters.



Use the settings panel to configure your alert.

- The Value column dropdown controls which field of your query result will be evaluated.
- The Condition dropdown controls the logical operation to be applied.
- The Threshold text input will be compared against the *Value column* using the *Condition* you specify.



If a target query returns multiple records, TONalytica Alerts only see the first one. As you change the Value Column setting, the current value of that field in the top row is shown beneath it.

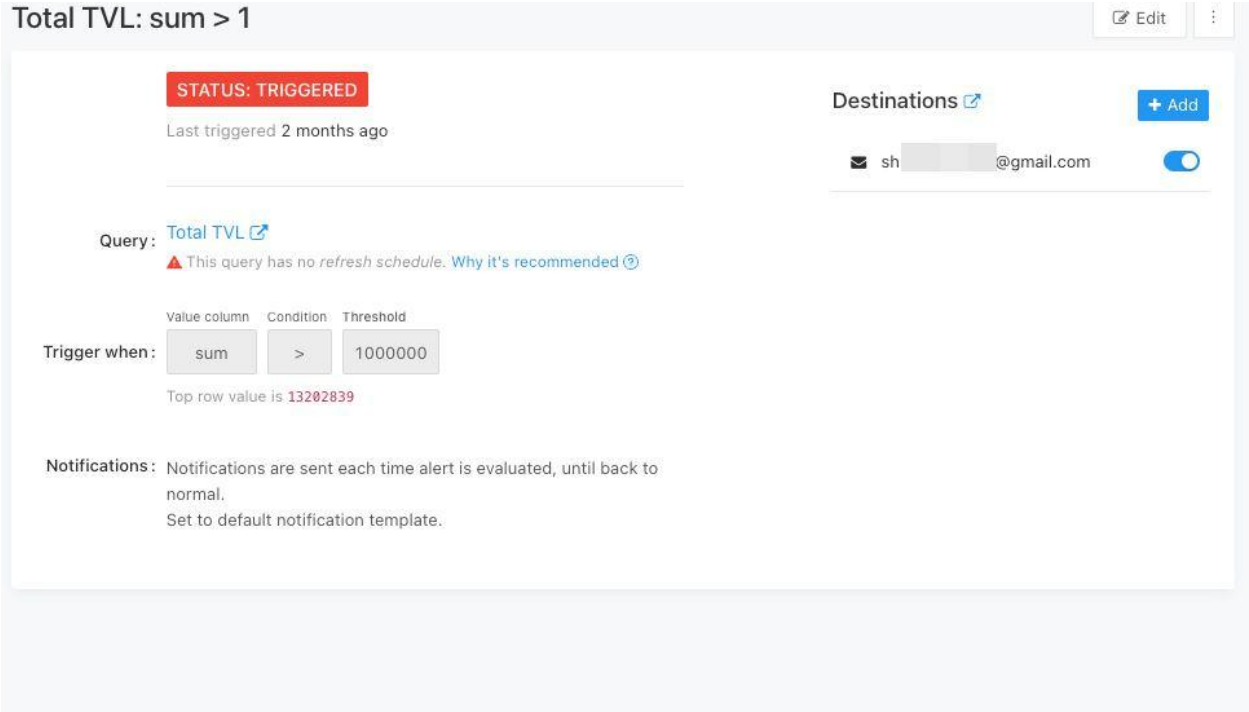
Next, adjust how many notifications to receive while your alert is triggered. There are three options:

- Just Once means a notification will fire any time the alert status changes from `OK` to `TRIGGERED`.
- Each time alert is evaluated means a notification will fire whenever the alert status is `TRIGGERED` regardless of its status as of the previous evaluation.
- At most every lets you set a minimum interval between notifications. It splits the difference between *Just Once* and *Each time alert is evaluated*. This choice lets you avoid notification spam for alerts that trigger often.

Regardless of which notification setting you pick here, you will receive a notification whenever the status goes from `OK` to `TRIGGERED` or from `TRIGGERED` to `OK`. The schedule settings above only impact how many notifications you will receive if the status remains `TRIGGERED` from one execution to the next.

Finally, pick a Template. The default template is a message with links to the Alert configuration screen and the Query screen. Many users will want to include more specific information about the Alert. To do this you can Customize The Alert Template.

When you're finished, click Create Alert and then choose an Alert Destination. If you skip this step you will not be notified when the alert is triggered.



Muting Alerts

You can temporarily mute an alert's notifications without deleting the alert entirely. Just click the vertical ellipsis (⋮) menu and choose *Mute Notifications*.

To resume notifications again, click the vertical ellipsis menu and choose *Unmute Notifications*.

Alert Statuses

- `TRIGGERED` means that on the most recent execution, the *Value Column* in your target query met the *Condition* and *Threshhold* you configured. If your alert checks whether "cats" is above 1500, your alert will be triggered as long as "cats" is above 1500.
- `OK` means that on the most recent query execution, the *Value Column* did not meet the *Condition* and *Threshhold* you configured. This doesn't mean that the Alert was not triggered previously. If your "cats" value is now 1470 your alert will show as OK.
- `UNKNOWN` means TONalytica does not have enough data to evaluate the alert criteria. You will see this status immediately after creating your Alert until the query has executed. You will also see this status if there was no data in the query result or if the most recent query result doesn't include the *Value Column* you configured.

Notification Frequency

TONalytica sends notifications to your chosen Alert Destinations whenever it detects that the Alert status has changed from `OK` to `TRIGGERED` or vice versa. Consider this example where an Alert is configured on a query that is scheduled to run once daily. The daily status of the Alert appears in the table below. Prior to Monday the alert status was `OK`.

Day	**Alert Status**
Monday	OK
Tuesday	OK
Wednesday	TRIGGERED
Thursday	TRIGGERED
Friday	TRIGGERED
Saturday	TRIGGERED
Sunday	OK

If the notification frequency is set to *Just Once*, TONalytica would send a notification on Wednesday when the status changed from `OK` to `TRIGGERED` and again on Sunday when it switches back. It will not send alerts on Thursday, Friday, or Saturday unless you specifically configure it to do so because the Alert status did not change between executions on those days.

Customizing Alert Notifications

TONalytica alerts can notify you when your queries match some arbitrary criteria. If you wish to modify the notification message, click the “Edit” button at the top of the alert page.

Next to the setting labeled “Template”, click the dropdown and select “Custom template”. A box will appear, consisting of input fields for subject and body.

Any static content is valid, and you can also incorporate some built-in template variables:

- `ALERT_STATUS` - The evaluated alert status (string).
- `ALERT_CONDITION` - The alert condition operator (string).
- `ALERT_THRESHOLD` - The alert threshold (string or number).
- `ALERT_NAME` - The alert name (string).
- `ALERT_URL` - The alert page url (string).
- `QUERY_NAME` - The correlated query name (string).
- `QUERY_URL` - The correlated query page url (string).
- `QUERY_RESULT_VALUE` - The query result value (string or number).
- `QUERY_RESULT_ROWS` - The query result rows (value array).
- `QUERY_RESULT_COLS` - The query result columns (string array).
- `QUERY_RESULT_TABLE` - Query results formatted as two dimensional array of values.

An example subject, for instance, could be: `Alert "{{ALERT_NAME}}" changed status to {{ALERT_STATUS}}`

Click the “Preview” toggle button to preview the rendered result and save your changes by clicking the “Save” button.

The preview is useful for verifying that template variables get rendered correctly. It is not an accurate representation of the eventual notification content, as each alert destinations can display notifications differently.

To return to the default TONalytica message templates, reselect “Default template” at any time.

Multiple Column Alert

There's an indirect way to set an Alert based on multiple columns of a query:

Your query can implement the alert logic and return a boolean value for the Alert to trigger on. Something like:

```
SELECT CASE WHEN drafts_count > 10000 AND archived_count > 5000 THEN 1 ELSE 0
END
FROM (
SELECT sum(CASE WHEN is_archived THEN 1 ELSE 0 END) AS archived_count,
sum(CASE WHEN is_draft THEN 1 ELSE 0 END) AS drafts_count
FROM queries) data
```

This query will return 1 when drafts*count > 10000 and archived*count > 5000. Then you can configure the alert to trigger when the value is 1.

Integrations and API

API Authentication

All the API calls support authentication with an API key. TONalytica has two types of API keys:

- User API Key: has the same permissions as the user who owns it. Can be found on a user profile page.
- Query API Key: has access only to the query and its results. Can be found on the query page.

Whenever possible we recommend using a Query API key.

Accessing with Python

We provide a light wrapper around the TONalytica API called `tonalytica-client`. It's a work-in-progress. The source code and library are published on PyPi.

Common Endpoints

Below is an incomplete list of TONalytica's API endpoints. These may change in future versions of TONalytica.

Each endpoint is appended to your TONalytica base URL. For example:

- `https://tonalytica.redoubt.online/<slug>`

Queries

`/api/queries`

- GET: Returns a paginated array of query objects.
    - Includes the most recent `query_result_id` for non-parameterized queries.
- POST: Create a new query object

`/api/queries/<id>`

- GET: Returns an individual query object
- POST: Edit an existing query object.
- DELETE: Archive this query.

`/api/queries/<id>/results`

- GET: Get a cached result for this query ID.
    - Only works for non parameterized queries. If you attempt to GET results for a parameterized query you'll receive the error: `no cached result found for this query`. See POST instructions for this endpoint to get results for parameterized queries.
- POST: Initiates a new query execution or returns a cached result.
    - The API prefers to return a cached result. If a cached result is not available then a new execution job begins and the job object is returned. To bypass a stale cache, include a `max_age` key which is an integer number of seconds. If the cached result is older than `max_age`, the cache is ignored and a new execution begins. If you set `max_age` to `0` this guarantees a new execution.
    - If passing parameters, they must be included in the JSON request body as a `parameters` object.

Here's an example JSON object including different parameter types:

```
{
    "parameters": {
      "number_param": 100,
      "date_param": "2020-01-01",
      "date_range_param": {
            "start": "2020-01-01",
            "end": "2020-12-31"
            }
      },
      "max_age": 1800
    }

}
```

Jobs

`/api/jobs/<job_id>`

- GET: Returns a query task result (job)
    - Possible statuses:
        - 1 == PENDING (waiting to be executed)
        - 2 == STARTED (executing)
        - 3 == SUCCESS
        - 4 == FAILURE
        - 5 == CANCELLED

    - When status is success, the job will include a `query_result_id`

Query Results

`/api/query_results/<query_result_id>`

- GET: Returns a query result
    - Appending a filetype of `.csv` or `.json` to this request will return a downloadable file. If you append your `api_key` in the query string, this link will work for non-logged-in users.

Dashboards

`/api/dashboards`

- GET: Returns a paginated array of dashboard objects.
- POST: Create a new dashboard object

`/api/dashboards/<dashboard_slug>`

- GET: Returns an individual dashboard object.
- DELETE: Archive this dashboard

`/api/dashboards/<dashboard_id>`

- POST: Edit an existing dashboard object.

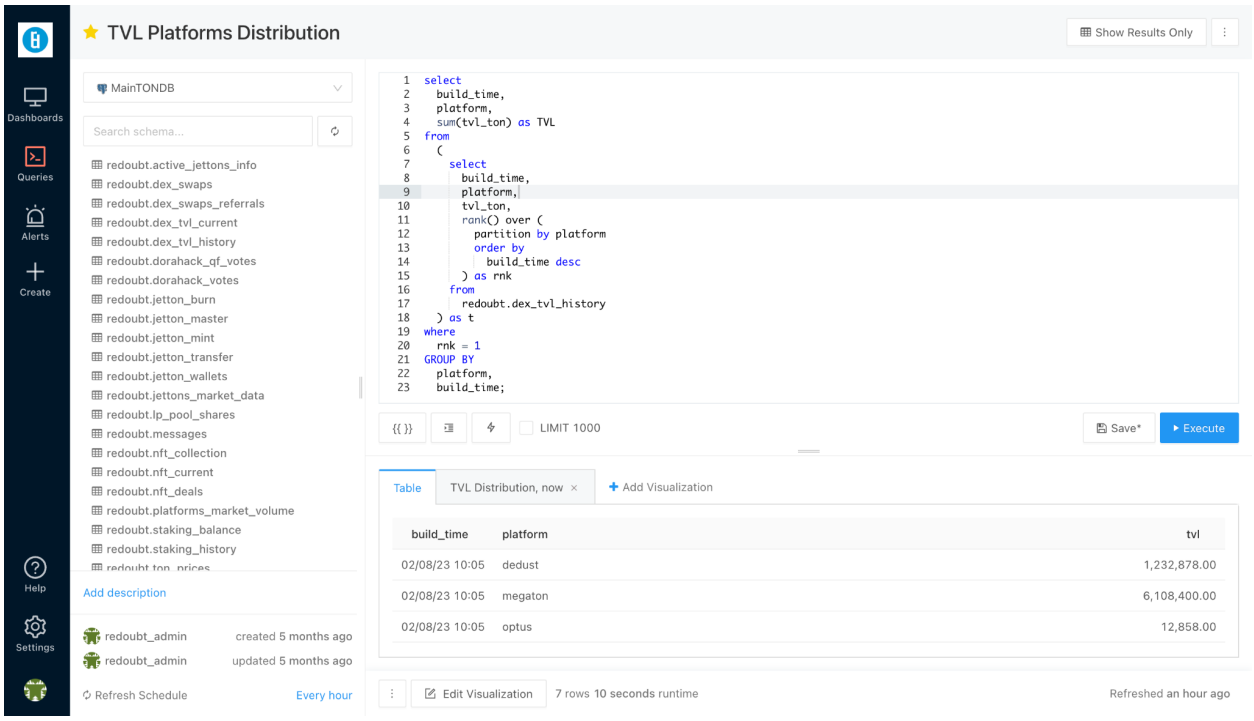
Use Cases: All the tools to unlock your data

TONalytica is a web-based platform that allows you to query TON blockchain data and aggregate it into beautiful dashboards.

TON are open and transparent, but TVM blockchains are unique—making it difficult to understand, ingest, and aggregate data. **TONalytica** gives you the proper tools to analyze cross-chain data for different tokens, wallets, and protocols. You can also easily share your work with the community.

Enjoy the power and comfort of TONalytica's query editor with powerful collaboration:

- Write queries in their natural syntax and explore schemas
- Live auto-complete and keyboard shortcuts
- Create snippets for elements you frequently use
- Results are cached for minimal running times
- Schedule auto-update times for results you rely on
- Use query results as data sources to join different databases



Visualize as you will & create amazing dashboards:

- Easily visualize your results in various formats: chart, cohort, pivot table, boxplot, map, counter, sankey, sunburst and word cloud
- Gather info from multiple sources into thematic dashboards
- Share your data-story with colleagues, other teams or external partners
- Share dashboards on a URL or embed widgets anywhere you need for timely and contextual data

Visualization types:
Charts: Line, Bar, Area, Pie, Scatter; Boxplot; Cohort; Sunburst; Word Cloud; Sankey; Map; Counter; Pivot Table; Funnel.

Product: https://tonalytica.redoubt.online/
Documentation (incl. Python/JS SDK):https://docs.tonalytica.redoubt.online/
Python SDK: https://pypi.org/project/tonalytica-client/

Dashboards samples: https://beta.redoubt.online/research

DEX Ecosystem key metrics

Here are the key metrics for the TON ecosystem - unique users, TVL, LPs count and tokens (Jettons) count. These metrics provide a rough overview, but a closer examination is necessary to fully understand how the ecosystem is growing.

DEX stat: unique users

Query link: https://tonalytica.redoubt.online/queries/83/source#161

Query snippet:

```
select  count(distinct swap_user)  as unique_users, count(distinct
msg_id)  from redoubt.dex_swaps
```

Sample output: 18,930 Total unique users

Dex stat: Total TVL

Query link: https://tonalytica.redoubt.online/queries/85/source#165

Query snippet:

```
select sum(tvl_ton) from redoubt.dex_tvl_current
```

Sample output: 9,122,427 Total TVL

Dex stat: active tokens

Query link: https://tonalytica.redoubt.online/queries/82/source#167
Query snippet:

```
select count(distinct address) from redoubt.jettons_market_data where
build_time > now() - interval '1  month'
and market_volume_ton > 1000
```

Sample output: 39 Active tokens (>1kTON daily in last month)

Platforms

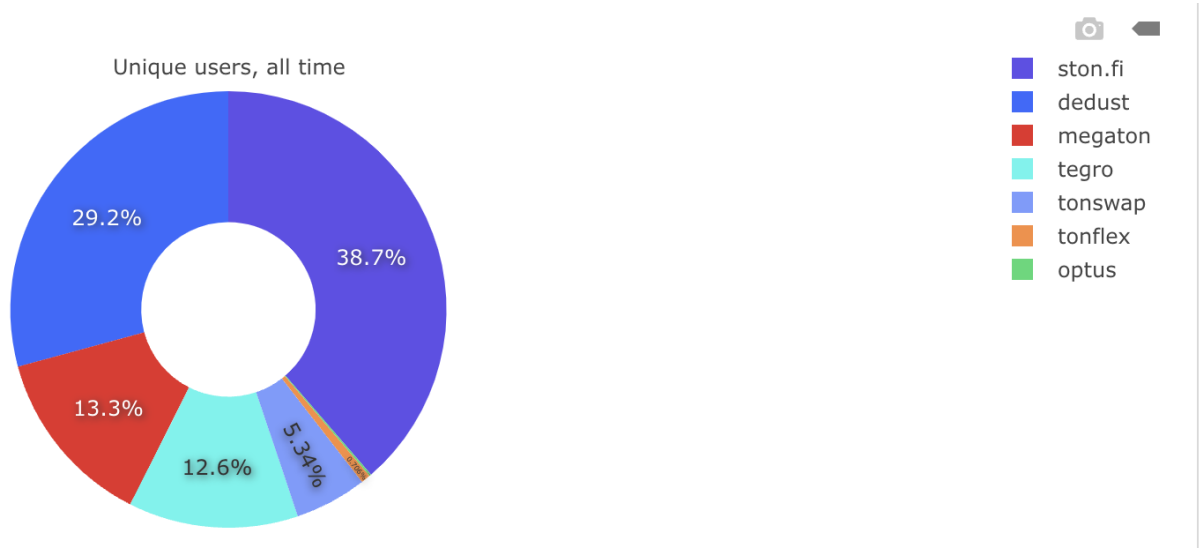
There were 7 DEXes, but one DEX (tonswap) has ceased operations. These DEXes are quite different, not only in terms of implementation, but also in terms of user audience.

DEX stat: unique users overall and swaps per user

Query link: https://tonalytica.redoubt.online/queries/82/source#158
Query snippet:

```
select  platform, count(distinct swap_user)  as unique_users,
1.0 * count(distinct msg_id) /  count(distinct swap_user)  as
swaps_per_user from redoubt.dex_swaps
group by 1
```

Sample output:

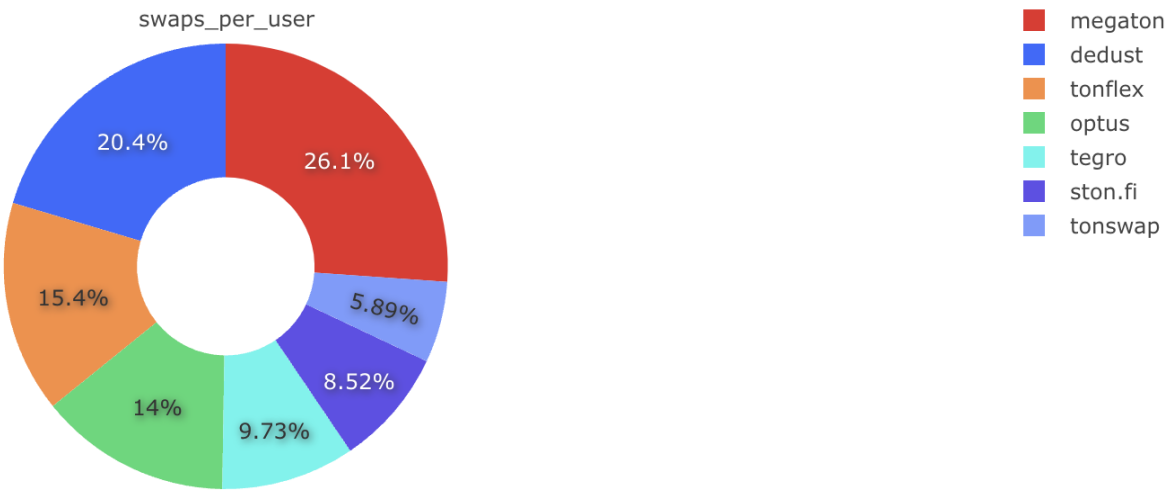


Dex stat: active tokens

Query link: https://tonalytica.redoubt.online/queries/82/source#159
Query snippet:

```
select  platform, count(distinct swap_user)  as unique_users,
1.0 * count(distinct msg_id) /  count(distinct swap_user)  as
swaps_per_user from redoubt.dex_swaps
group by 1
```

Sample output:



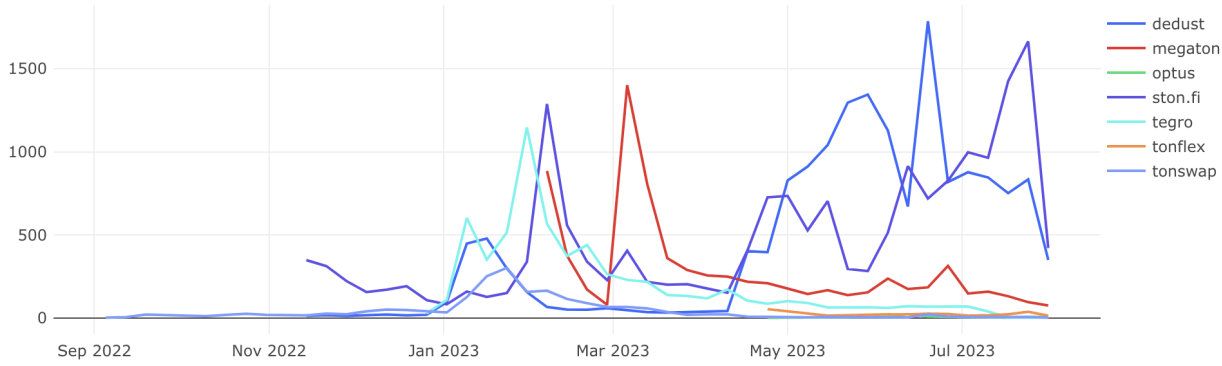
User base growth

DEX stat: WAU by platforms

Query link: https://tonalytica.redoubt.online/queries/81/source#156
Query snippet:

```
select  platform, date_trunc('week', swap_time::date) as month,
count(distinct swap_user)  as unique_users from redoubt.dex_swaps
group by 1, 2
```

Sample output:

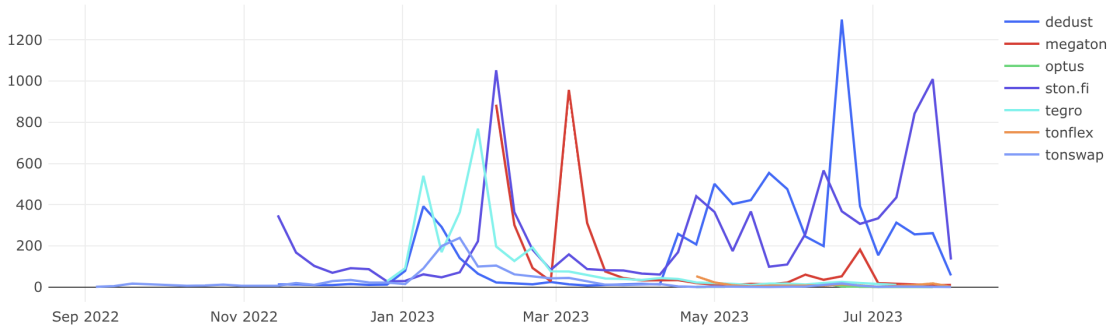


DEX stat: new users weekly

Query link: https://tonalytica.redoubt.online/queries/84/source#163
Query snippet:

```
with first_engagement as (
  select  platform, swap_user, min(swap_time::date) as first_swap
from redoubt.dex_swaps
  group by 1, 2
)
select platform, date_trunc('week', first_swap) as date,
count(distinct swap_user) as new_users
from first_engagement
group by 1, 2
```

Sample output:



Audience intersection

DEX stat: number of DEXes user by user

Query link: https://tonalytica.redoubt.online/queries/87/source#169
Query snippet:

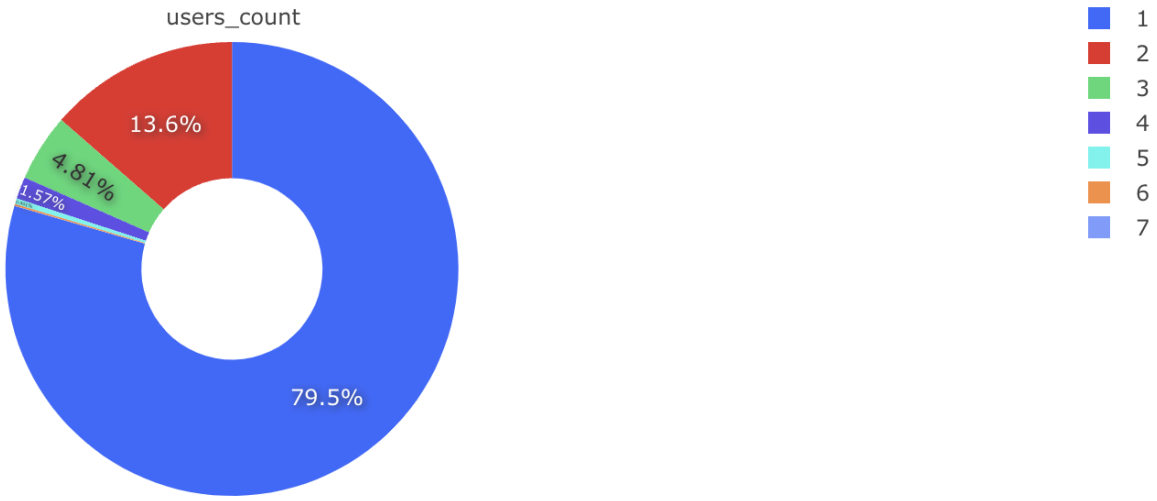
```
with users_stat as (
  select  swap_user,count(distinct platform) as plaforms_count from
redoubt.dex_swaps
  group by 1
)
select plaforms_count, count(distinct swap_user) as users_count
from users_stat
```

```
group by 1
```

Sample output:

DEX stat: unique audience percent

Platform	Unique audience share (for thi...	Total audience
dedust	62 %	7,204
megaton	64 %	3,291
optus	21 %	33
ston.fi	67 %	9,531
tegro	46 %	3,103
tonflex	22 %	174
tonswap	49 %	1,316



TVL Jettons

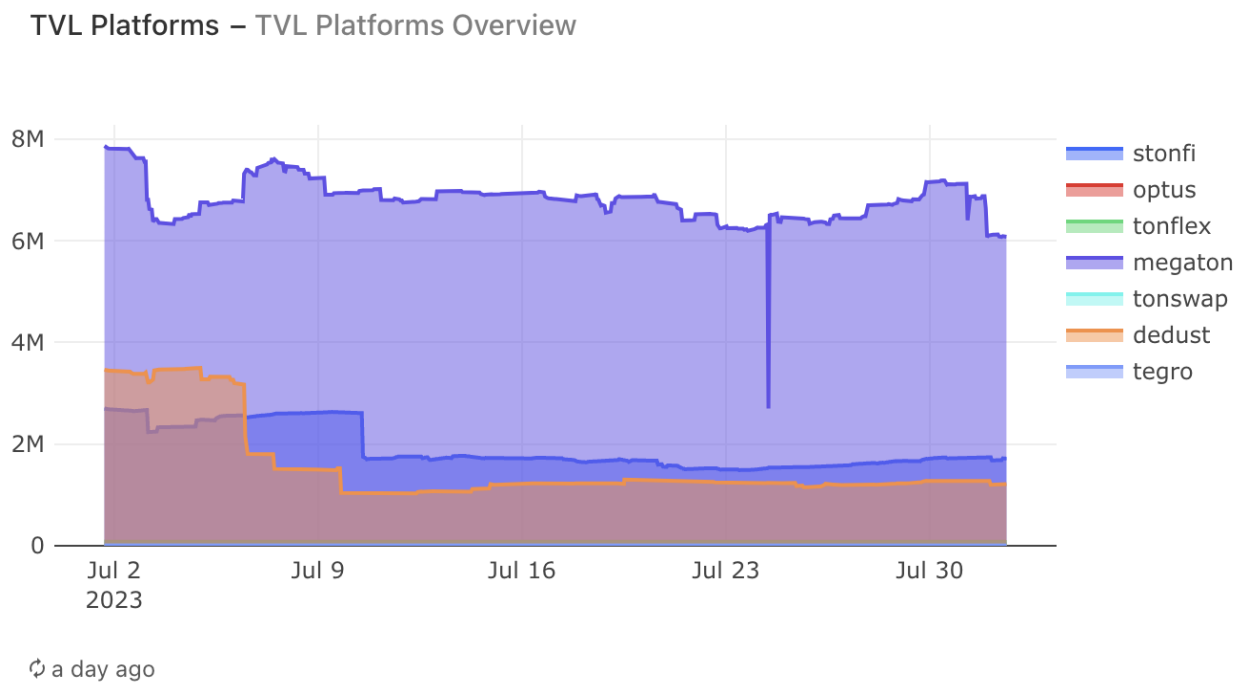
TVL Platforms Overview

Query link: https://tonalytica.redoubt.online/queries/3

Query snippet:

```
select build_time, platform, sum(tvl_ton) as TVL from
redoubt.dex_tvl_history
WHERE build_time > now() - interval '{{ interval }}'
GROUP BY build_time, platform
ORDER BY build_time DESC;
```

Sample output:



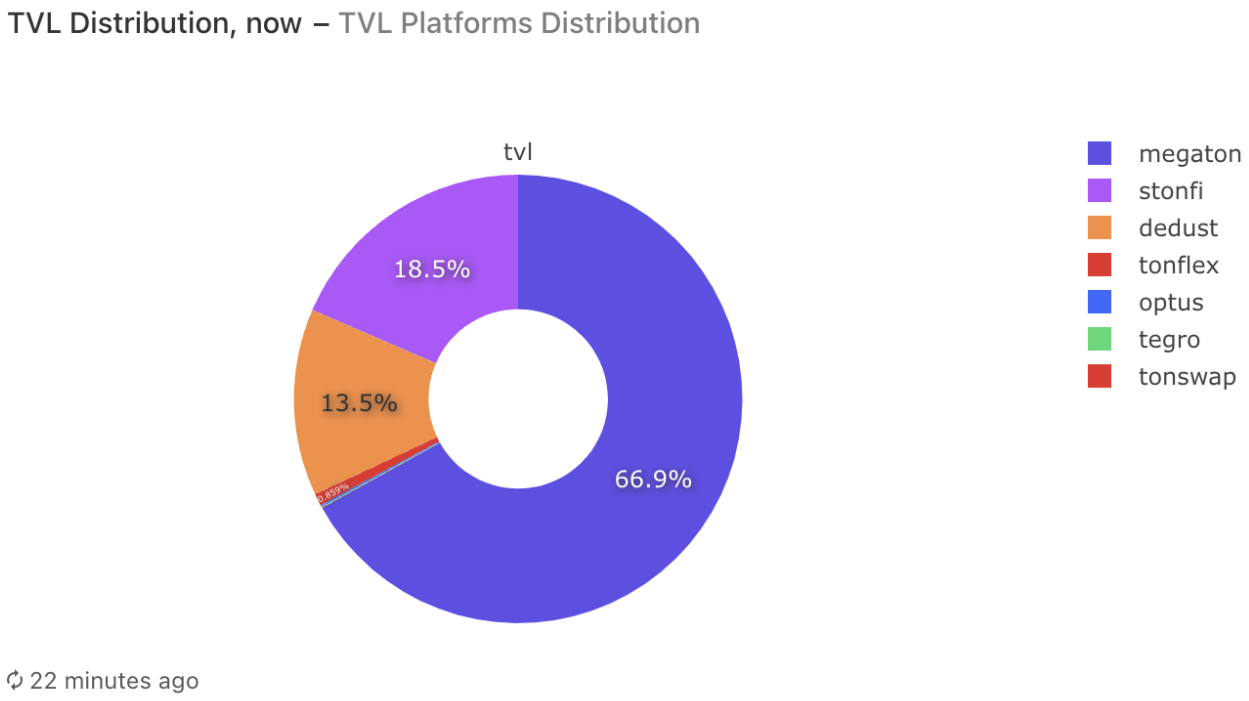
TVL Platforms Distribution

Query link: https://tonalytica.redoubt.online/queries/14/source#35
Query snippet:

```
select build_time, platform, sum(tvl_ton) as TVL
from
  ( select build_time, platform, tvl_ton,
```

```
           rank() over (partition by platform
                        order by build_time desc) as rnk
    from redoubt.dex_tvl_history
  ) as t
where rnk = 1
GROUP BY platform, build_time;
```

Sample output:



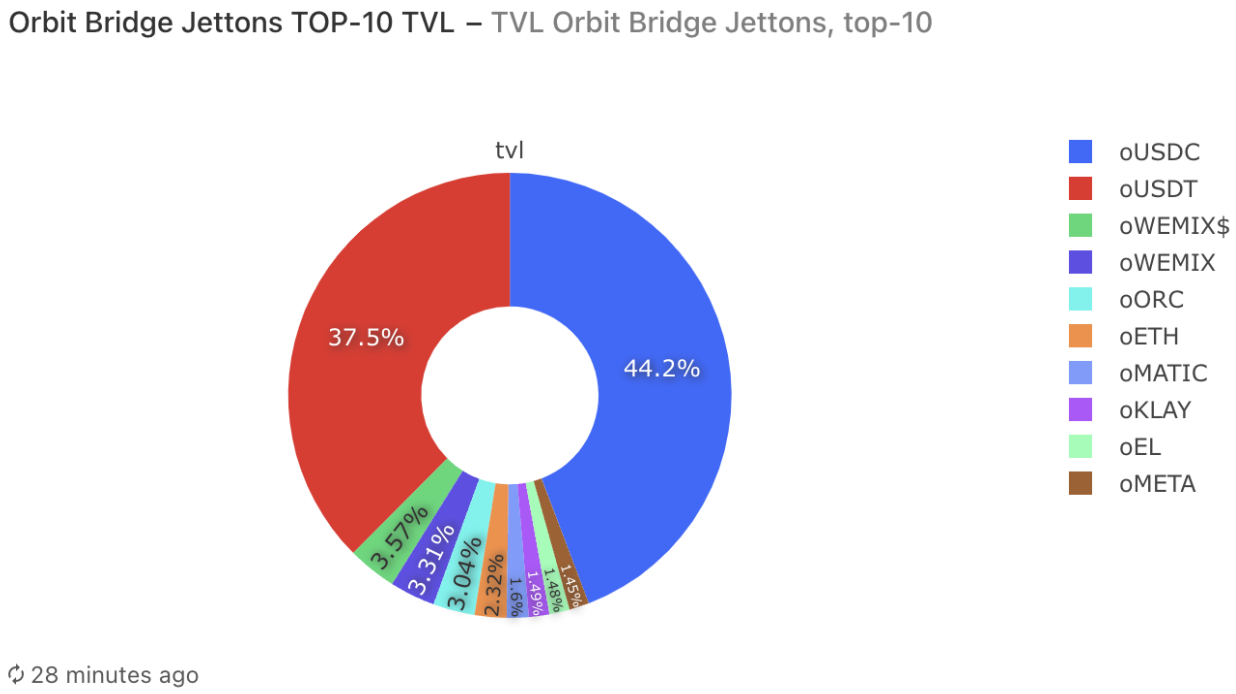
TVL Orbit Bridge Jettons, top-10

Query link: https://tonalytica.redoubt.online/queries/15/source#37
Query snippet:

```
SELECT jetton,
       sum(tvl_ton)/2 AS tvl
FROM redoubt.dex_tvl_current
```

```
CROSS JOIN unnest(array[jetton_a, jetton_b]) AS t(jetton)
WHERE jetton LIKE 'o%'
GROUP BY jetton
HAVING sum(tvl_ton)/2>1000
ORDER BY tvl DESC,
         jetton DESC
LIMIT 10;
```

Sample output:



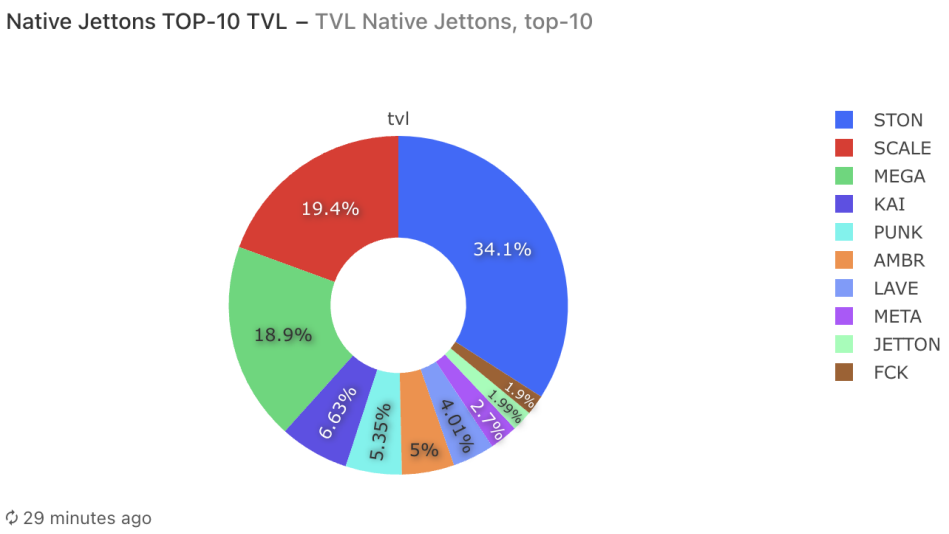
TVL Native Jettons, top-10

Query link: https://tonalytica.redoubt.online/queries/16/source#39
Query snippet:

```
SELECT
  jetton,
  sum(tvl_ton) / 2 AS tvl
FROM
```

```
  redoubt.dex_tvl_current
  CROSS JOIN unnest(array [jetton_a, jetton_b]) AS t(jetton)
WHERE
  jetton not LIKE 'o%'
  and jetton not LIKE 'j%'
  and jetton not in ('WTON', 'pTON', 'jTON', 'JTON', 'TON')
GROUP BY
  jetton
HAVING
  sum(tvl_ton) / 2 > 1000
ORDER BY
  tvl DESC,
  jetton DESC
LIMIT
  10;
```

Sample output:



Market Volumes Jettons

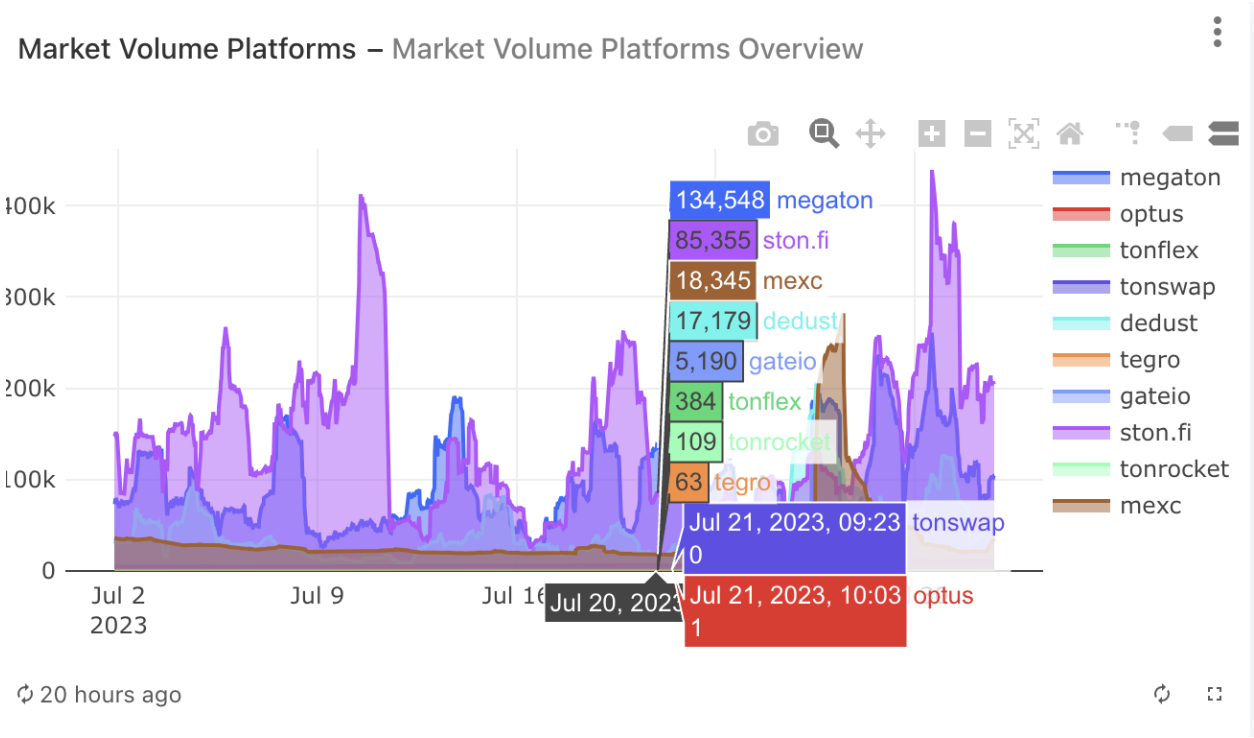
Market Volume Platforms Overview

Query link: https://tonalytica.redoubt.online/queries/17
Query snippet:

```
SELECT build_time,
       platform,
```

```
       sum(market_volume_ton) as volume
FROM redoubt.platforms_market_volume
WHERE build_time > now() - interval '{{ interval }}'
GROUP BY build_time,
         platform
ORDER BY build_time DESC;
```

Sample output:



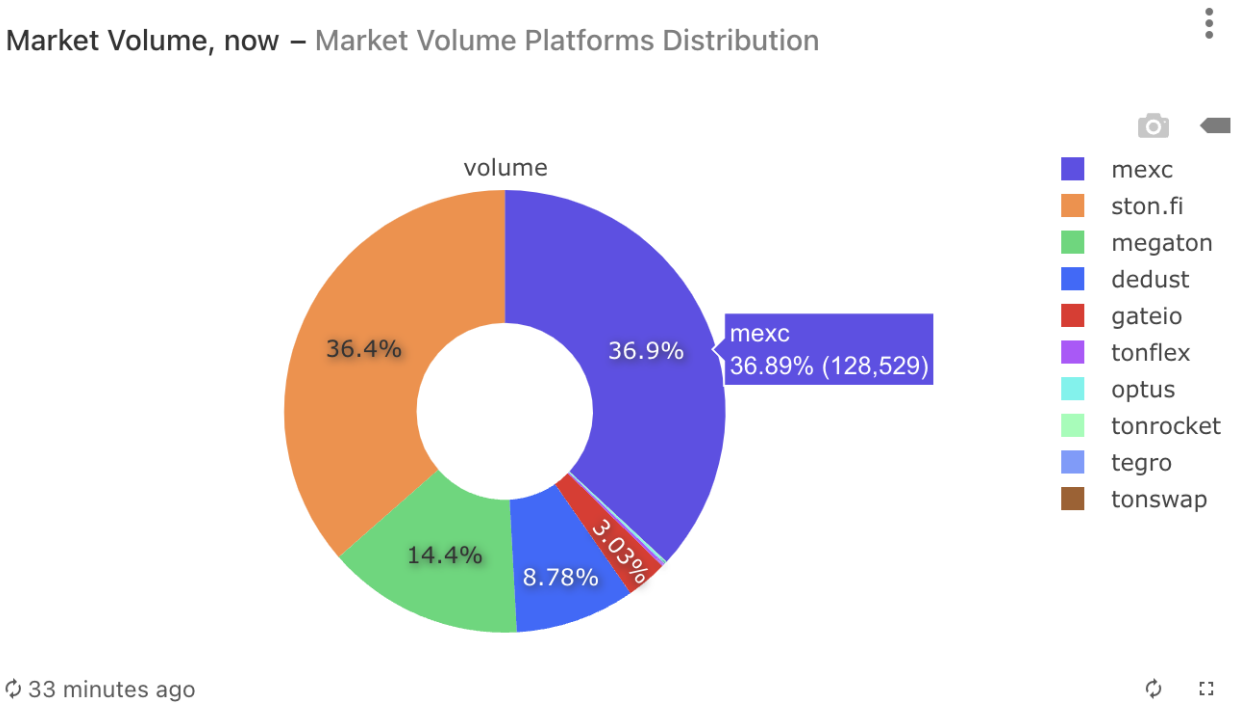
Market Volume Platforms Distribution

Query link: https://tonalytica.redoubt.online/queries/18
Query snippet:

```
select build_time, platform, sum(market_volume_ton) as volume
from
  ( select build_time, platform, market_volume_ton,
           rank() over (partition by platform
```

```
                        order by build_time desc) as rnk
    from redoubt.platforms_market_volume
  ) as t
where rnk = 1
GROUP BY platform, build_time;
```

Sample output:



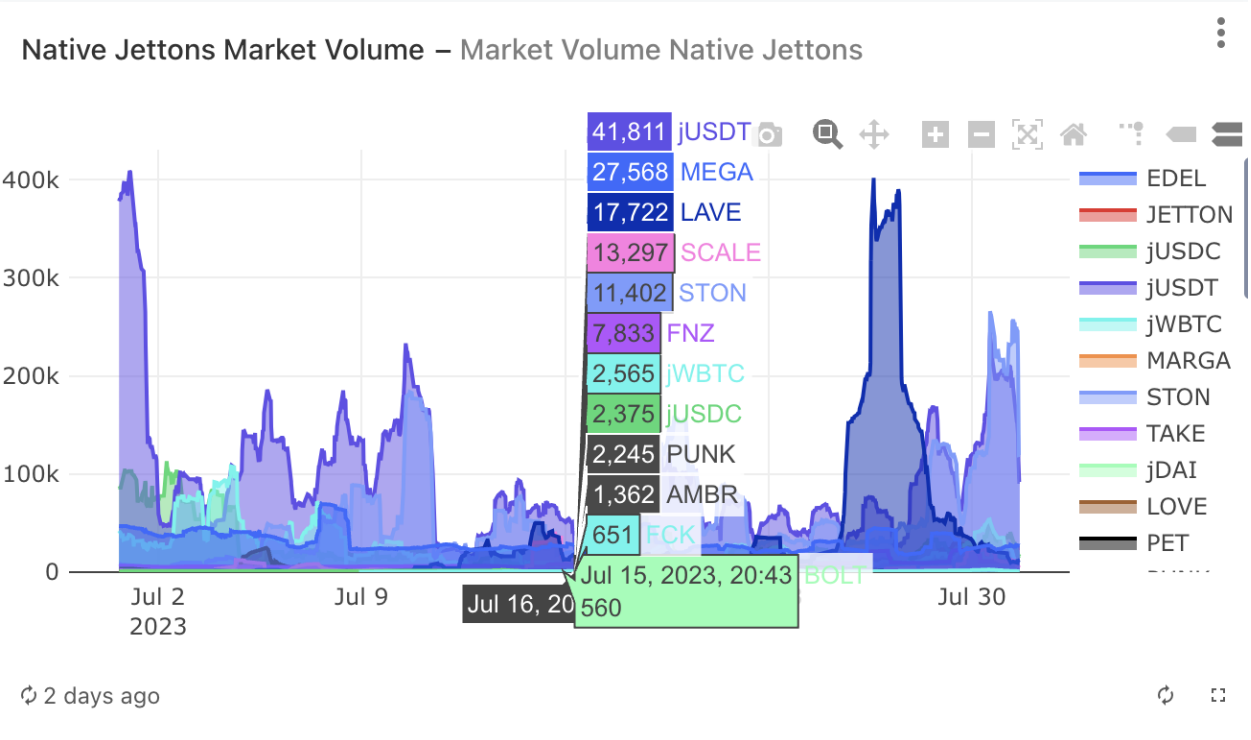
Market Volume Native Jettons

Query link: https://tonalytica.redoubt.online/queries/19
Query snippet:

```
SELECT build_time,
       symbol,
       active_owners_24,
       total_holders,
       market_volume_ton
FROM redoubt.jettons_market_data
```

```
WHERE market_volume_ton>300
  AND symbol NOT LIKE 'o%'
  AND symbol NOT IN ('WTON',
                     'pTON',
                     'jTON',
                     'JTON',
                     'TON')
  AND build_time > now() - interval '{{ interval }}'
ORDER BY build_time DESC,
         symbol ASC;
```

Sample output:



Market Volume Orbit Bridge Jettons

Query link: https://tonalytica.redoubt.online/queries/20
Query snippet:

```
SELECT build_time,
       symbol,
       active_owners_24,
       total_holders,
```

```
       market_volume_ton
FROM redoubt.jettons_market_data
WHERE market_volume_ton>300
  AND symbol LIKE 'o%'
  AND build_time > now() - interval '{{ interval }}'
ORDER BY build_time DESC,
         symbol ASC;
```

Sample output:)