Contents

[Build data lake house from data in a web page 1](#_Toc136050415)

[Goal of system 1](#_Toc136050416)

[**What data do I need?** 1](#_Toc136050417)

[**What is the data lake house?** 1](#_Toc136050418)

[Technique used in this system. 2](#_Toc136050419)

[Workflow of system: 3](#_Toc136050420)

[Set up: 3](#_Toc136050421)

[Explain dags(parts) in this system: 5](#_Toc136050422)

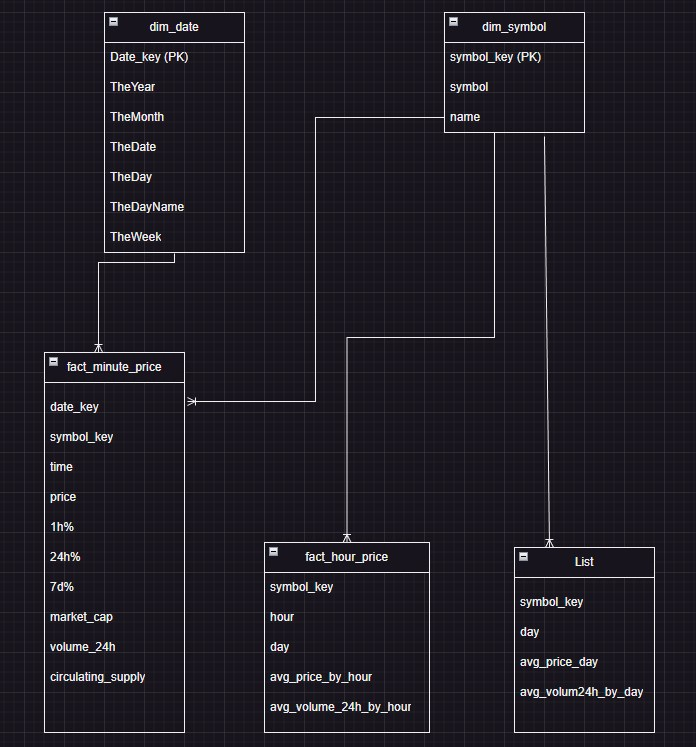
[How to use system? 5](#_Toc136050423)

[How to you visualize data in data lake house? 6](#_Toc136050424)

**Build data lake house from data in a webpage**

**Goal of system**

* The system monitors changes in the price of many popular coins in the world.
* What does the system really do?
  + Extracting data from a [web page](https://coinmarketcap.com/).
  + Transforming raw data.
  + Store clean data in a ‘data lake house’.



**What data do I need?**

* + Name
  + Symbol
  + Extract data about price
  + volume\_24h
  + 1h%, 24h%, 7d%
  + marketCap
  + Circulating Supply

**What is the data lake house?**

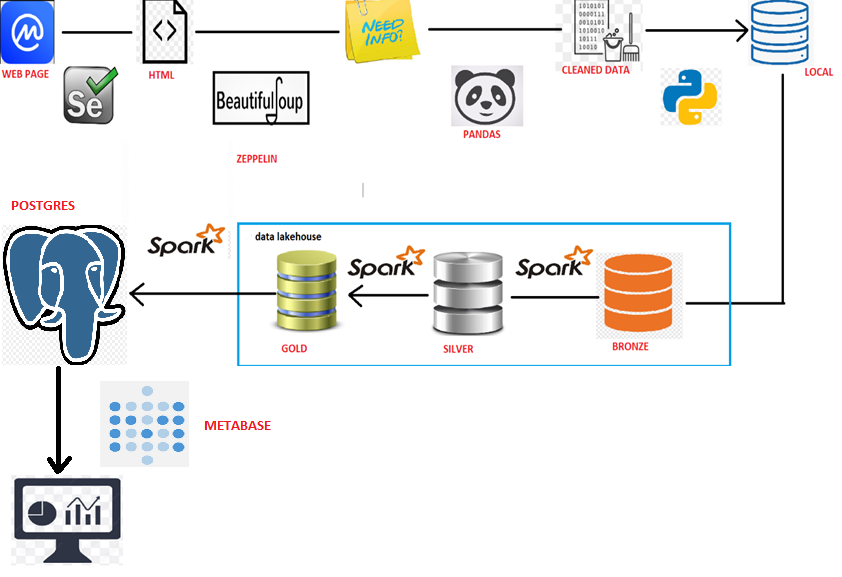
It is a new architecture which provides the ability of both data warehouse and data lake. It can store structured, unstructured, and semi structured data. With architecture divide into 3 layers:

* + Bronze layer: store raw data.
  + Silver layer: store cleaned data.
  + Gold layer: store data which is enriched and ready to make a report or use to other purpose.

**Technique used in this system.**

* **Selenium**:  is a free (open source) automated testing framework used to validate web applications across different browsers and platforms. You can use multiple programming languages like Java, C#, Python, etc. to create Selenium Test Scripts.
* **Bs4** - beautifulSoup: is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.
* **Pandas**: is a Python library used for working with data sets.
* **Spark**: Apache Spark is an open-source, distributed processing system used for big data workloads. It utilizes in-memory caching, and optimized query execution for fast analytic queries against data of any size. It provides development APIs in Java, Scala, Python and R, and supports code reuse across multiple workloads—batch processing, interactive queries, real-time analytics, machine learning, and graph processing.
* **Minio**: is a high-performance, S3 compatible object store. It is built for large scale AI/ML, data lake and database workloads. It runs on-prem and on any cloud (public or private) and from the data center to the edge. Minio is software-defined and open source under GNU AGPL v3.
* **Metabase**: is an easy-to-use, open source business intelligence tool that lets you analyze data from a variety of data destinations and sources.
* **Airflow**: is an open-source platform for developing, scheduling, and monitoring batch-oriented workflows. Airflow’s extensible Python framework enables you to build workflows connecting with virtually any technology. A web interface helps manage the state of your workflows.
* **Docker**: Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same way you manage your applications. By taking advantage of Docker’s methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

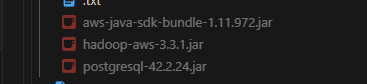
**Workflow of system:**



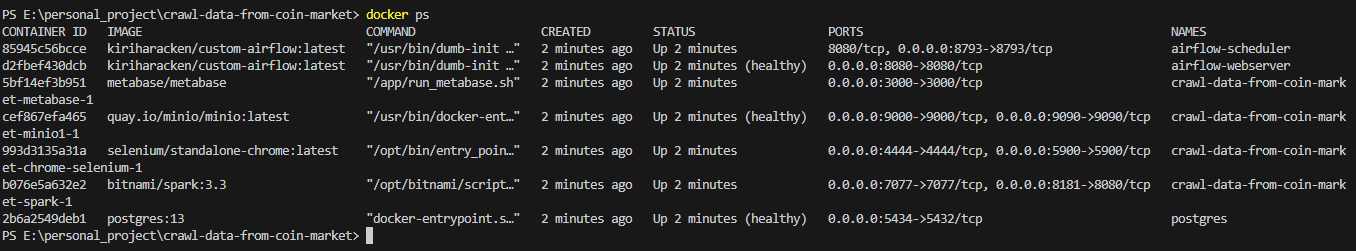
* start point: web page.
* using selenium to get a html source.
* using bs4 to extract needed info.
* using pandas to transform data from raw to clean.
* using python to store clean data locally. It is like a backup when you need it.
* using python to upload data from local to minio - data will store in bronze layer.
* using spark to connect minio → read data in csv file → incremental load into a parquet file - data store in silver layer.
* using spark to read only new part in silver layer → transform → load into dim and fact table - data store in gold layer.
* using spark to read data from gold layer and incremental load into data warehouse on postgres.
* end point: using metabase to visualize a dashboard

**Set up:**

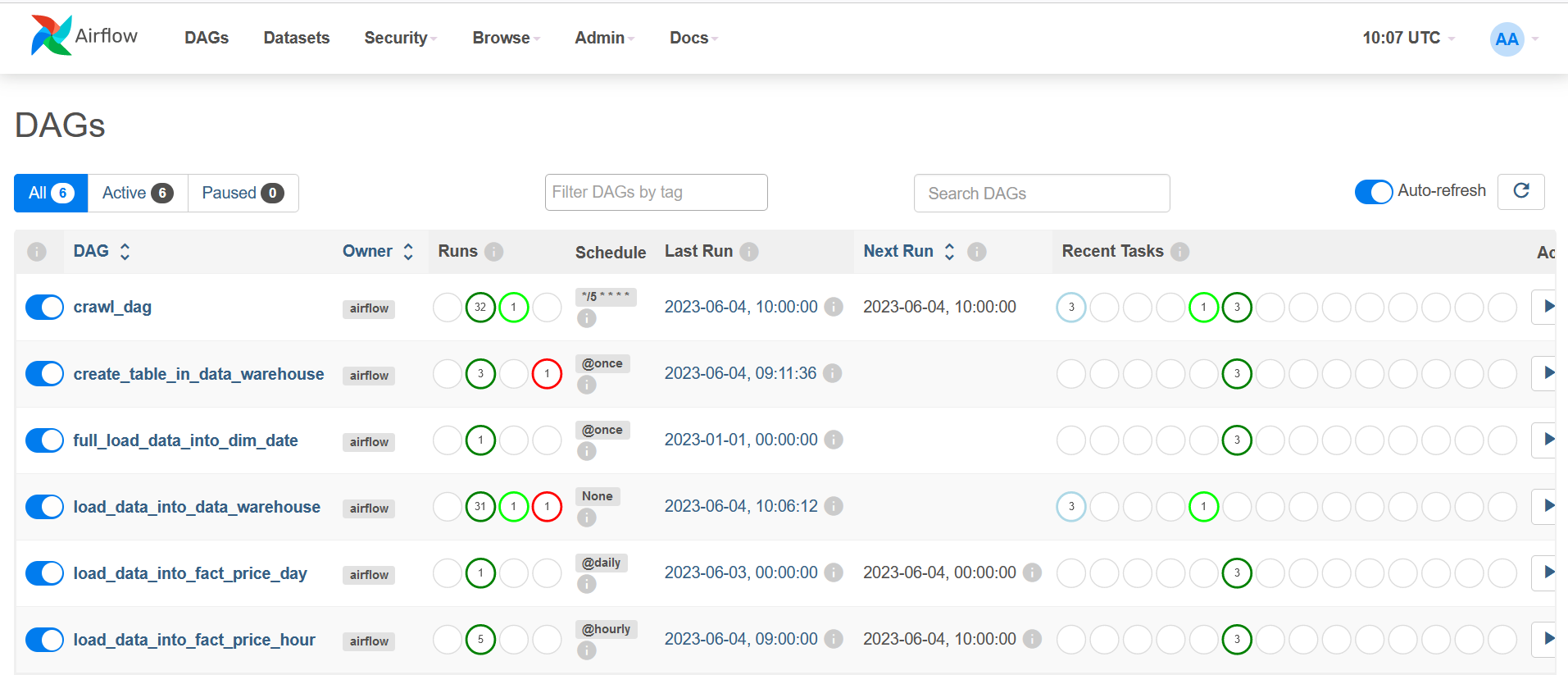
1. Download my folder - pull my project in github from [here](https://github.com/ckikriehanra/crawl_data_from_webpage_and_visualize).
2. Download and install docker desktop [here](https://docs.docker.com/desktop/install/windows-install/).
3. Start docker desktop.
4. Download and install Visual studio code [here](https://code.visualstudio.com/download).
5. You can modify user account of Minio inside file with path ./minio/.env
6. You can modify your folder name. file name which will be used to save data in Minio inside file with path ./my\_lib/config.py
7. Pull my image:
   1. Open Docker Desktop
   2. Type Ctrl+K
   3. Enter key words: kiriharacken/custom-airflow
   4. A window open, you click pull to pull this image to the local machine.
8. Open folder in vs code:
   1. Open vs code
   2. Type: Ctrl + K + O
   3. Choose your folder need open which you downloaded named: Crawl\_Webpage\_docker.
9. Download .jar for spark. To know more information you need to read file .txt in ./spark/resources/jars/



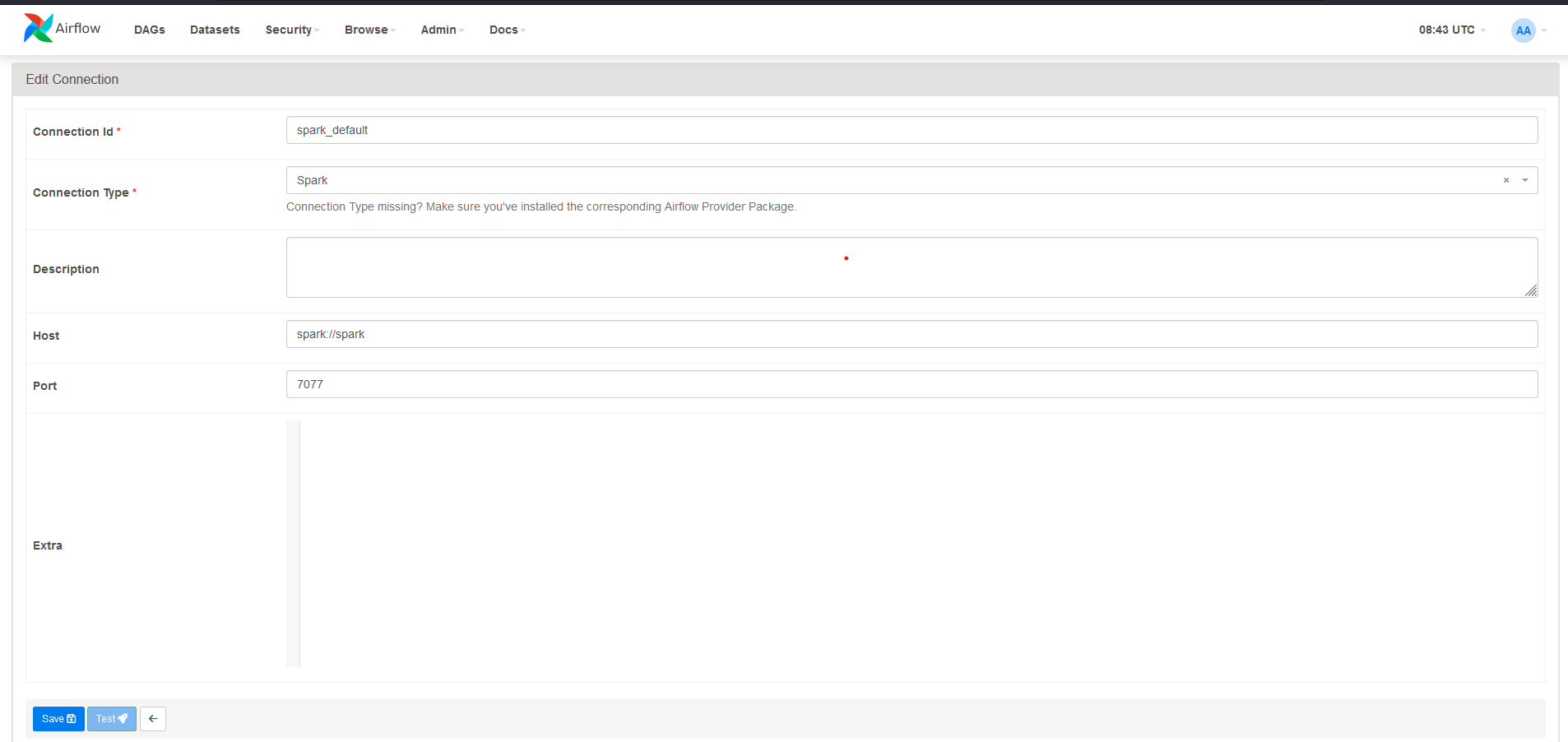
1. Open terminal: Ctrl + J
2. Start project by scripts:
   1. docker-compose up airflow-init
   2. Open new terminal by (Ctrl + Shift + `) → type: docker-compose up -d
   3. Open new terminal by (Ctrl + Shift + `) → type: docker container ls
   4. Wait several minutes and retry: docker container ls. It ok if you see:



* 1. Open browser, in search bar enter: <http://localhost:8080> → Sign in with username: airflow and password: airflow. It is ok if you see console like image. In your image, because you never trigger any dag then you won’t see round color red, green:



1. In header, you choose Admin → choose connection. Now you need to configure your connection to help airflow interact with spark:
   1. Click Add a new record.
   2. Add a connection with information then click save.



**Explain DAGs in this system**

* Create\_table\_in\_data\_warehouse: used to create table in data warehouse in postgres.
* Full\_load\_data\_into\_dim\_date: used to full load data into dim\_date, can be triggered by handle.
* Load\_data\_into\_data\_warehouse: load data into data warehouse from data lakehouse.
* Crawl\_data: crawl data from web page, transform and load data into data lake house.
* Load\_data\_into\_fact\_price\_hour: transform data into average value and load data into fact\_price\_hour. This value beyond latest last hour.
* Load\_data\_into\_fact\_price\_day: transform data into average value and load data into fact\_price\_day. This value beyond lastest last day.

**How to use this system?**

* Before running this project, you need to check data lake:
  + Open browser and enter <http://localhost:9090>.
  + Sign in by username: kirihara, password: minioadmin. You can change this account by going inside ./minio/.env
  + If you see the existing bucket then it’s ready to start. Like this:

A picture containing text, screenshot, font

Description automatically generated

Else you need to create a new bucket with the name “cken-coins-data”. You can set specific name by going inside file with path ./my\_lib/config.py.

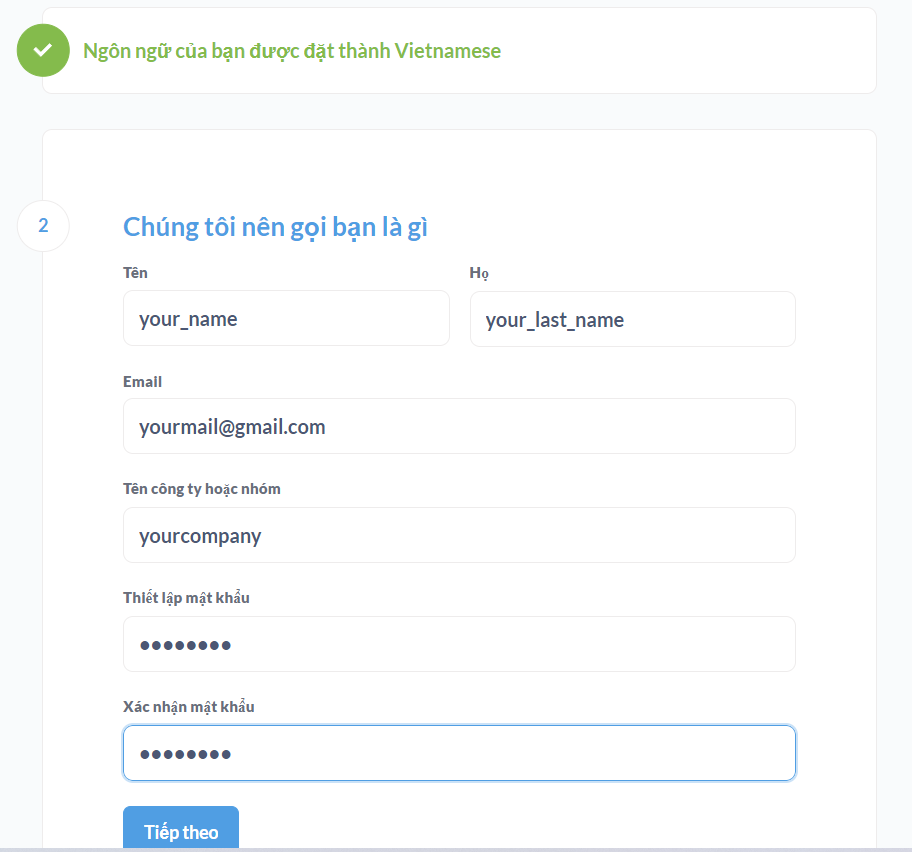
* Firstly, dag needs to be run is ‘full\_load\_data\_into\_dim\_date’. With this dag you only need to trigger once time. After dag success you should wait for a two minutes to data about date write into minio.
* Secondly, trigger dag name ‘create\_table\_in\_data\_warehouse’ to create table in data warehouse in postgres.
* Thirdly, trigger dag name ‘load\_data\_into\_data\_warehouse’ to prepare run immediately after run success dag ‘crawl\_data’.
* Fourth, before trigger dag needs run is ‘crawl\_data’, you need check time. You must trigger this dag at a time which has minute is a number can divisible by 5. This dag set time to automatically runs every 5 minutes. If you want to change the number of pages which you want crawl, you can change two variables start\_page and end\_page in file with path ./my\_lib/config.py.
* Fifth, you can trigger dag ‘load\_data\_into\_fact\_price\_hour’.
* Sixth, you can trigger dag ‘load\_data\_into\_fact\_price\_day’.

**How do you visualize data?**

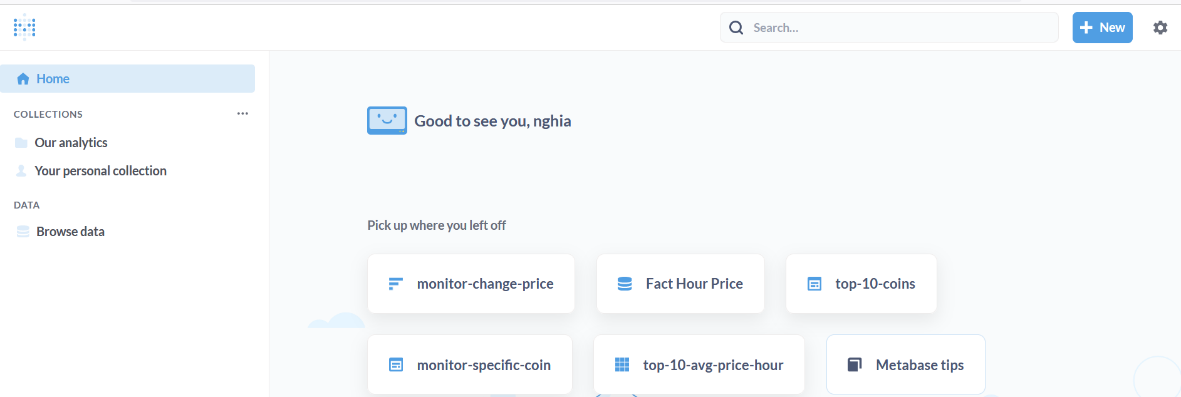
Here I choose Metabase to demo. You can use other tool BI like powerBI, superset, etc. to visualize data. So how to setup Metabase and how to use it to visualize data as a dashboard.

## How to setup Metabase?

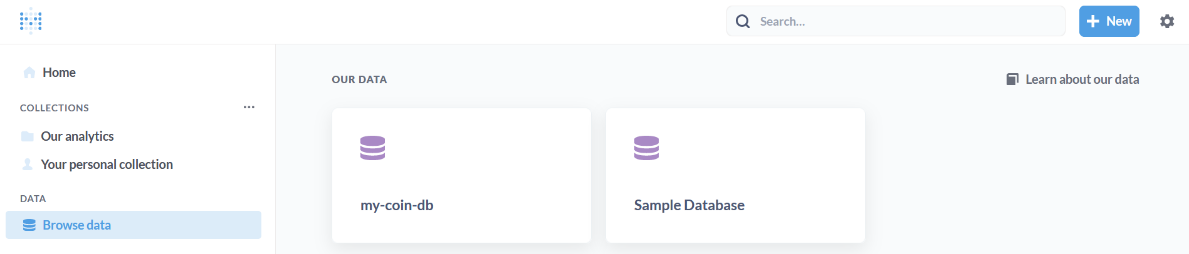
* Go to UI of metabase by this [link](http://localhost:3000).
* Sign up your account
* Add data:
  + Select postgres
  + Fill needed data. Here you must fill like me because info about database only can be change in source code if you want.



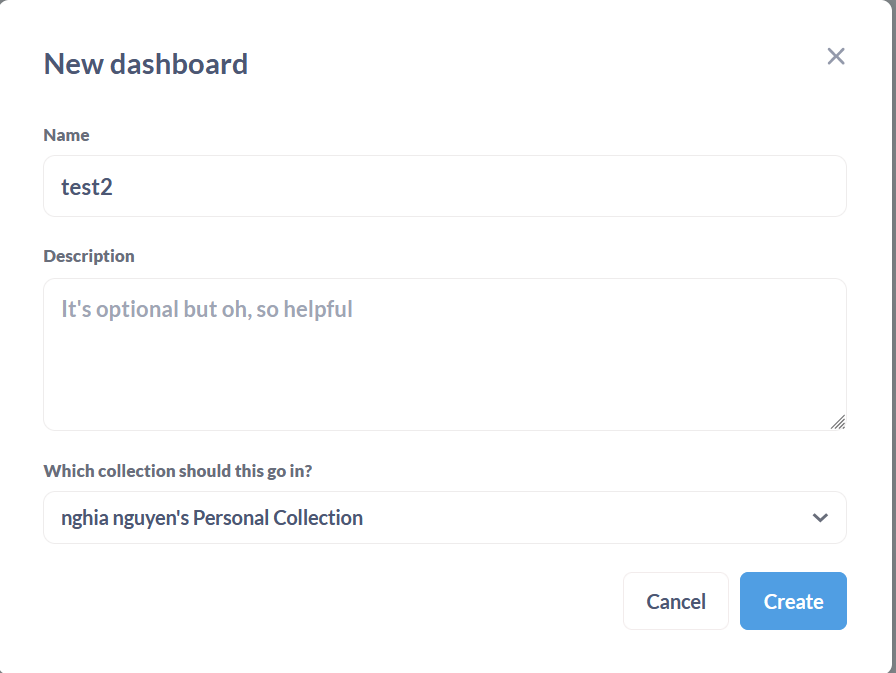
* + Click connect database -> click finish -> go to meatabase
  + After sign in with data successfully. You will wee:



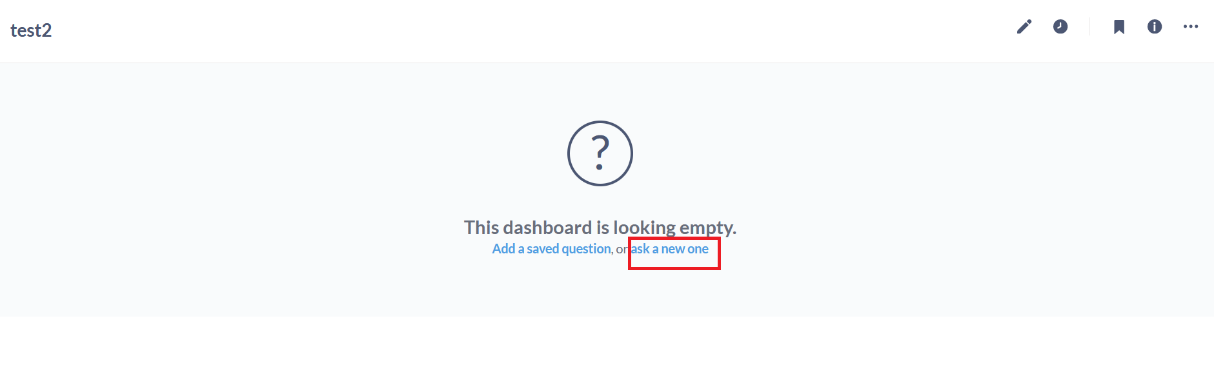
* + Click ‘browse data’, if your database successfully add, you will see:



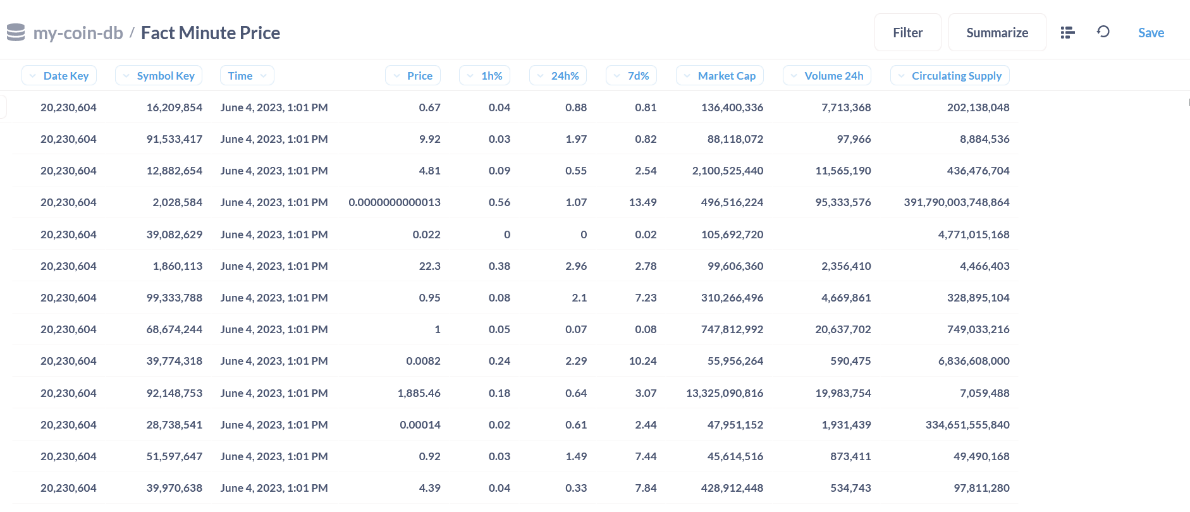
* + Choose my-coin-db
  + Now you can check your table:
    - dim\_date,
    - dim\_symbol
    - fact\_minute\_price
    - fact\_hour\_price
    - fact\_day\_price.
  + Note when check fact\_hour\_price and fact\_day\_price you will not see anything because these table store data of last hour before and last day before. In the next hour, you can see data in fact\_hour\_price and in the next day you can see data in fact\_day\_price.
  + Here, I will help you make an easy dashboard to monitor change in price with a filter to select wanted coin:
    - Click button ‘New’ in top right of screen.
    - A list pop down appear. Then choose Dashboard.
    - Enter name you want and choose where you want to save this dashboard:



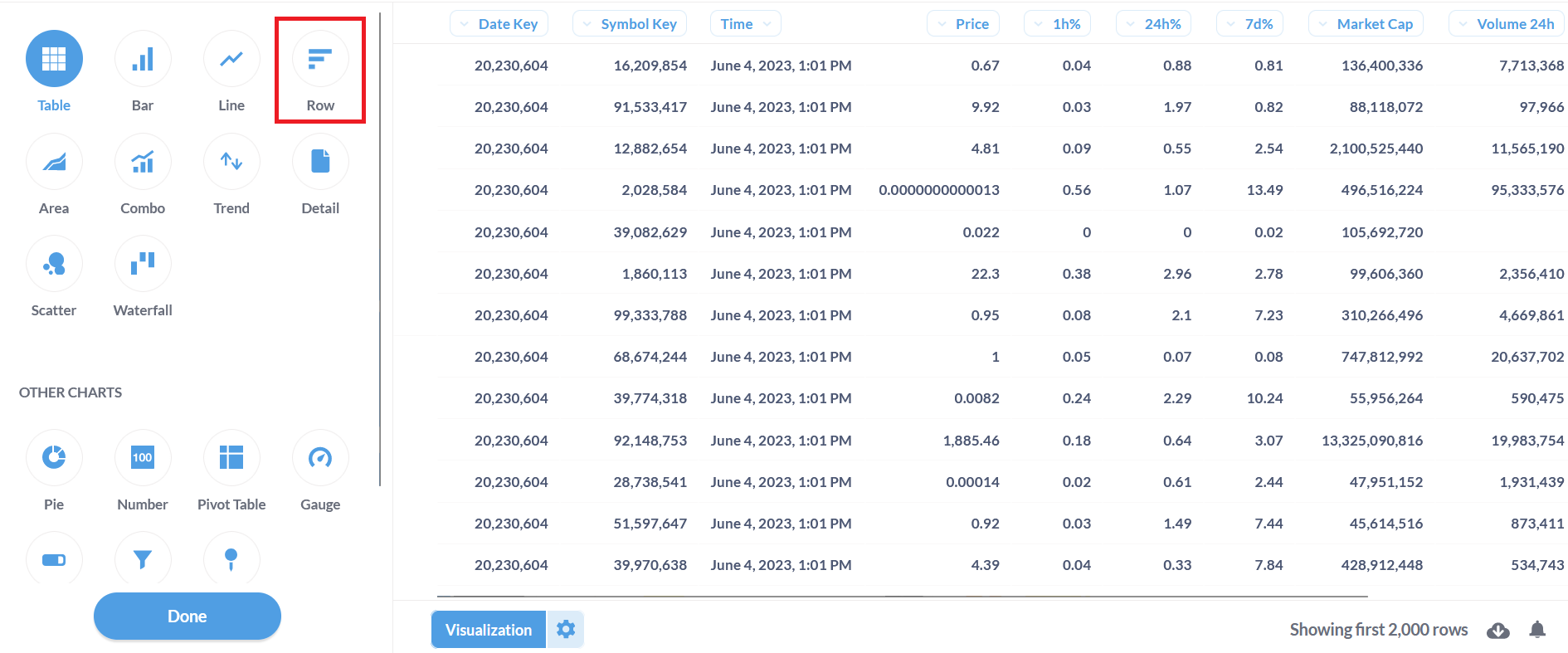
* + - Select ‘Create’
    - Click ‘ask a new one’:



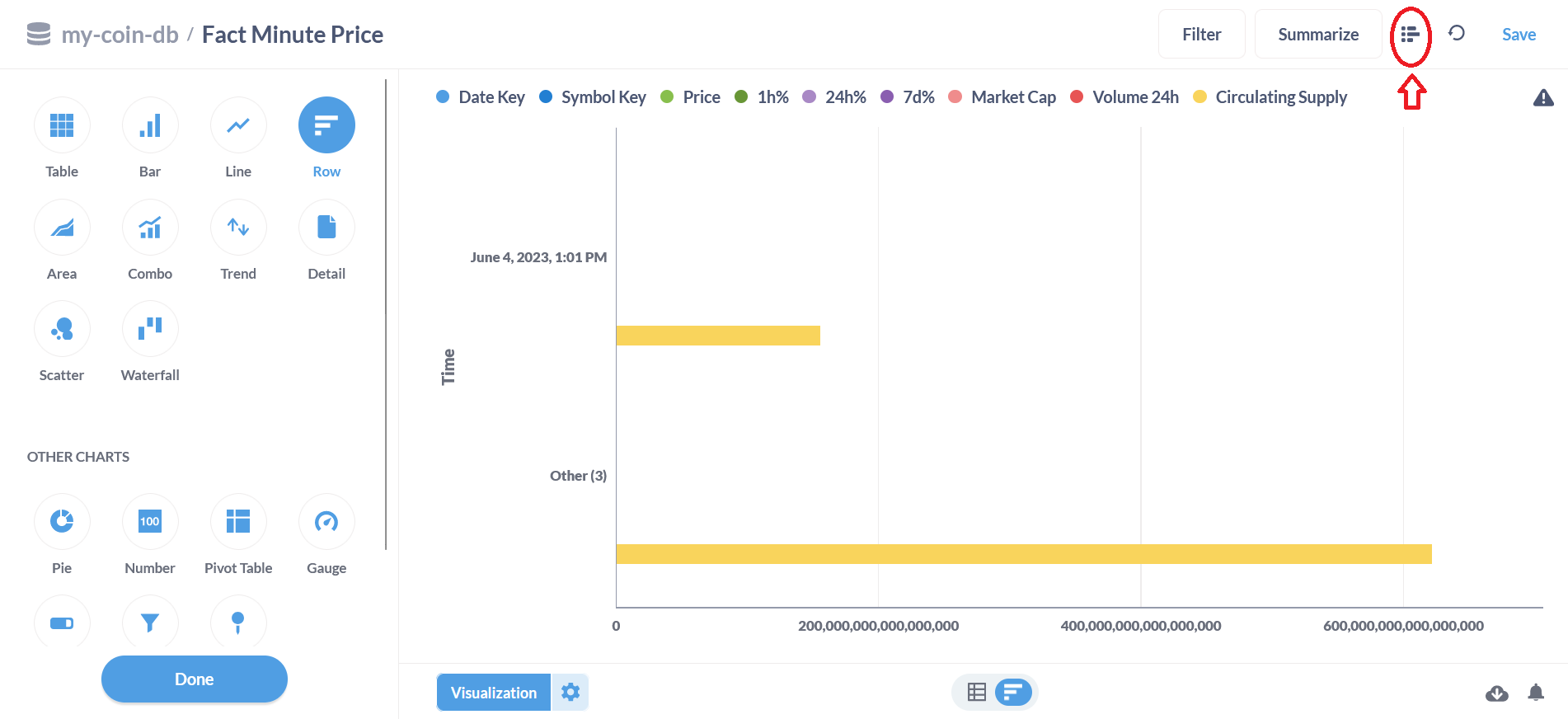
* + - Choose my-coin-db -> Fact Minute Price. You will see:



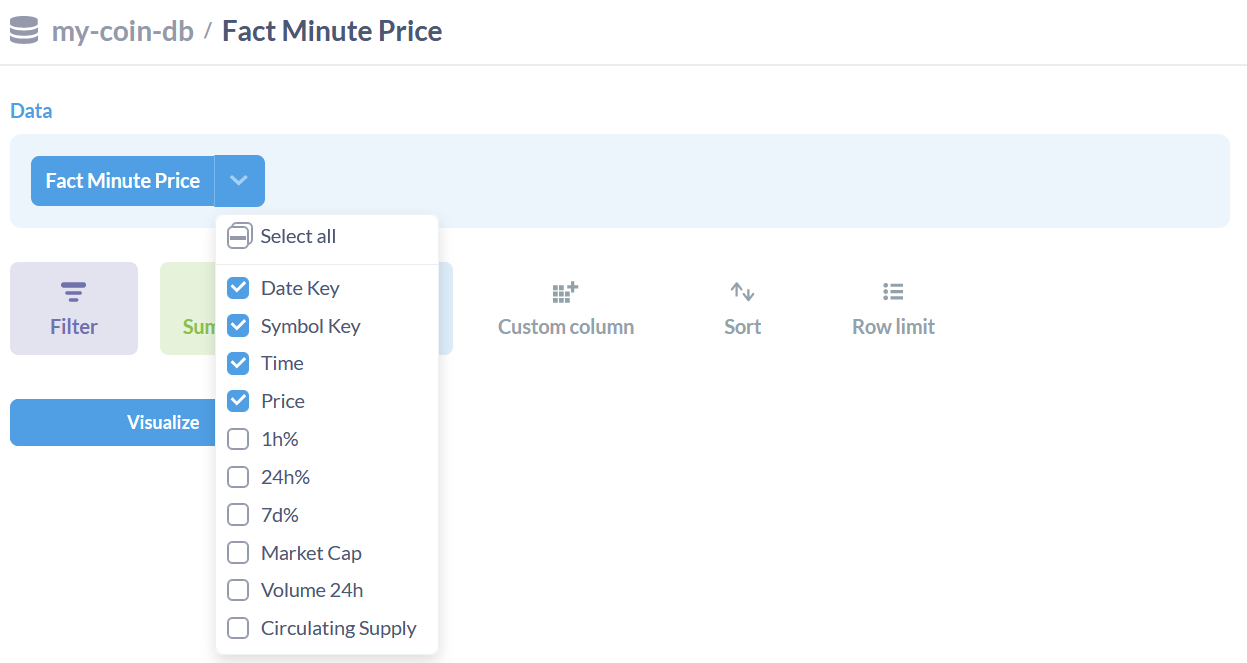
* + - Choose ‘Visuallization’ in bottom left
    - Choose ‘Row’:



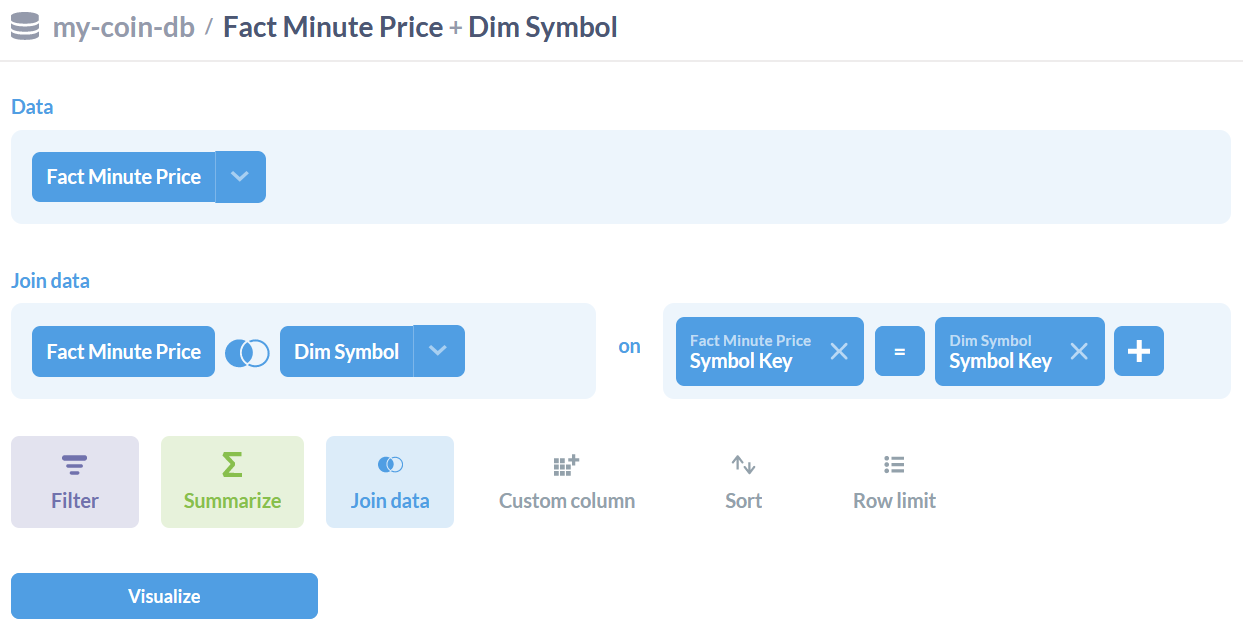
* + - Now we need modify data set: Choose editor in top right:



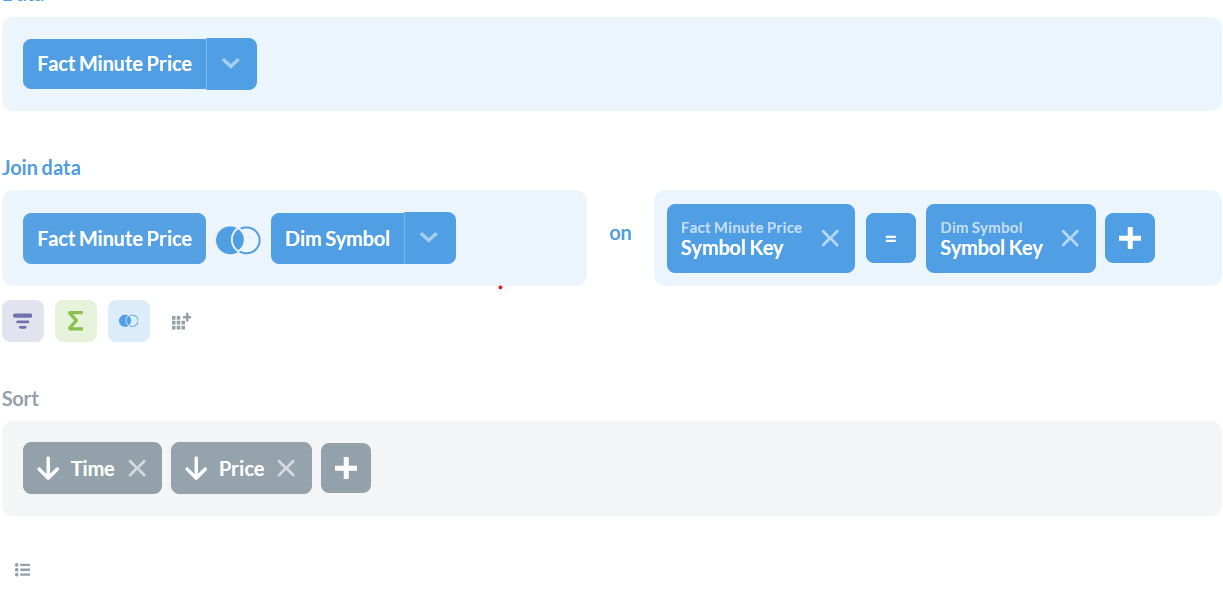
* + - Select only needed data:



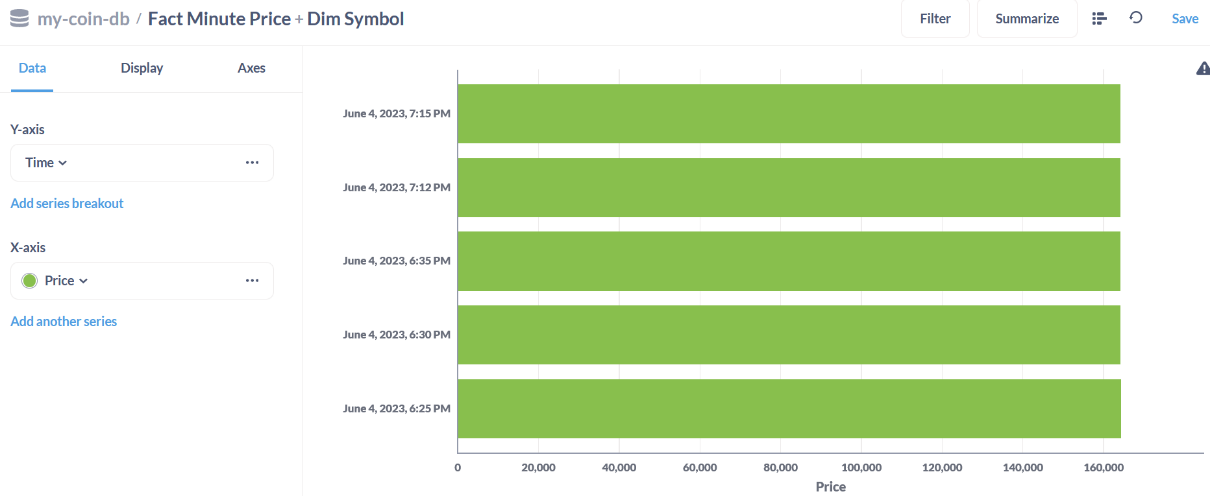
* + - Click ‘join data’:



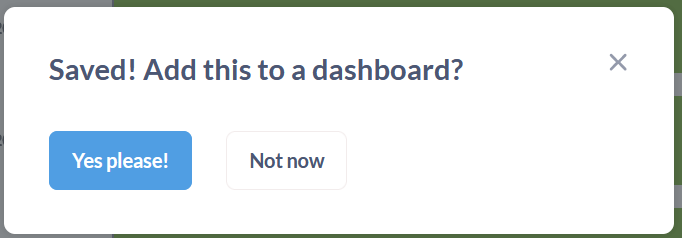
* + - Click ‘Sort’ and sort descending by time, price



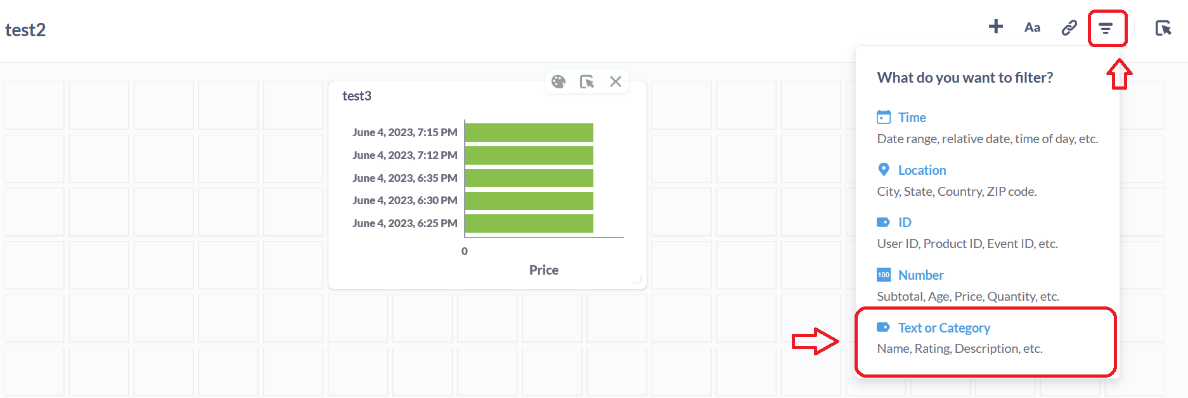
* + - You can limit number of rows by icon below and enter number of row into it. You should check number of row in your database first.
    - Click ‘Visualize’
    - To set dimension for chart, click setting symbol in bottom left
    - Choose ‘Data’ then set dimension for chart:



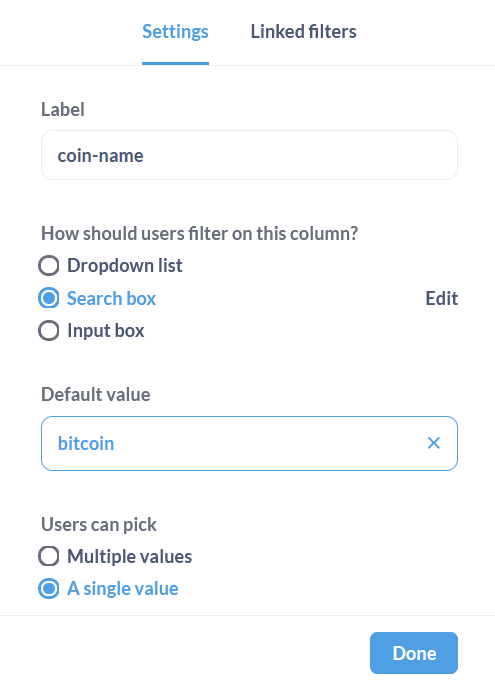
* Choose ‘Display’ then choose ‘show values on data points’
  + - Choose ‘Save’ in top right.
    - Enter name for it and attach it to a dashboard.
    - Choose ‘Yes please’



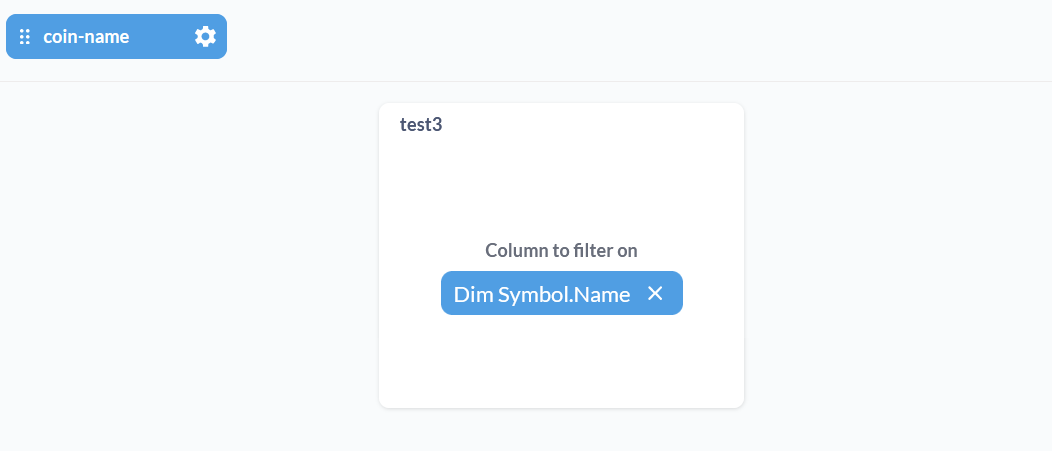
* + - Choose dashboard you want attach to.
    - Now we need add a filter for this dash board:
      * Choose ‘Add a filer’ and select ‘Text or Category’:



* Select type ’Is’ then config it like:



* Set column for filter:



* Click ‘Save’
* Now you can see dashboard:

